software AG
A SOFTWARE GMBH BRAND

Natural

Database Management System Interfaces

Version 8.2.8

November 2024

ADABAS & NATURAL

Table of Contents

Database Management System Interfaces

This documentation provides an overview of the Natural database management system interfaces and a short summary of their functions.

The following topics are covered:

Natural for DB2	The Natural interface to DB2 enables Natural users to access data in a DB2 database. Natural for DB2 is supported under the TP monitors Com-plete, CICS, IMS TM, in batch mode, and TSO.
Natural for Db2 for zIIP	The Natural for Db2 for zIIP interface enables Natural users to run Db2 workloads on IBM System z Integrated Information Processors (zIIP).
Natural for VSAM	The Natural interface to VSAM enables Natural users to access data stored in VSAM files.
Natural for DL/I	The Natural interface to DL/I enables Natural users to access and update data stored in a DL/I database. The Natural user can be executing in batch mode or under the control of the TP monitor CICS or IMS TM.

Note: See also *Database Access* (in the *Programming Guide*) on how to access data in Adabas.

1   About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documenta-tion.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.software-ag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.

- Ask questions and find answers in our discussion forums.

- Get the latest Software GmbH news and announcements.

- Explore our communities.

- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://containers.softwareag.com/products** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.

- Search the Knowledge Center for technical information and tips.

- Subscribe to early warnings and critical alerts.

- Open and update support incidents.

- Add product feature requests.

Data Protection

Software GmbH products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

I Natural for DB2

This documentation describes the functionality and the use of Natural for DB2, which is a Natural interface required to access data in a DB2 database.

Interface Subprograms	Several Natural and non-Natural subprograms to be used for various purposes.
Natural File Server for DB2	Description of the Natural File Server for DB2 in the various supported environments.
Natural for DB2 Version 8.4 - Documentation Updates	Documentation updates that only apply to Natural for DB2 Version 8.4.

Related Documentation

For installation instructions and a description of the Natural for DB2 parameter module, refer to *Installing Natural for DB2* in the *Installation for z/OS* documentation.

For the various aspects of accessing data in a database with Natural, see also *Database Access* in the Natural *Programming Guide*.

For information on logging SQL statements contained in a Natural program, refer to *DBLOG Trace Screen for SQL Statements* in the *DBLOG Utility* documentation.

2 General Information

Purpose

Natural for DB2 is a Natural interface designed to access data in a DB2 database.

In general, there is no difference between using Natural with DB2 and using it with Adabas, VSAM or DL/I. The Natural interface to DB2 allows Natural programs to access DB2 data, using the same Natural native data manipulation (DML) statements that are available for Adabas, VSAM, and DL/I. Therefore, programs written for DB2 tables can also be used to access Adabas, VSAM, or DL/I databases. In addition, Natural SQL DML statements are available.

All operations requiring interaction with DB2 are performed by Natural for DB2.

Environment-Specific Considerations

Natural for DB2 is supported in the following environments:

- Natural for DB2 under Com-plete
- Natural for DB2 under CICS
- Natural for DB2 under IMS TM
- Natural for DB2 under TSO
- Natural for DB2 Using CAF
- Natural for DB2 Using DB2 DL/I Batch Support

Natural for DB2 under Com-plete

DB2 is supported by Com-plete. Programs running under Com-plete can access DB2 databases through the DB2 Call Attachment Facility (CAF). This facility, together with the Com-plete interface to DB2, allows fully conversational access to DB2 tables.

If the DB2 plan created during the installation process is not specified in your DB2 `SERVER` parameter list for Com-plete, you must explicitly call `NATPLAN` before the first SQL call to allocate this plan.

Natural for DB2 under CICS

CICS/DB2 Attachment Facility

Under CICS, Natural uses the CICS/DB2 Attachment Facility to access DB2. Therefore, ensure that this attachment is started. If not, the Natural session is abnormally terminated with the CICS abend code `AEY9`, which leads to the Natural error message NAT0954 if the Natural profile parameter `DU` is set to `OFF`.

CICS DB2 Plan Selection

If the Natural CICS transaction ID is not assigned to any DB2 plan in the RCT by `DB2ENTRY` and `DB2TRAN` definitions, you must explicitly execute `NATPLAN` before the first SQL call to specify the required DB2 plan and define `NDBUEXT` as dynamic plan selection exit (`PLANExit` attribute). The actual plan allocation is performed by the dynamic plan selection exit.

CICS Pseudo-Conversational Mode

Under CICS, a Natural program usually runs in pseudo-conversational mode (Natural profile parameter `PSEUDO` set to `ON`, default value). In this case, at the end of the CICS transaction, the DB2 transaction is commited and all open DB2 cursors are closed implicitly and there is usually no way to resume open Natural `FIND`/`SELECT` database access loops after the terminal I/O.

To circumvent the problem of CICS terminating a pseudo-conversational transaction during loop processing and thus causing DB2 to close all cursors and lose all selection results, Natural for DB2 uses the file server to support the Natural transaction logic. If you intend to operate in CICS pseudo-conversationl mode, specify the Natural for DB2 parameter `FSERV=ON` and provide a file server file in the CICS region.

CICS Conversational Mode

If you do not provide a file server file in the CICS region and the Natural for DB2 parameter `CONVERS` is set to `ON`, Natural for DB2 switches to conversational mode whenever a terminal I/O takes place during an open database loop. This means the CICS transaction is spawned across terminal I/O as long as there are open database loops. This could cause DB2 deadlocks, as DB2 resources are allocated across terminal I/Os.

CICS Conversational Mode 2

In order to support applications, which do not deploy the implicit commit at CICS terminal I/O and which instead code explicit `ROLLBACK` or `COMMIT` to end their database transaction, a conversational mode 2 has been introduced.

Conversational mode 2 means that a DB2 update transaction is spawned across CICS terminal I/Os until an explicit `COMMIT` or `ROLLBACK` is issued.

Conversational mode 2 could be requested by the Natural for DB2 parameter `CONVRS2=ON` or it can dynamically set or rest by calling the `CALLNAT` program `NDBCONV`.

> 🔴 **Caution:** These kinds of application tend to tie up CICS and DB2 resources, as the resources are not freed across terminal I/O!

File Server under CICS

The usage of the file server depends on the `FSERV` parameter in the `NTDB2` macro.

In a CICS environment, the file server is an optional feature to relieve the problems of switching to conversational processing. Before a screen I/O, Natural detects if there are any open cursors and if so, saves the data contained by these cursors into the file server. With the file server, database loops can be continued across terminal I/Os, but database modifications made before a terminal I/O can no longer be backed out.

For a detailed description of the file server, refer to the section *Natural File Server for DB2*.

Natural for DB2 under IMS TM

Under IMS TM, Natural uses the IMS DB2 Attachment Facility to access DB2. Therefore, ensure that this attachment is started.

In IMS TM transaction processing environments, DB2 closes all cursors and thereby loses all selection results whenever the program returns to the terminal to send a reply message. This operation mode is different from the way DB2 works in CICS conversational mode or TSO environments, where cursors can remain open across terminal communication and therefore selected rows can be retained for a longer time.

File Server under IMS TM MPP

The usage of the file server depends on the `FSERV` parameter in the `NTDB2` macro.

The file server is required to support the Natural for DB2 cursor management, while IMS TM issues an implicit end-of-transaction to DB2 after each terminal I/O operation. With the file server, database loops can be continued across terminal I/Os, but database modifications made before a terminal I/O can no longer be backed out.

For a detailed description of the file server, refer to the section *Natural File Server for DB2*.

Natural for DB2 under TSO

Natural for DB2 can run under TSO without requiring any changes to the Natural/TSO interface.

Apart from z/OS Batch, the batch environment for Natural can also be the TSO background, which invokes the TSO terminal monitor program by an `EXEC PGM=IKJEFT01` statement in a JCL stream.

Both TSO online or batch programs can be executed either under the control of the DSN command or by using the Call Attachment Facility (CAF); the CAF interface is required if plan switching is to be used.

File Server under TSO

The usage of the file server depends on the `FSERV` parameter in the `NTDB2` macro.

In a TSO environment, the file server is an optional feature to be able to emulate during development status a future CICS or IMS TM production environment.

With each terminal I/O, Natural issues a `COMMIT WORK` command to simulate CICS or IMS TM Syncpoints. Therefore, database modifications made before a terminal I/O can no longer be backed out.

For a detailed description of the file server, refer to the section *Natural File Server for DB2*.

Natural for DB2 Using CAF

If you run Natural for DB2 under TSO or in batch mode and use the CAF interface, you must explicitly call `NATPLAN` before the first SQL call to allocate the required DB2 plan.

`NATPLAN` can be edited to specify the appropriate DB2 subsystem ID.

Natural for DB2 Using DB2 DL/I Batch Support

If you want to access DB2 and DL/I in the same Natural session in batch mode (not BMP), you can use the DB2 DL/I batch support facility, which allows you to coordinate recovery of both DB2 and DL/I database systems.

If you want to use this facility, you must execute the `DLIBATCH` procedure to run the `DSNMTV01` module as the application program. `DSNMTV01` in turn executes the Natural batch nucleus which must be linked with the DB2 interface `DFSLI000`.

If your PSB is generated with `CMPAT=YES`, all Syncpoints are executed and you must issue an `END TRANSACTION` statement before you end your Natural session; otherwise, any database modifications are lost, because Natural implicitly issues a `BACKOUT TRANSACTION` statement at the end of the session.

If your PSB is generated with `CMPAT=NO`, all Syncpoints are ignored.

Integration with Predict

Predict, Software AG's open, operational data dictionary for fourth-generation-language development with Natural, is a central repository of application metadata and provides documentation and cross-reference features. Predict lets you automatically generate code from definitions, enhancing development and maintenance productivity.

Since Predict supports DB2, direct access to the DB2 catalog is possible via Predict and information from the DB2 catalog can be transferred to the Predict dictionary to be integrated with data definitions for other environments.

DB2 databases, tables and views can be incorporated and compared, new DB2 tables and views can be generated, and Natural DDMs can be generated and compared. All DB2-specific data types and the referential integrity of DB2 are supported. See the relevant Predict documentation for details.

In addition, the Predict active references support static SQL for DB2 as described in *WITH XREF Option* in *Preparing Programs for Static Execution*.

Integration with Natural Security

When run in an environment that is controlled by Natural Security, the use of certain features of Natural for DB2 can be restricted by the security administrator, for example:

- **Natural Tools for DB2**

  Access to the Natural system library `SYSDB2`

  Individual functions

- **Static SQL**

  Static generation can be disallowed by

  - restricting access to the Natural system library `SYSDB2`,

  - disallowing the module CMD,

  - restricting access to the libraries that contain the relevant Natural objects,

  - disallowing one of the Natural system commands `CATALOG` or `STOW` for a library that contains relevant Natural objects.

  If a library is defined in Natural Security and the DBID and FNR of this library are different from the default specifications, the static generation procedure automatically switches to the DBID and FNR specifications defined in Natural Security.

For further information, ask your security administrator.

Incompatibilities and Constraints

This section lists the known incompatibilities and constraints against DB2 when using Natural for DB2 to access data from DB2.

■ **Data Type DECIMAL or NUMERIC**

Most SQL database systems support packed decimal numbers with a maximal precision of 31 digits and a scale (fractional part of the number) of up to 31 digits. The scale has to be positive and not greater than the precision. Natural allows precision and scale of up to 29 digits.

Messages Related to DB2

The message number ranges of Natural system messages related to DB2 are 3275 - 3286, 3700-3749, and 7386-7395.

For a list of error messages that may be issued during static generation, see *Static Generation Messages and Codes Issued under NDB* in the Natural *Messages and Codes* documentation.

Terms Used in this Documentation

Term	Explanation
DB2	DB2 refers to IBM's DB2 UDB for z/OS.
DBRM	Database request module
DDM	Data definition module.
DML	Data manipulation language (Natural).
File Server	In this document, the term "file server" refers to the **Natural file server** for DB2.
NDB	This is the product code of Natural for DB2. In this documentation the product code is often used as prefix in the names of data sets, modules, etc.

3 Accessing a DB2 Table

⟫ **To enable access to a DB2 table with a Natural program**

1 Use the Natural Tools for DB2 to define a DB2 table; see *Using Natural Tools for DB2*.

2 Use Predict or the **SQL Services** function of the Natural SYSDDM utility to create a Natural data definition module (DDM) of the defined DB2 table.

3 Once you have defined a DDM for a DB2 table, you can access the data stored in this table by using a Natural program.

Natural for DB2 translates the statements of a Natural program into SQL statements.

Natural automatically provides for the preparation and execution of each statement. In dynamic mode, a statement is only prepared once (if possible) and can then be executed several times. For this purpose, Natural internally maintains a table of all prepared statements (see *Statement Table* in *Internal Handling of Dynamic Statements*).

Almost the full range of possibilities offered by the Natural programming language can be used for the development of Natural applications which access DB2 tables. For a number of Natural DML statements, however, there are certain restrictions and differences as far as their use with DB2 is concerned; see *Using Natural Native DML Statements*. In the *Statements* documentation, you can find notes on Natural usage with DB2 attached to the descriptions of the Natural DML statements concerned.

As there is no DB2 equivalent to Adabas internal sequence numbers (ISNs), any Natural features which use ISNs are not available when accessing DB2 tables with Natural.

For SQL databases, in addition to the Natural native DML statements, Natural provides Natural SQL statements; see *Using Natural SQL Statements*. They are listed and explained in the *Statements* documentation.

4 Using Natural Tools for DB2

This section describes how to invoke Natural Tools for DB2 and maintain DB2-specific objects and SQL statements. In addition, this section provides information on global PF-key settings and global maintenance commands within Natural Tools for DB2.

📄 **Notes:**

1. See also *Special Requirements for Natural Tools for DB2* in *Installing Natural for DB2 on z/OS*.

2. If you have created a new SYSDB2 library when installing Natural for DB2, ensure that it contains all Predict interface programs necessary to run the Natural Tools for DB2. These programs are loaded into SYSDB2 at Predict installation time (see the relevant *Predict* documentation).

Invoking Natural Tools for DB2

≫ **To invoke Natural Tools for DB2**

■ Enter the Natural system command SYSDB2

The Natural Tools for DB2 **Main Menu** is displayed, which offers you the functions listed below.

```
15:04:05              ***** NATURAL TOOLS FOR DB2 *****              2009-11-27
                            - Main Menu -



                       Code Function

                        A    Application Plan Maintenance
                        C    Catalog Maintenance
                        I    Interactive SQL
                        R    Retrieval of System Tables
                        S    Environment Setting
                        X    Explain PLAN_TABLE
                        F    File Server Statistics
                        D    DB2 Commands Execution
                        ?    Help
                        .    Exit

                Code .. _



Command ===>

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                Canc  ↵
```

Main Menu Functions

Function	Description
Application Plan Maintenance	Maintain DB2 application plans online.
Catalog Maintenance	Maintain the DB2 catalog.
Interactive SQL	Process SQL statements that are not embedded.
Retrieval of System Tables	Display/print DB2 objects and user authorizations.
Environment Setting	Execute SQL statements and display special register values.
Explain PLAN_TABLE	Interpret your PLAN_TABLE.
File Server Statistics	Display statistics on the generation and use of the file server.
DB2 Commands Execution	Issue DB2 commands from Natural.

Editing within the Natural Tools for DB2

The free-form editor available within the Natural Tools for DB2 requires that the Software AG Editor is installed. The main and line commands available for use within the Natural Tools for DB2 are a subset of those available within this editor.

Both main commands and line commands are described in detail as part of the Natural Tools for DB2 online help facility, which is invoked by pressing PF1 (Help). For further details, please refer to the Software AG Editor documentation.

Overview of Editor Main Commands

Main commands are entered in the command line of the editor screen. The most important main commands are:

Command	PF Key	Description
BOTTOM (++)		Positions to the bottom of the data.
CHANGE		Scans for a specified string and replaces each such string found with another specified string.
CLEAR		Clears the editor source area.
DELETE		Deletes the line(s) containing a given string according to the specified selection operands.
DOWN (+)	PF8	Scrolls the specified scroll amount downwards.
FIND		Finds a string specified by command operands at the location(s) specified by selection operands.
LEFT	PF10	Scrolls the specified scroll amount to the left.
LIMIT n		Sets a limit for the FIND command; n lines are processed.
PRINT		Prints the data displayed.

Command	PF Key	Description
RESET		Resets all pending line commands.
RFIND	PF5	Repeats the last FIND command.
RIGHT	PF11	Scrolls the specified scroll amount to the right.
TOP (--)		Positions to the top of the data.
UP (-)	PF7	Scrolls the specified scroll amount upwards.

The scroll amount for the UP, DOWN, LEFT, and RIGHT commands is specified in the SCROLL field at the top right corner of the list screen. Valid values for the scroll amount are:

Value	Explanation
CSR	Scroll amount is determined by cursor position
DATA	Scroll amount equals the page size less one line
HALF	Scroll amount is half the page size
MAX	Scroll amount equals the amount of data to the bottom/top
PAGE	Scroll amount is equal to the page size
n	Scroll amount is equal to *n* lines

Overview of Editor Line Commands

Line commands are entered in the editor prefix area of the corresponding statement line. The most important line commands are:

Command	Description
A	Inserts line(s) to be moved or copied after the current line.
B	Inserts line(s) to be moved or copied before the current line.
C	Copies the current line.
CC	Marks the beginning and end of a block of lines to be copied.
D	Deletes the current line.
DD	Marks the beginning and end of a block of lines to be deleted.
I*nn*	Inserts *nn* new lines after the current line.
M	Moves the current line.
MM	Marks the beginning and end of a block of lines to be moved.

Global PF-Key Settings

Within the Natural Tools for DB2, the following global PF-key settings apply:

Key	Setting	Description
PF1	Help	Pressing PF1 invokes the Natural Tools for DB2 online help system from any screen within the Natural Tools for DB2.
PF3	Exit	Pressing PF3 always takes you to the previous screen or function. When pressed on an editor screen where modifications have been made, the **Exit Function** window is displayed (as described in **Exit Function** in the sections Program Editor and Data Area Editor in the Natural *Editors* documentation). When you press PF3 on the Main Menu, you leave the Natural Tools for DB2.
PF12	Canc	Pressing PF12 always takes you back to the menu from where the current screen has been invoked. When you press PF12 on the **Main Menu**, you leave the Natural Tools for DB2.

Global Maintenance Commands

Within the Natural Tools for DB2, the following global maintenance commands apply:

Command	Description
COPY *name*	Copies the specified member from the current library into the editor, after (A) or before (B) the current line.
LIBRARY *name*	Specifies the Natural library *name* as current library.
LIST *name**	Lists all members from the current library whose names start with *name*. From the list, you can select a member by marking it with "S".
PURGE *name*	Purges the specified member from the current library.
READ *name*	Reads the specified member from the current library into the editor. The current name is set to *name*.
SAVE [*name*]	Saves the generated code as the member *name* in the current library. If no name is specified, the current name is taken. Current library and member names are displayed above the Command line.

Member and library names must correspond to the Natural naming conventions: see *Object Naming Conventions* and *Library Naming Conventions* in *Using Natural*. Members can be JCL members, SQL members, or output members.

5   Application Plan Maintenance

Introduction

The application plan maintenance part of the Natural Tools for DB2 is used to generate JCL code to:

- create database request modules (DBRMs) from your Natural programs,

- maintain DB2 application plans and packages from within your Natural environment.

Two modes of operation are available: fixed mode and free mode.

Fixed Mode

In fixed mode, maintenance screens with syntax graphs help you to specify the correct commands. Complete JCL members can be generated using predefined job profiles. You simply enter the required data in input maps. The data are checked to ensure that they comply with the correct syntax. Then JCL members are generated from these data. The members can be submitted directly by pressing PF4 (Submi). But you can also switch to free mode by pressing PF5 (Free).

Free Mode

Pressing PF5 in fixed mode invokes the free-mode editor, which can be used to modify JCL code generated in fixed mode, without the syntactical restrictions imposed. In free mode you can submit the JCL member currently in the source area by pressing PF4 (as in fixed mode).

Invoking the Application Plan Maintenance Function

≫ **To invoke the Application Plan Maintenance function**

■ On the **Natural Tools for DB2 Main Menu**, enter function code A.

The **Application Plan Maintenance** menu is displayed:

```
 16:14:02                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                         - Application Plan Maintenance -

                   Code Function               Parameter

                    PP  Prepare Job Profile
                    CD  Create DBRMs           Lib
                    BI  Bind                   Lib, Obj
                    RB  Rebind                 Lib, Obj
                    FR  Free                   Lib, Obj
                    LJ  List   JCL             Lib, JCL
                    JO  Display Job Output     Node
                    ?   Help
                    .   Exit

              Code .. __   Object ...... _____
                           Library ..... SAG_____
                           JCL Member .. _____
                           Node ........ 148

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit                                          Logn        Canc
```

The following functions are available:

Code	Description
PP	Defines job profiles for DBRM creation and plan/package maintenance; see *Prepare Job Profiles*.
CD	Generates JCL to create database request modules.
BI	Generates JCL to bind a plan or package.
RB	Generates JCL to rebind a plan or package.
FR	Generates JCL to free a plan or package.
LJ	Invokes the free-mode editor.
JO	Displays job output. **Note:** This function only applies if the Entire System Server is installed.

In addition, four parameters are available, which must be specified according to the selected function:

Parameter	Description
Object	Specifies whether to maintain a plan (`PLAN` or `PL`) or a package (`PACKAGE` or `PK`).
Library	Specifies the name of a Natural source library.
	All existing libraries except the ones beginning with `SYS` can be specified; a library must be specified for JCL maintenance. The library name is preset with your Natural user ID.
JCL Member	If a valid member name is specified, the corresponding JCL member is displayed.
	If a value is specified followed by an asterisk (*), all JCL members in the specified library whose names begin with this value are listed.
	If asterisk notation is specified only, a selection list of all JCL members in the specified library is displayed.
	If the JCL Member field is left blank, the empty free-mode editor screen is displayed.
Node	Specifies the number of the node to be used by the Entire System Server. The default number "148" can be overwritten.

Commands and PF-Key Settings

Within the maintenance screens in fixed mode, various windows can be invoked. These windows are accessed via 1-byte control fields.

≫ **To invoke a window**

■   Enter `S` in the corresponding control field.

If the control field displays an `X`, data have already been entered in the corresponding window.

In addition, the following PF-key settings apply in fixed mode:

Key	Function
PF4	Generates JCL code and submits it.
PF5	Generates JCL code and enters free mode.
PF6	Scrolls to the top of a window.
PF7	Scrolls backwards in windows.
PF8	Scrolls forwards in windows.
PF9	Scrolls to the bottom of a window.
PF10	Either shows the previous screen (<) or displays a **Natural Process Logon** window (Logn).
PF11	Shows the next screen.

In free mode, JCL code can be edited and submitted. Editing of JCL code is done via edit and line commands; see *Editing within the Natural Tools for DB2*.

Generated JCL code is submitted by pressing PF4.

Apart from being submitted, JCL code can also be copied, listed, purged, retrieved from, or saved in a Natural library. All this is done via maintenance commands; see *Global Maintenenance Commands*.

Prepare Job Profile

If you want to generate JCL to create a DBRM or to bind, free, or rebind a plan or package, you have to specify a job name, job cards, and the name of a job profile. Thus, you have to prepare the job profiles first. Once your job profiles are defined, you can always immediately select the corresponding function if you want to create a new DBRM or if you want to bind, free, or rebind an a plan or package using your predefined job profiles.

⟫ To define a job profile

1   On the **Natural Tools for DB2 Main Menu**, enter function code A.

    The **Application Plan Maintenance** menu is displayed.

2   On the **Application Plan Maintenance** menu, invoke the **Prepare Job Profile** function by entering function code PP.

    The **Prepare Job Profile** menu is displayed.

Prepare Job Profile Menu

```
 16:14:33                  ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                             - Prepare Job Profile -




                        Code Function

                         J   Default Job Cards
                         D   Profile for Create DBRM Job
                         P   Profile for DSN Jobs
                         ?   Help
                         .   Exit

                  Code .. _     Profile .. _____

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit                                                    Canc
```

Code	Description
J	Defines user-specific default job cards.
D	Defines job profiles for the DBRM creation function.
P	Defines job profiles for the plan or package maintenance functions.

In addition, the parameter `Profile` is available, which is relevant to function codes `D` and `P` only. With function code "J", `Profile` corresponds to `USER`.

Parameter	Description
`Profile`	Specifies the name of an already existing job profile. If a valid profile name is specified, the free-mode editor with the specified job profile is invoked, where the profile can be modified and saved. If a value is specified followed by an asterisk (*), all existing job profiles whose names begin with this value are listed. If asterisk notation is specified only, a selection list of all existing job profiles is displayed. If the field is left blank, the corresponding fixed-mode profile screen is invoked, where a new job profile can be created. To save the new profile, you have to switch to free mode.

Job profiles can be maintained (that is, copied, listed, purged, retrieved from, or saved in a Natural library) via maintenance commands; see *Global Maintenance Commands*.

> **Note:** Job profiles are saved on the Natural system file `FNAT`.

Default Job Cards

All jobs generated by the **Application Plan Maintenance** function require job cards. With the **Default Job Cards** function, you can define a default job card for each user. The default job cards apply to all function screens on which you can generate JCL. Default job cards can be invoked and modified on all these screens. Asterisk notation (*) can be used to select the desired job card from a list.

≫ **To define a default job card**

■ On the **Prepare Job Profile** menu, enter function code J and press ENTER.

The **Default Job Cards** screen is displayed.

```
 16:14:33                 ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                                 - Default Job Cards -



  R ead,  S ave,  L ist or  P urge  Default Job Cards _      User ID .. _____

     Job Name ... _____
     Job Cards ..
      //       JOB _____
      // _____
      // _____
      // _____
      // _____



 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                    Canc
```

On this screen, you can create and save your user-specific job cards. To do so, you can also read (directly or from a list) and modify an already existing default job card. Existing job cards can be purged, too.

📄 **Note:** All other function screens used to specify jobs contain the same two fields - **Job Name** and **Job Cards** - as the **Default Job Cards** screen. Thus, it is possible to override the default job cards in each of these screens, too.

≫ **To modify the job name**

■ Enter the new job name in the **Job Name** field and press ENTER.

≫ **To modify the job cards**

■   In the **Job Cards** field, enter an `S` and press ENTER.

A window is displayed where you can modify all the job cards.

Profile for Create DBRM Job

The **Profile for Create DBRM Job** function enables you to define profiles for the **Create DBRMs** functions. Job profiles for DBRM creation consist of JCL which includes the following predefined set of substitution parameters:

Parameter	Description
@JOBCARDS	Is replaced by the job cards entered on the **Create DBRMs** screen (up to five lines). You can also code the job cards in the profile and omit the job cards modifier.
@COMMAND	Is replaced by the string `CREATE DBRM`.
@DBRMNAME	Is replaced by the name of the DBRM, which can be up to eight characters long.
@CREATE-DBRM	Is replaced by the command input for the static generation step. This parameter must be placed *after* the `//CMSYNIN` card and must comply with the Assembler naming conventions.
@COMMAN2	Is replaced by the string `MODIFY`.
@MODIFY	Is replaced by the command input for the static modification step.
@XR-START @XR-END	Both mark the JCL to contain the Natural Assembler `XREF` data; if no `XREF` option is specified, the JCL is deleted again.

≫ **To modify or rename a job profile for DBRM creation**

1   On the **Prepare Job Profile** menu, invoke the **Profile for Create DBRM Job** function by entering function code `D`.

2   In the **Profile** field, specify a valid profile name and press ENTER.

The free-mode editor containing the specified profile is invoked, where you can modify, save, and rename the displayed profile.

≫ **To create a job profile for DBRM creation**

1   On the **Prepare Job Profile** menu, enter function code `D`.

2   Leave the **Profile** field blank, and press ENTER.

The **Profile for Create DBRM Job** screen is invoked, which helps you in creating a new profile.

```
 16:15:18                  ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                           - Profile for Create DBRM Job -


 +------------------------- NATURAL Parameters -------------------------+
 !  Name of Batch NATURAL     : _____                               ↵
 !
 !  NATURAL Parameter       : _____ !
 !  STEPLIB  DD              : _                                          ↵
 !
 +------------------------ Precompile Parameters ------------------------+
 !  DBRMLIB  DD              : _____ !
 !  STEPLIB  DD              : _                                          ↵
 !
 +------------------------ Ass-Nat-Xref-Library ------------------------+
 !  CMWKF02  DD              : _____ !
 +----------------------------------------------------------------------+






 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit        Free                                      Canc
```

≫ **To save the newly created job profile**

■   Switch to free mode by pressing PF5.

Profile for DSN Jobs

The **Profile for DSN Jobs** function enables you to define profiles for the **Bind**, **Rebind**, and **Free** functions. The same profiles can be used for each of the three functions.

Profiles for DSN jobs consist of JCL which includes the following predefined set of substitution parameters:

Parameter	Description
@JOBCARDS	Is replaced by the current job cards; you can also code the job cards in the profile and omit the job cards modifier.
@DSNCMD	Is replaced by the command input for the bind, rebind or free function.
@PLANNAME	For the bind function, it is replaced by the name of the plan or package. For the rebind and free functions, it is set to blank.
@COMMAND	Is replaced by the string BIND, REBIND or FREE, respectively.

> **To modify or rename a profile for DSN jobs**

1   On the **Prepare Job Profile** menu, enter function code P.

2   In the **Profile** field, specify a valid profile name, and press ENTER.

The free-mode editor containing the specified profile is invoked, where you can modify, save, and rename the displayed profile

> **To create a new profile for DSN jobs**

1   On the Prepare Job Profile menu, enter function code P.

2   Leave the **Profile** field blank, and press ENTER.

The **Profile for DSN Jobs** screen is invoked, which helps you in creating a new profile for DSN jobs.

```
 16:15:18                 ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                               - Profile for DSN Jobs -


  DB2 System .. _____                                    Retries .. ___

  +--------------------- Steplibs for DSN Jobs --------------------------+
  ! STEPLIB    DD            : _____ !
  !                          : _____ !
  !                          : _____ !
  !                          : _____ !
  !                          : _____ !
  +---------------------------------------------------------------------+


  +--------------------- DBRM Libraries for Bind ------------------------+
  ! DBRMLIB    DD            : _____ !
  !                          : _____ !
  !                          : _____ !
  !                          : _____ !
  !                          : _____ !
  +---------------------------------------------------------------------+


  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit        Free                                       Canc
```

> **To save the newly created job profile**

■   Switch to free mode by pressing PF5.

Loading Job Profiles

Job profiles for DBRM creation and plan/package maintenance are loaded from the data set `CMWKF01` in batch mode.

≫ **To load a job profile**

1 Logon to library `SYSDB2`.

2 In the command line, issue the command `LOADPROF`.

The **Load Job Profiles** menu is displayed.

```
 16:53:20                 ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                             - Load Job Profiles -




                        Code Function
                        ---- ----------------------------
                         D   Load Profile for Create DBRM
                         B   Load Profile for DSN Jobs
                         .   Exit
                        ---- ----------------------------
                  Code .. _   Profile .. _____
                             Replace .. N




 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit                                                      Canc
```

The following functions are available:

Code	Description
D	Serves to load job profiles for DBRM creation.
B	Serves to load job profiles for plan or package maintenance.

The following parameters apply:

Parameter	Description	
`Profile`	Specifies the name of the profile to be loaded. This parameter must be specified.	
`Replace`	Specifies whether it is to be replaced or not if a profile with the specified name already exists.	
	`Y`	An already existing profile is replaced.
	`N`	An already existing profile is *not* replaced. This parameter is optional; the default setting is `N`.

Unloading Job Profiles

Job profiles for DBRM creation and plan/package maintenance are unloaded and written to the data set `CMWKF01` in batch mode.

≫ To unload a job profile

1   Logon to library `SYSDB2`.

2   In the command line, issue the command `UNLDPROF`.

The **Load Job Profiles** menu is displayed.

```
 16:53:20                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              Load Job Profiles




                        Code  Function

                          D    Unload Profile for Create DBRM
                          B    Unload Profile for DSN Jobs
                          .    Exit

                 Code .. _    Profile .. _____




  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                    Exit                                                     Canc
```

The following functions are available:

Code	Description
D	Unloads job profiles for DBRM creation.
B	Unloads job profiles for plan or package maintenance.

The following parameter applies:

Parameter	Description
Profile	Specifies the name of the profile to be unloaded. This parameter must be specified.

Create DBRMs

To create a DBRM, you have to generate JCL for DBRM creation.

> **To create a DBRM**

1   On the **Application Plan Maintenance** menu, enter function code CD, and press ENTER.

The **Create DBRM** screen is displayed where, in addition to a job name, your user-specific default job cards, and the desired job profile, you can specify all necessary information for the CREATE DBRM and MODIFY commands; see also *Generation Procedure: CMD CREATE Command* and *Modification Procedure: CMD MODIFY Command* in the section *Preparing Programs for Static Execution*.

```
 16:15:44                  ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                               - Create DBRM -

  Job Name ... DBRMJOB_          Job Cards .. X              Profile ..EXDBRM__


  >>-- CREate DBRM -- DBRM1___ -- USing --+-- _ -- PREDict DOCumentation --+-->
                                          +-- _ -- INput DAta ------------+


   >-+---------------------+------+--------------------+-----+------------+->
     +- With XRef - _____ -+      +- LIBrary - _____ -+    +- FS - ___ -+
          ( NO, YES, FORCE )              !               ( ON, OFF )

  >---------+-------------------------------------------------+---------->< 
         +---- _ --- NAT Library , NAT Member +--------------+-+
                                          + , excl.Member-+


  >>------------------------- MODify-+-----------+----------------------><
                              +- _ - XRef -+

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Submi Free                                      Canc
```

2 In the **Job Name** field, a valid job name must be specified. If you only want to change the name of the job, you can do this using the **Job Name** field, too.

3 Via the **Job Cards** field, you can override your default job cards. To do so, enter an S in the **Job Cards** field.

A window containing your job cards is displayed.

An "X" in the **Job Cards** field indicates that job cards for DBRM creation are defined. A blank **Job Cards** field indicates that no job cards are defined.

4 In the **Profile** field, you can specify the name of a valid job profile for DBRM creation. If a value is specified followed by an asterisk (*), all existing job profiles whose names begin with this value are listed. If asterisk notation is specified only, a selection list of all available job profiles is displayed.

5 If you use the **INput DAta** option, a window is displayed, where you have to specify the Natural libraries and programs (members) to be contained in the DBRM.

```
 16:15:44              ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - Create DBRM -


Job Name ... DBRMJOB_          Job Cards .. X             Profile .. EXDBRM__


 >>-- CREate DBRM -- DBRM1___ -- USing --+-- _ -- PREDict DOCumentation --+-->
                                         +-- _ -- INput DAta ------------+


 >-+--------------------+------+--------------------+-----+------------+->
   +- With XRef - _____ -+       +- LIBrary - _____ -+     +- FS - ___ -+
         ( NO, YES, FORCE )                   !               ( ON, OFF )


 >---------+------- +--------------------------------------+ ---------->< 
         +---- S  ! NAT Library,NAT Member,excl.Member  1 / 2  !
                  !      Test____ , PROG1___ , _____          !
                  !      Test____ , P*_____ , PROG1___          !
 >>--------------- !     _____ , _____ , _____        ! ---------><
                  !     _____ , _____ , _____          !
                  !     _____ , _____ , _____          !
Command ===>      !                                           !
Enter-PF1---PF2---P +------------------------------------------------+ F11--PF12---
     Help      Exit  Submi Free  --    -     +    ++                      Canc
```

In the third column of the above window, you can specify a program that is to be excluded
from the DBRM; this is possible only if you specify an asterisk (*) with the program name in
the second column.

Within the window, you can scroll using PF6 (--), PF7 (-), PF8 (+), or PF9 (++).

The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free),
or submitted immediately by pressing PF4 (Submi).

Bind Plan

To generate JCL to bind a plan, you have to invoke the **Bind** function. All parameters necessary
to bind a plan are entered on four screens, which show the syntax of the DB2 BIND PLAN command.

> **To generate JCL to bind a plan**

1   On the **Application Plan Maintenance** menu, enter function code BI.

    In the **Object** field, enter PLAN or PL, and press ENTER.

    The first **Bind Plan** screen is displayed, where all necessary information must be specified.

```
 23:16:38                 ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                                 - Bind Plan -

  Job Name ... BINDJOB_         Job Cards .. X            Profile .. EXBIND1_


   >>- BIND +-----+-----------+-+-----------------+----------------------+>
          !                   ! !                 ! !                    ↵
 !
          + PLAN ( TESTPLAN )+ + OWNER ( _____ )+ + QUALIFIER ( _____ )+
                 plan-name            auth-id                  qualifier-name

    >-+->-- MEMBER +- X ---(member name)---+------------------------------+-+
      !                                    !                              ! !
      !                                    +- LIBRARY -- _ --(library name)-+ !
      !                                                                      !
      +->-- PKLIST -- X --(+------+---------+collection-id.package-id)--------+->
                          +-location-name.-+

      Read member name/package list from PREDICT?  N (Y/N)  DONE

  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help       Exit  Submi Free                                Next  Canc
```

2   Apart from the **specifications** to be made in the **Job Name**, **Job Cards**, and **Profile** fields, to bind a plan, you have to specify the name of the plan and all DBRMs and/or packages that are to be bound into the specified plan.

3   You invoke the window to specify the DBRM members and/or package lists by entering an S in the **MEMBER** and/or **PKLIST** field respectively. Either or both windows must be invoked; otherwise, you are prompted by the system to do so.

   Within the windows for DBRM and package specification, you can scroll using PF6 (--), PF7 (-), PF8 (+), or PF9 (++).

4   If Predict is installed and a plan is documented in Predict, the DBRM members and/or package lists assigned to a plan in Predict can be read by entering Y for this option (default is N). A maximum of 50 DBRM members and/or 20 package lists can be read.

   If you use this option and DBRM members and/or package lists have been successfully read, the **MEMBER** and **PKLIST** selection fields are marked with X, and DONE is displayed next to the **(Y/N)** input field; FAILED is displayed if:

   ■ inconsistencies in the member/package list definition were detected,

   ■ over 50 DBRM members or more than 20 package lists were defined for the specified plan,

   ■ no members or package lists were defined for the specified plan,

   ■ the plan was not documented in Predict at all.

> **Note:** If Predict is not installed, the field **Read member name / package list from Predict?** does not appear on the above screen.

5   Pressing PF11 (Next) takes you to a second **Bind Plan** screen, where you can specify further options of the DB2 BIND command.

A keyword is generated by entering its first letter in the corresponding input field; the default values are highlighted.

```
 16:17:05                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                               - Bind Plan -

   >---+-----------------------+--+--------------+--+--------------------+--->
       !                       ! !              ! !                      !
       +- _____ --( PREPARE )-+  +- FLAG --( _ )-+  +- EXPLAIN --( ___ )-+
      ( NODEFER or DEFER)           ( I, W, E or C)          ( YES or NO )
   >---+-------------------+--+-----------------+--+--------------------+--->
       !                   ! !                 ! !                      !
       +- VALIDATE ( ____ )-+  +- ISOLATION ( __ )-+  +- CACHESIZE ( ____ )+
            ( RUN or BIND )         ( RR, UR or CS )         ( 0 - 4096 )
   >---+----------------------------+---+----------------------------+--->
       !                            ! !                              !
       +--- ACQUIRE --( _____ )-----+   +--- RELEASE --( _____ )---+
              ( USE or ALLOCATE )             ( COMMIT or DEALLOCATE )
   >---+-------------------+----------+---+----------------------------+--->
       !                                  ! !                          !
       +- CURRENTSERVER ( _____ )-+   +-- CURRENTDATA ( ___ )--+
                       location-name                      ( NO or YES )

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit  Submi Free                          Prev  Next  Canc
```

Pressing PF10 (Prev) takes you back to the previous screen.

6   Pressing PF11 (Next) takes you to a third **Bind Plan** screen, where you can again specify further options of the DB2 BIND command.

```
 16:17:18              ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - Bind Plan -
   >--------------+------------------------------------------------------+--->
                  !                                                      !
                  +-- ACTION --+---- _ (REPLACE) --+-------------+----+--+
                               !                   +-- _ RETAIN --+    !
                               +---- _ (ADD) ------------------------+
   >--------------+------------------------------------------------------+--->
                  !                                                      !
                  +-- DYNAMICRULES - _ ( RUN or BIND ) ------------------+

   >-+------------------------------------------------------------------+->< 
     !                                                                  !
    +-+- _ - ENABLE -------- (*) --------+-+---------------------------+-+
      !                                  ! +->- DLIBATCH- _ -(con.-names)-+
      +- _ - ENABLE --+- _ -(con.-types)-+ +->- CICS ---- _ -(applids)----+
      +- _ - DISABLE -+                    +->- IMSBMP -- _ -(imsids)-----+
                                           +->- IMSMPP -- _ -(imsids)-----+
 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help         Exit  Submi Free                         Prev  Next  Canc
```

7   Pressing PF11 (Next) takes you to a fourth **Bind Plan** screen, where you can again specify further
    options of the DB2 BIND command.

```
 16:17:38              ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - Bind Plan -



   >----+--------------------+-------+------------------------+------------->
        !                    !       !                        !
        +-- DEGREE --- ___ ----+     +-- SQLRULES ---  ___ ----+
               ( 1 or ANY )                    ( DB2 or STD )



   >----+--------------------------------------------------------+------------->
        !                                                        !
        +-- DISCONNECT ----+--- _ - ( EXPLICIT ) -----+----------+
                           +--- _ - ( AUTOMATIC ) ----+
                           +--- _ - ( CONDITIONAL) ---+



 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help         Exit  Submi Free                         Prev        Canc
```

8   The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free), or submitted immediately by pressing PF4 (Submi).

Rebind Plan

To generate JCL to rebind a plan, you have to invoke the **Rebind** function. All parameters necessary to rebind a plan are entered in three screens, which show the syntax of the DB2 REBIND PLAN command.

≫ **To generate JCL to rebind a plan**

1   On the **Application Plan Maintenance** menu, enter function code RB.

     In the **Object** field, enter PLAN or PL, and press ENTER.

     The first **Rebind Plan** screen is displayed, where all necessary information must be specified.

```
 19:17:55             ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                            - Rebind Plan -

 Job Name ... FREEJOB_         Job Cards .. X            Profile .. EXBIND1_


  >>- REBIND PLAN ---------------------------------------------------------->

  >-+-(plan names)- X -+-+--------------------+-+----------------------+-->
    !                   ! !                    ! !                      !
    +-- (*) -- _ ------+-+- OWNER ( _____ )-+ +- QUALIFIER ( _____ )-+
                             auth-id                    qualifier-name

  >---+---------------------------------------------------------------+----->
      !                                                               !
      +-- PKLIST ---- _ --(+---------------+collection-id.package-id)--+
      !                    +-location-name.-+                         !
      +-- NOPKLIST -- _ -----------------------------------------------+


 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit  Submi Free                               Next  Canc
```

2   Apart from the **specifications** to be made in the **Job Name**, **Job Cards**, and **Profile** fields, you have to specify the names of the plans to be rebound in a window. If you specify asterisk notation (*), all existing plans are rebound.

3   Pressing PF11 (Next) takes you to a second **Rebind Plan** screen, where you can specify further
     options of the DB2 REBIND command.

     A keyword is generated by entering its first letter in the corresponding input field; the default
     values are highlighted.

```
 16:18:15                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                               - Rebind Plan -


   >---+-----------------------------+--+--------------+--+---------------------+--->
       !                             !  !              !  !                     !
       +- _____ --( PREPARE )-+   +- FLAG --( _ )-+  +- EXPLAIN --( ___ )-+
      ( NODEFER or DEFER)              ( I, W, E or C)         ( YES or NO )
   >---+-------------------+--+----------------+--+--------------------+--->
       !                   !  !                !  !                    !
       +- VALIDATE ( ____ )-+  +- ISOLATION ( __ )-+  +- CACHESIZE ( ____ )+
           ( RUN or BIND )        ( RR, CS or UR )         ( 0 - 4096 )
   >---+-----------------------------+---+-----------------------------+--->
       !                             !   !                             !
       +--- ACQUIRE --( _____ )-----+   +--- RELEASE --( _____ )---+
              ( USE or ALLOCATE )              ( COMMIT or DEALLOCATE )
   >---+-------------------+----------+---+-----------------------------+--->
       !                              !   !                             !
       +- CURRENTSERVER ( _____ )-+   +-- CURRENTDATA ( ___ )--+
                          location-name                    ( NO or YES )
 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help         Exit  Submi Free                          Prev  Next  Canc
```

     Pressing PF10 (Prev) takes you back to the previous screen.

4   Pressing PF11 (Next) takes you to a third **Rebind Plan** screen, where you can again specify
     further options of the DB2 REBIND command.

```
16:18:38               ***** NATURAL TOOLS FOR DB2 *****           2009-10-30
                              - Rebind Plan -


 >-+-----------------+---+-----------------+---+----------------------+->
   !                 ! !                   ! !                        !
   +- DEGREE - ___ ---+  +- SQLRULES - ___ --+  +- DYNAMICRULES - ____ --+
        ( 1 or ANY )         ( DB2 or STD )           ( RUN or BIND )


 >-+-----------------------------------------+------------------------+->
   +- DISCONNECT --+-- _ --( EXPLICIT ) ------+
                   +-- _ --( AUTOMATIC ) -----+
                   +-- _ --( CONDITIONAL ) ---+


 >-+---------------------------------------------------------------+->><
   !                                                               !
   +-+- _ - ENABLE -------- (*) --------+-+-----------------------------+-+
     !                                  ! +->- DLIBATCH- _ -(con.-names)-+
     +- _ - ENABLE --+- _ -(con.-types)-+ +->- CICS ---- _ -(applids)----+
     +- _ - DISABLE -+                     +->- IMSBMP -- _ -(imsids)-----+
                                           +->- IMSMPP -- _ -(imsids)-----+
 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Submi Free                      Prev        Canc
```

5   The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free),
     or submitted immediately by pressing PF4 (Submi).

Free Plan

A free plan can be generated with the **Free** function of the **Application Plan Maintenance** menu.

≫ **To generate JCL to free a plan**

1   On the **Application Plan Maintenance** menu, enter function code FR.

     In the Object field, enter PLAN or PL, and press ENTER.

     The **Free Plan** screen is displayed, where all necessary information must be specified.

```
  16:19:35               ***** NATURAL TOOLS FOR DB2 *****                2009-10-30
                              - Free Plan -

   Job Name ... FREEJOB_          Job Cards .. X            Profile .. EXBIND1_


   >>------- FREE PLAN -----+---(plan name)---- X ------+--------------------->
                            !                           !
                            +--------- (*) ---- _ ------+




    >--------------------+-------------------------+--------------------->
                         !                         !
                         +--- FLAG -----( _ )--------+
                                  (I, W, E or C)



   Command ===>
   Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help         Exit  Submi Free                                       Canc
```

2   Apart from the **specifications** to be made in the **Job Name**, **Job Cards**, and **Profile** fields, all
     parameters necessary to free a plan are entered in a screen showing the syntax of the DB2
     FREE PLAN command. The names of the plans to be freed are entered in a window. If you
     specify asterisk notation (*), all plans are freed.

3   The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free),
     or submitted immediately by pressing PF4 (Submi).

Bind Package

Packages can be bound with the Bind function of the **Application Plan Maintenance** menu. All
parameters necessary to bind a package are entered on three screens, which show the syntax of
the DB2 BIND PACKAGE command.

≫ **To generate JCL to bind a package**

1   On the **Application Plan Maintenance** menu, enter function code "BI".

     In the Object field, enter PACKAGE or PK, and press ENTER.

     The first **Bind Package** screen is displayed, where all necessary information must be specified.

```
 16:19:58                  ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - Bind Package -

 Job Name ... BINDJOB_          Job Cards .. X              Profile .. EXBIND2_


 >>- BIND PACKAGE -(-+---------------------+- _____ ---------->
                     +- _____ . -+    collection-id
                          location-name

 >-------------+------------------+-+---------------------+-------------->
               + OWNER ( _____ )+ + QUALIFIER ( _____ )+
                       auth-id              qualifier-name

 >-+- MEMBER ( _____ )+----------------------------------------------+-+
   !         member-name  +- LIBRARY --- _ (library-name)--------------+ !
    !                                                                     !
   +- COPY ( _____ . _____ )-+------------------------+-+->
               collection-id     package-id +- COPYVER - _ (version-id)-+

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit  Submi Free                               Next  Canc
```

2 Apart from the **specifications** to be made in the **Job Name**, **Job Cards**, and **Profile** fields, to bind a package, you have to specify the collection ID of the package and a DBRM or a further package to be bound into the specified package.

You specify the DBRM or the second package in the **MEMBER** or **COPY** field respectively. Either of the fields must be selected and the package ID will be either the DBRM name or the package ID of the copied package.

3 Pressing PF11 (Next) takes you to a second **Bind Package** screen, where you can specify further options of the DB2 `BIND` command.

A keyword is generated by entering its first letter in the corresponding input field; the default values are highlighted.

```
 16:20:05                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - Bind Package -

  >----------+---------------------+----------+--------------+-------->
            !                     !          !              !
            +- SQLERROR ( _____ )-+        +- FLAG --( _ )-+
              ( NOPACKAGE or CONTINUE )         ( I, W, E or C)
  >----------+-----------+--+------------+-----------------+---------->
            !           ! !            !                 !
            +- EXPLAIN --( ___ )-+        +- VALIDATE ( ____ )-+
                 ( NO or YES )               ( RUN or BIND )
  >----------+-----------------+------+--------------------+---------->
            !                 !      !                    !
            +- ISOLATION ( __ )-+      +- RELEASE -( _____ )-+
              ( RR, RS, CS, UR or NC)     ( COMMIT or DEALLOCATE )
  >----------+-------------------+---+--------------------+---------->
            !                 ! !                      !
            +- CURRENTDATA ( ___ )-+   +- DYNAMICRULES --( ____ )-+
                    ( NO or YES )             ( RUN or BIND )

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Submi Free                        Prev  Next  Canc
```

Pressing PF10 (Prev) takes you back to the previous screen.

4   Pressing PF11 (Next) takes you to a third **Bind Package** screen, where you can again specify further options of the DB2 `BIND` command.

```
 16:20:18                 ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - Bind Package -

 >-+-------------------------------------------------+---+-----------------+->
   !                                                 !   !                 !
   +- ACTION -+- _ (REPLACE) -+--------------+--+--+   +- DEGREE - ___ ----+
             !                 + REPLVER - _ -+  !          ( 1 or ANY )
             !                    (version-id)  !
             +- _ (ADD) ---------------------+


 >-+-----------------------------------------------------------------------+-><
   !                                                                       !
   +-+- _ - ENABLE -------- (*) --------+-+-----------------------------+-+
     !                                  ! +->- DLIBATCH- _ -(con.-names)-+
     +- _ - ENABLE --+- _ -(con.-types)-+ +->- CICS ---- _ -(applids)----+
     +- _ - DISABLE -+                    +->- IMSBMP -- _ -(imsids)-----+
                                          +->- IMSMPP -- _ -(imsids)-----+
                                          +->- REMOTE -- _ -(loc/lu-name)+

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit  Submi Free                          Prev        Canc
```

5   The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free),
     or submitted immediately by pressing PF4 (Submi).

Rebind Package

A package can be rebound with the **Rebind** function of the **Application Plan Maintenance** menu.
All parameters necessary to rebind a package are entered in two screens, which show the syntax
of the DB2 REBIND PACKAGE command

≫ **To generate JCL to rebind a package**

1   On the **Application Plan Maintenance** menu, enter function code RB.

     In the **Object** field, enter PACKAGE or PK, and press ENTER.

     The first Rebind Package screen is displayed, where all necessary information must be specified.

```
 16:20:55                    ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                                 - Rebind Package -

  Job Name ... FREEJOB_          Job Cards .. X              Profile .. EXBIND2_


  >>- REBIND PACKAGE -------------------------------------------------------->


   >-+--- _ ------------------------- (*) --------------------------------+->
     !                                                                    !
      +--- _ -(+--------------+-collection-id.package-id-+--------------+)-+
             +-location-name.-+                          +-.(version-id)-+


   >------------+--------------------+-+----------------------+-----------+>
                !                    ! !                      !
                +- OWNER ( _____ )-+ +- QUALIFIER ( _____ )-+
                       auth-id                  qualifier-name
  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit  Submi Free                                Next  Canc
```

2   Apart from the **specifications** to be made in the **Job Name**, **Job Cards**, and **Profile** fields, you
    have to specify the names of the packages to be rebound in a window. If you specify asterisk
    notation (*), all locally existing packages are rebound.

3   Pressing PF11 (Next) takes you to a second **Rebind Package** screen, where you can specify
    further options of the DB2 REBIND command.

    A keyword is generated by entering its first letter in the corresponding input field; the default
    values are highlighted.

```
 16:21:21                  ***** NATURAL TOOLS FOR DB2 *****            2009-10-30
                                 - Rebind Package -


  >----------+-------------------+----------+-------------------+--------->
            !                   !          !                   !
            +- FLAG -----( _ )---+          +- DEGREE --( ___ )--+
               ( I, W, E or C)                  ( 1 or ANY )
  >----------+-------------------+----------+-------------------+--------->
            !                   !          !                   !
            +- EXPLAIN --( ___ )-+          +- VALIDATE ( ____ )-+
               ( NO or YES )                   ( RUN or BIND )
  >----------+-------------------+-----+----------------------+----------->
            !                   !     !                      !
            +- ISOLATION ( __ )--+     +- RELEASE -( _____ )-+
            ( RR, RS, CS, UR or NC )     ( COMMIT or DEALLOCATE )
  >----------+--------------------+---+-----------------------+----------->
            !                    !   !                       !
            +- CURRENTDATA ( ___ )-+  +- DYNAMICRULES -( ____ )--+
                    ( NO or YES )            ( RUN OR BIND )


 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit  Submi Free                          Prev  Next  Canc
```

Pressing PF10 (Prev) takes you back to the previous screen.

4   Pressing PF11 (Next) takes you to a third **Rebind Package** screen, where you can again specify further options of the DB2 REBIND command.

```
 16:21:38                    ***** NATURAL TOOLS FOR DB2 *****               2009-10-30
                                - Rebind Package -



  >-+---------------------------------------------------------------------------+-><
    !                                                                           !
    +-+- _ - ENABLE -------- (*) --------+-+---------------------------------+-+
      !                                  ! +->- DLIBATCH- _ -(con.-names)-+
      +- _ - ENABLE --+- _ -(con.-types)-+ +->- CICS ---- _ -(applids)----+
      +- _ - DISABLE -+                    +->- IMSBMP -- _ -(imsids)-----+
                                           +->- IMSMPP -- _ -(imsids)-----+
                                           +->- REMOTE -- _ -(loc/lu-name)+




  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit  Submi Free                           Prev        Canc
```

5   The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free),
    or submitted immediately by pressing PF4 (Submi).

Free Package

A package can be freed with the **Free Package** function of the **Application Plan Maintenance**
menu.

≫ **To generate JCL to free a package**

1   On the **Application Plan Maintenance** menu, enter function code FR.

    In the **Object** field, enter PACKAGE or PK, and press ENTER.

    The Free Package screen is displayed, where all necessary information must be specified.

```
 16:22:05                  ***** NATURAL TOOLS FOR DB2 *****                2009-10-30
                              - Free Package -

 Job Name ... FREEJOB_          Job Cards .. X              Profile .. EXBIND2_


  >>-- FREE PACKAGE --------------------------------------------------------------->

   >--+- _ ------------------------ (*) ---------------------------------+-->
      !                                                                  !
      +- _ --(-+--------------+collection-id.+----------- (*) ---------+-)--+
               +location-name.+                +package-id+------------++
                                                          +.---- (*) ---+
                                                          +.(version-id)+


  >----------------------+------------------------+---------------------><
                         !                        !
                         +--- FLAG -----( _ )--------+
                                   ( I, W, E or C )

 Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit  Submi Free                                        Canc
```

2   Apart from the **specifications** to be made in the **Job Name**, **Job Cards**, and **Profile** fields, all
    parameters necessary to free a package are entered in a screen showing the syntax of the DB2
    `FREE PACKAGE` command. The names of the packages to be freed are entered in a window. If
    you specify asterisk notation (*), all local packages are freed.

3   The generated JCL code can be either edited and/or saved in free mode by pressing PF5 (Free),
    or submitted immediately by pressing PF4 (Submi).

List JCL Function

The **List JCL** function serves to invoke the free-mode editor via the **Application Plan Maintenance**
menu.

≫ **To invoke the List JCL function**

1   On the **Application Plan Maintenance** menu, enter function code `LJ`.

    ■ If you leave the **JCL Member** field blank and press ENTER, the empty free-mode editor is
      invoked.

    ■ If you specify a value followed by an asterisk, or specify asterisk notation only and press
      ENTER, a list of JCL members is displayed for selection.

■ If you specify a valid member name and press ENTER, the invoked free-mode editor contains the corresponding JCL.

```
 16:18:18              ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
 APM - free mode     TESTLIB(TESTPLAN)     S 01- --------------Columns 001 072
 =====>                                                   Scroll ===>  PAGE
 ***** **************************** top of data ****************************
 00001 //BINDJOB  JOB TESTPLAN,CLASS=K,MSGCLASS=X
 00002 //********************************************************
 00003 //* EXAMPLE JOB PROFILE FOR BIND, FREE AND REBIND          *
 00004 //*                                                        *
 00005 //* BIND PLAN                                              *
 00006 //********************************************************
 00007 //BINDJOB  EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=4096K
 00008 //STEPLIB  DD DSN=DB2.Vnnn.DSNLOAD,DISP=SHR
 00009 //DBRMLIB  DD DSN=DB2.Vnnn.DBRMLIB.DATA,DISP=SHR
 00010 //SYSTSPRT DD SYSOUT=*
 00011 //SYSPRINT DD SYSOUT=*
 00012 //SYSUDUMP DD SYSOUT=*
 00013 //SYSTSIN  DD *
 00014 DSN SYSTEM (DB2)
 00015    BIND PLAN (PLAN1)
 00016    MEMBER (  DBRM1)
 00017 END

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help         Exit  Submi Rfind Rchan -     +          <     >    Canc
```

Within the free-mode editor, JCL members can be copied, listed, purged, retrieved from, or saved in a Natural library. All this is done via maintenance commands; see *Global Mainten-enance Commands*.

2   Press PF4 (Submi) to submit JCL code listed in the editor, press PF5 (Fix) to switch to fixed mode.

Display Job Output

The **Display Job Output** function can be used to display the output of a JCL member.

> **Note:** The Display Job Output function is available only if the Entire System Server is installed.

≫ **To display the output of a JCL member**

1   On the **Application Plan Maintenance** menu, enter function code J0.

In the Node field, the default node number (148) for Entire System Server can be modified.

A screen is displayed, where you can specify the desired job name and job number, as well as the numbers of the SYSOUT types.

```
 16:20:05                 ***** NATURAL TOOLS FOR DB2 *****            2009-10-30
                           - Application Plan Maintenance -




              Job Name ........ _____
              Job Number ...... _____
              Sysout Type ..... __           ( CC,JL,SI,SM,SO    )
              Sysout Number ... __           ( Sysout file number )
              Node ........... 148






 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
       Help      Exit                                        Logn       Canc
```

2   In the **Job Name** field, a valid job name can be specified.

   ■ If you specify a value followed by an asterisk (*), or specify asterisk (*) notation only, a list
     of job output members is displayed for selection. In a job output member selection list, you
     can mark an output member with either B to display the member only, or L to display a list
     of all the job output's SYSOUT data sets, which in turn can be marked with B for display.

   ■ If you leave the **Job Name** field blank, you must specify a job number.

3   In the **Job Number** field, you can specify a unique job number. Only if a unique job number
    has been specified, specifications can be made in the **Sysout Type** and **Sysout Number** fields,
    too.

4   In the **Sysout Type** field, you can specify the type of SYSOUT data set of the job with the specified
    job number to be displayed. The following codes apply:

Code	SYSOUT Type
CC	Condition Code
JL	Job Listing
SI	System Input
SM	System Message
SO	System Output

5   In the **Sysout Number** field, you can specify a file number to display a specific SYSOUT data set of the type specified in the Sysout type field.

If you leave the **Sysout Number** field blank, all SYSOUT data sets of the specified type are displayed.

6   Catalog Maintenance

The **Catalog Maintenance** part of the **Natural Tools for DB2** enables you to generate SQL statements to maintain the DB2 catalog (that is, DB2 tables and other DB2 objects) without leaving your development environment.

The **Catalog Maintenance** function incorporates an SQL generator that automatically generates from your input the SQLCODE required to maintain the desired DB2 object. You can display, modify, save, and retrieve the generated SQLCODE.

The DDL/TML definitions are stored in the current Natural library.

Fixed Mode and Free Mode

The catalog maintenance function offers two modes of operation: fixed mode and free mode.

≫ **To switch from fixed mode to free mode**

■   Press PF5 (Free).

≫ **To return from free mode to fixed mode**

■   Press PF3 (Exit) in free mode.

Fixed Mode

In fixed mode, input screens with syntax graphs help you to specify correct SQLCODE. You simply enter the required data in the input screens, and the data are automatically checked to ensure that they comply with the DB2 SQL syntax. If the input is incomplete, you are prompted for the missing data. Then, SQL members are generated from the entered data. The members can be executed directly by pressing PF4 (Submi). But you can also press PF5 (Free) to switch to free mode, where the generated SQLCODE can be modified.

After the execution of an SQL statement, a message is returned, which indicates that the statement has been successfully executed. If an error occurred, the resulting DB2 error message can be displayed by pressing PF2 (Error), which executes the `SQLERR` command.

Input screens consist of various kinds of input fields. There are:

▪ fields to enter DB2 object names,

▪ fields to invoke windows,

▪ fields to be marked for selection,

▪ fields to enter keywords,

▪ fields to specify numeric values,

■ fields to enter string constants.

For each field where a window can be invoked, you can specify an S. When you press ENTER, the window appears and you can select or enter the necessary information. If such a selection is required, an S is already preset when the corresponding screen is invoked.

When you press ENTER again, the window closes and if data have been entered, the field is marked with X instead of S. If not, the field is left blank or marked with S again.

This will continue each time you press ENTER until no S remains. To redisplay a window where data have been entered, you change its X mark back to S.

If another letter or character is used, an error message appears on the screen.

Mark field with S to show window.

The wrong character is automatically replaced by an S and if you press ENTER again, the corresponding window appears.

In fields where keywords are to be entered, you must enter one of the keywords displayed beneath the field. Default keywords are highlighted.

Free Mode

When free mode is invoked from fixed mode (by pressing PF5 (Free)), the data that were entered in fixed mode are shown as generated SQLCODE which can be saved for later use or modification.

If you modify an SQL member in free mode, this has no effect on the fixed-mode version of the member. You can save your modified code in free mode, but when you return to fixed mode, the original data appear again. Thus, both original and modified data are available.

In free mode you can execute the member currently in the source area by pressing PF4 (Submi), as in fixed mode.

Execution of SQL statements automatically switches to the output screen, which shows the SQL return code of the executed commands.

See the list of the SQLCODE maintenance commands available in free mode in the section *Global Maintenenance Commands*.

Invoking the Catalog Maintenance Function

 ≫ **To invoke the Catalog Maintenance function**

■ On the **Natural Tools for DB2 Main Menu**, enter function code C, and press ENTER.

The **Catalog Maintenance** menu is displayed:

```
 16:03:13                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                            - Catalog Maintenance -

        Code    Maintenance  Parameter      Code    Authorization  Parameter

         CR     CREATE       Object          GR     GRANT          Object
         AL     ALTER        Object          RE     REVOKE         Object
         DR     DROP                         LO     LOCK TABLE
         SC     SET SQLID


        Code    Description  Parameter      Code    Function       Parameter

         EN     EXPLAIN                       F     Free Mode       Member
         CO     COMMENT ON                    ?     Help
         LB     LABEL ON                      .     Exit


     Code .. __    Object .... _____
                   Library ... _____
                   Member .... _____


 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit                                                     Canc
```

In the **Code** field, the function code assigned to the desired function can be specified, together with the desired **Object**, **Library**, and/or **Member** name.

If you switch to **free mode** and enter a valid member name, you can read this member from the Natural library specified with the **Library** parameter. The **Library** parameter is preset with your Natural user ID.

With the CREATE VIEW and EXPLAIN functions, a subselect or an explainable SQL statement must be entered, respectively. Both can be done in a separate editor session, where previously saved members can be used. The editor is invoked by entering an S in the appropriate field.

With the functions CREATE, ALTER, GRANT, and REVOKE, an object code must be specified, for example, TB for TABLE. If you leave the object field blank, a window is displayed which shows you a list of all available objects together with their object codes.

If you enter for example the CREATE function without specifying an object, a window is invoked which prompts you for the type of object to be created:

```
 16:03:13              ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                            - Catalog Maintenance -

       Code   +--------------------+      Code    Authorization  Parameter
              ! CREATE             !
        CR    !                    !       GR      GRANT          Object
        AL    ! AL    ALIAS        !       RE      REVOKE         Object
        DR    ! DB    DATABASE     !       LO      LOCK TABLE
        SC    ! IX    INDEX        !
              ! ST    STOGROUP     !
       Code   ! SY    SYNONYM      !      Code     Function       Parameter
              ! TB    TABLE        !
        EN    ! TS    TABLESPACE   !       F       Free Mode      Member
        CO    ! VI    VIEW         !       ?       Help
        LB    ! .     Exit         !       .       Exit
              !                    !
     Code ..  ! __ .. Enter Object !
              !                    !
              +--------------------+


 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                                    Canc
```

In the following section some examples illustrate how to use the **Catalog Maintenance** function in fixed mode.

Create Table Function

≫ **To invoke the Create Table function**

1   In the CREATE function, enter the object code TB, and press ENTER.

The first **Create Table** syntax input screen is displayed.

You can enter the creator and table names on this screen, as well as the individual column names, formats, and lengths, as shown below:

```
   09:47:19                  ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - Create Table -                              1  / 9

 >>- CREATE TABLE - SAG_____ . DEMOTABLE_____ ---------------------------->
                    <creator.>table-name
 >+--- LIKE ------- _____ . _____ +---------------------+-+>
  !                 <creator.>table/view-name   +- _ - INCLUDING IDENTITY + +
  !                                                                        ↵
 !
  +( COL1_____     CHAR_____ ( 20_____ ) _ - __ - _ - __ - _ , +
  +- COL2_____     INTEGER_____ ( _____ ) _ - NN - _ - 2_ - _ , +
  +- COL3_____     SMALLINT_____ ( _____ ) _ - NN - _ - 1_ - _ , +
  +- COL4_____     CHAR_____ ( 2_____ ) S - __ - _ - __ - _ , +
  +- COL5_____     VARCHAR_____ ( 30_____ ) _ - NN - _ - 3_ - _ , +
  +- COL6_____     DECIMAL_____ ( 2,5_____ ) _ - __ - X - __ - _ , +
  +- COL7_____     FLOAT_____ ( _____ ) _ - NN - _ - __ - _ , +
  +- COL8_____     DATE_____ ( _____ ) _ - __ - _ - __ - _ , +
  +- COL9_____     TIME_____ ( _____ ) _ - __ - _ - __ - _ , +
  +- _____     _____ ( _____ ) _ - __ - _ - __ - _ , +
        column-name          format          length    S/M  NN  fld  PK/ R/C
                                                         B       proc UK  D/G
 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec  Free  --    -      +    ++          Next  Canc
```

> **Note:** Since the specification of any special characters as part of a Natural field or DDM name does not comply with Natural naming conventions, any special characters allowed within DB2 should be avoided. The same applies to DB2 delimited identifiers, which are not supported by Natural.

In the top right-hand corner of the screen, the index of the top most column (1), and the total number of columns specified (9) is displayed. If you want to specify more columns than fit on one terminal screen, press PF8 (+) to scroll one page forward.

An S in the **S/M/B** field of column 4 means that the FOR SBCS DATA option is selected for this column. Other possible values for this field are M (FOR MIXED DATA) and B (FOR BIT DATA).

Columns 3, 2, and 5 form the primary key, in the specified order. Primary key columns must be selected with an S or ordered by specifying appropriate numbers between 1 and 16. In the present example, all primary key columns are defined as NOT NULL. In addition, column 7 is specified as NOT NULL.

For column 6, a field procedure has been entered in a window invoked by S. The window has been closed again, and the **fld proc** field is now marked with X.

2   If you enter an R in the **R/C/D/G** field for a given column and press ENTER, a window is displayed, in which you can specify a references clause, which identifies this column as a foreign key of a referential constraint.

```
+-------------------------------------------------------+
!  References-Clause for Column:  COL1                  !
!                                                       !
! >--- REFERENCES ---- _____ . _____ --> !
!                       <creator.>table-name            !
! >-+----------------------------------+--------------->  !
!   +- ON DELETE --+-- _ - RESTRICT --+                 !
!                  +-- _ - CASCADE ---+                 !
!                  +-- _ - SET NULL --+                 !
!                  +-- _ - NO ACTION -+                 !
!                                                       !
+-------------------------------------------------------+
```

You must specify the name (with an optional creator name) of the parent table to be referenced. In addition, you must specify the action to be taken when a row in the referenced table is deleted. The following options are provided:

- RESTRICT or NO ACTION prevents the deletion of the parent row until all dependent rows are deleted.

- CASCADE deletes all dependent rows, too.

- SET NULL sets to null all columns of the foreign key in each dependent row that can contain null values.

- A key that consists of more than one column must be defined by a FOREIGN KEY clause.

3   If you enter a C in the **R/C/D/G** field for a given column and press ENTER, a window is displayed, in which you can specify a check constraint for this column.

```
 16:08:09                  ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - Create Table -                             1  / 9

 >>--- CREATE TABLE --------- SAG_____ . DEMOTABLE_____ --------------->
                                   <creator.>table-name
 >+------- LIKE ------------- _____ . _____ ---------------+->
  !                                <creator.>table/view-name              !
  +( COL1_____ - CHAR_____ ( 20___ ) - _ - _ -- _ - __ - C ,-+
+--------------------------------------------------------------------------+
! --- check-constraint  for   Column:    COL1                   ---------- ↵
!
 !                                                                         ↵
 !
 ! >-+----------------------+- CHECK  ( _____ !
 !   !                      !         _____ !
 !  +- CONSTRAINT - _____ -+        _____ !
 !            constraint-name          _____ !
 !                                     _____ !
 !                                     _____ !
 !                                                                    ↵
 !
 !                                                                    ↵
 !
 +--------------------------------------------------------------------------+

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
                  Exit                                                   Canc
```

You must specify a column check condition. A check condition is a search condition with various restrictions which are described in detail in the relevant DB2 literature by IBM. In addition, you may specify a name for the check constraint.

4   If you enter a D in the **R/C/D/G** field for a given column and press ENTER, a window is displayed, in which you can specify a default value other than the system default value for this column.

```
 10:14:04                  ***** NATURAL TOOLS FOR DB2 *****                2009-10-30
                               - Create Table -                             1  / 9

+----------------------------------------------------------------------------+
!      Default-Clause for Column:  COL1                                      !
!                                                                            !
! >--- _ - WITH DEFAULT ----------------------------------------------------> !
! >-+-----------------------+-+--------------------------------+-+---+-> !
!   +- _____ ( -+ +-- _ - USER --------------------+ + ) +    !
!      cast-function-name      +-- _ - CURRENT SQLID -------------+      !
!                              +-- _ - NULL --------------------+        !
!                              _____          !
!                              _____          !
!                              _____          !
!                              _____          !
!                              _____          !
!                                        constant                        !
!                                                                            !
+----------------------------------------------------------------------------+
                                                   B        proc UK   D/G
  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
                  Exit                                                     Canc
```

One of the following types of default values can be specified:

- USER: an execution-time value of the special register USER.

- CURRENT SQLID: the SQL authorization ID.

- NULL: the null value.

- constant: a constant which names the default value for the column.

For further information on default values, refer to the relevant DB2 literature by IBM.

5   If you enter a G in the **R/C/D/G** field for a given column and press ENTER, a window is displayed, in which you can define the **GENERATED-Clause** for this column.

```
 10:18:29                 ***** NATURAL TOOLS FOR DB2 *****                2009-10-30
                            - Create Table -                                 1  / 9

>>- CREATE TABLE - SAG_____ . DEMOTABLE_____ ----------------------------->
+-----------------------------------------------------------------------------+
!        GENERATED-Clause for Column:  COL1                                   !
!                                                                             !
! >------ GENERATED ---------+-- _  ALWAYS -------+----------------------->   !
!                            +-- _  BY DEFAULT ---+                           !
! >-+----------------------------------------------------------------------+-> !
!   +- _  AS IDENTITY -+------------------------------------------------+-+    !
!                      +- ( -+-- _  START WITH --- 1_____ --+- ) -+       !
!                            +-- _  INCREMENT BY - 1_____ --+              !
!                            ++- _  NO CACHE ------------------++              !
!                             +- _  CACHE -------- 20_____ -+               !
!                                                                             !
+-----------------------------------------------------------------------------+
 +- _____ _____ ( _____ ) _ - __ - _ - __ - _ , +
       column-name         format         length    S/M  NN  fld  PK/ R/C
                                                     B        proc UK   D/G
   Command ===>
   Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
                 Exit                                                       Canc
```

GENERATED can only be defined if the column has a ROWID data type (or a distinct type that is based on a ROWID data type), or if the column is to be an identity column.

For further information on the *GENERATED-Clause*, refer to the relevant DB2 literature by IBM.

6   Windows like the one below may help you in making a valid selection. They are invoked by entering the help character (?) in the appropriate field on the screen:

```
 10:23:44                   +--------------------------------+         2009-10-30
                            ! I       INTEGER               !             1  / 9
                            ! S       SMALLINT              !
>>- CREATE TABLE - SAG_ ! F       FLOAT(integer)         ! ---------------->
                   <cr !  RE      REAL                  !
>+--- LIKE ------- ____ !  DO      DOUBLE                ! -------------+-+>
  !                 <cr !  DE      DECIMAL(integer,integer) ! DING IDENTITY + +
  !                     !  N       NUMERIC(integer,integer) !                 !
 +( COL1_____ !  CH      CHAR(integer)         !  - _ - __ - _ , +
 +- COL2_____ !  VARC    VARCHAR(integer)      !  - _ - 2_ - _ , +
 +- COL3_____ !  CL      CLOB(integer)         !  - _ - 1_ - _ , +
 +- COL4_____ !  B       BLOB(integer)         !  - _ - __ - _ , +
 +- COL5_____ !  G       GRAPHIC(integer)      !  - _ - 3_ - _ , +
 +- COL6_____ !  VARG    VARGRAPHIC(integer)   !  - _ - __ - _ , +
 +- COL7_____ !  DB      DBCLOB(integer)       !  - _ - __ - _ , +
 +- COL8_____ !  DA      DATE                  !  - _ - __ - _ , +
 +- COL9_____ !  TIME    TIME                  !  - _ - __ - _ , +
 +- _____ !  TIMES   TIMESTAMP             !  - _ - __ - _ , +
      column-name       !                               !   fld  PK/ R/C
                        !  RO      ROWID                 !   proc UK  D/G
                        !                               !
Command ===>            !  _____ .. Enter Value        !
Enter-PF1---PF2---PF3-- +--------------------------------+ F10--PF11--PF12---
      Help  Error Exit  Exec  Free  --    -     +    ++         Next  Canc
```

In the case of complex SQL statements, more than one input screen may be required. If so, you can switch to the following screen by pressing PF11 (Next), or return to the previous screen by pressing PF10 (Prev).

As you can see on the above screen, the beginning of the syntax specification for an SQL statement is always indicated by >>.

7 Since the syntax of the CREATE TABLE statement is a rather complex one, three more screens are required. Once all necessary information has been entered on the first screen, you press PF11 (Next) to display the next **Create Table** input screen, where you can specify additional optional parameters.

```
 10:31:51                 ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                            - Create Table -                               1  / 0

 +--------------------------------<--------------------------------+
>-+-+------------------------------------------------------------+-+->
   !                                                                 !
   +- , - FOREIGN KEY ----- _____ ------- _ --- (column-name) ->  !
                     <constraint-name>                               !
                                                                     !
    >---- REFERENCES ------ _____ . _____ -------->  !
                            <creator.>table-name                     !
                                                                     !
    >-+-------------------------+----- ON DELETE -+- S - RESTRICT -+-+
      +-- _ --- (column-name) ----+               +- _ - CASCADE --+
                                                  +- _ - SET NULL -+
                                                  +- _ - NO ACTION +




 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec  Free  --    -     +     ++    Prev  Next  Canc
```

On this screen, you can specify a referential constraint to another table. To do so, enter an S in the **column-name** field. A list of all columns available in the current table (dependent table) is displayed, where you can select the column(s) to comprise the foreign key related to another table (parent table). You can also specify a name for the constraint. If not, the constraint name is derived from the first column of the foreign key.

A foreign key consists of one or more columns in a dependent table that together must take on a value that exists in the primary key of the related parent table.

In the **REFERENCES** part, you must specify the table name (with an optional creator name) of the parent table which is to be affected by the specified constraint. In addition, you must specify the action to be taken when a row in the referenced parent table is deleted.

The following options are provided:

- RESTRICT or NO ACTION prevents the deletion of the parent row until all dependent rows are deleted.

- CASCADE causes all dependent rows to be deleted, too.

- SET NULL sets to null all columns of the foreign key in each dependent row that can contain null values.

In the top right-hand corner of the screen, the index of the currently displayed referential constraint block (1) and the total number of referential constraint blocks defined (0) is displayed.

When all information has been entered, you can press either PF10 (Prev) to return to the previous screen, or PF11 (Next) to go to the next screen.

8   On the next screen you have again the possibility to specify columns as unique. This time, however, up to six groups of unique columns can be defined, with up to 16 columns per group. The individual columns are specified in a window, which can be invoked for each group.

```
 10:43:52                   ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                                  - Create Table -


>---+------------------------------------- +-----------------------+ -->
   !                                        ! ------ column-name ---- !
   +- , - UNIQUE --------------------------- !  __   COL1             !
   !                                        !  __   COL2             !
   +- , - UNIQUE --------------------------- !  __   COL3             !
   !                                        !  __   COL4             !
   +- , - UNIQUE --------------------------- !  __   COL5             !
   !                                        !                        !
   +- , - UNIQUE --------------------------- !  __   COL6             !
   !                                        !  __   COL7             !
   +- , - UNIQUE --------------------------- !  __   COL8             !
   !                                        +-----------------------+
   +- , - UNIQUE --------------------------   _ -- (column-name) ---+




Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit              --    -    +    ++              Canc
```

Since unique columns must not contain null values, a further window is invoked automatically, on which you can define the columns specified as unique also as NOT NULL (unless you already defined them as such on the first Create Table input screen).

When all information has been entered, you can press either PF10 (Prev) to return to the previous screen or PF11 (Next) to go to the last syntax input screen.

9   On the last syntax input screen, you can now:

- Restrict dropping of the current table (and also of the database and tablespace that contain this table).

- Define a check constraint for the current table. To define a check constraint, you must specify a table check condition. A check condition is a search condition with various restrictions which are described in the relevant DB2 literature by IBM. In addition, you may specify a name for the check constraint.

```
 10:47:02                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                             - Create Table -

  >-----+- IN ----------- _____ . _____ --------------------+------>
        !                 <database-name.>tablespace-name          !
        +- IN DATABASE -------------- _____ --------------------+
                                 database-name

  >------- EDITPROC ------ _____ --------- VALIDPROC -- _____ -------->

  >------- AUDIT --------- _____ ---------- OBJID ------ _____ ----------->
                ( NONE, CHANGES, ALL )                   integer

  >------- DATA CAPTURE -- _____ ---------- CCSID ------ _____ --------->
                  ( NONE, CHANGES )                   ( ASCII, EBCDIC )

  >------- WITH RESTRICT ON DROP -- _ ------------------------------------->

  >------- CHECK --------- _ --------------------------------------------><
                check-condition

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
      Help  Error Exit  Exec  Free                        Prev        Canc
```

If you press PF10 (Prev) on this screen, you return to the previous screen.

As you can see on the above screen, the end of the syntax specification for an SQL statement is always indicated by ><.

An active help facility that consists of selection lists in windows is available for all fields referencing existing database objects. Selection lists are invoked by entering either an asterisk (*) or part of an object name followed by an asterisk in the corresponding input field.

If, for example, you enter D* in the "database-name" field of the above screen, a window appears where you can check your selection criteria. When you press ENTER, a list of all databases whose names begin with D appears.

```
 10:47:02              ***** NATURAL TO +----------------------+   2009-10-30
                          - Create ! Database   Tablespa  !
                                   ! D*_____ . _____   !
  >-----+- IN ----------- d*_____ . !                    ! ----+------>
        !              <database-name.> +----------------------+    !
        +- IN DATABASE -------------- ! Select ==> __        ! ----+
                               dat !                         !
                                   !   1 DSNDB04  ALLDATA0 !
  >------- EDITPROC ------ _____ -- !  2 DSNDB04  CANTABRD ! __ -------->
                                   !   3 DSNDB04  CDBPRO6  !
  >------- AUDIT --------- _____ --- !  4 DSNDB04  DATEGRP ! ----------->
                  ( NONE, CHANGES, AL !  5 DSNDB04  DECIMALR !
                                   !   6 DSNDB04  DEMO     !
  >------- DATA CAPTURE -- _____ --- !  7 DSNRGFDB DSNRGFTS ! _ --------->
                  ( NONE, CHANGES  !   8 DSNRLST  DSNRLS01 ! CDIC )
                                   !   9 DB27WRK  DSN32K01 !
  >------- WITH RESTRICT ON DROP -- _ !                     ! ----------->
                                   !  10 DB27WRK  DSN4K01  !
  >------- CHECK --------- _ --------- !  11 DSN8D71L DSN8S71B ! ---------><
                  check-condition  !  12 DSN8D71P DSN8S71C !
                                   +----------------------+
Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
     Help  Error Exit  Exec  Free                         Prev        Canc
```

Within the selection list, you can scroll up (PF6 / "--" or PF7 / "-") or down (PF8 / "+" or PF9 /
"++"), and select the desired database. The name of the selected database is copied to the
corresponding field in your input screen.

10 When all information has been entered, you can either switch to free mode (PF5) or submit
    the created member directly to DB2 for execution (PF4). If execution is successful, you receive
    the message:

```
Statement(s) successful, SQLCODE = 0
```

If not, an error code is returned.

In free mode, the following editor screen displays the generated SQLCODE:

```
 10:53:50                 ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
 FREE - Input         SAG                    S 01- --------------Columns 001 072
 =====>                                                    Scroll ===>  PAGE
 ***** ***************************** top of data *****************************
 00001 CREATE TABLE SAG.DEMOTABLE
 00002  (COL1                  CHAR(20),
 00003   COL2                  INTEGER              NOT NULL,
 00004   COL3                  SMALLINT             NOT NULL,
 00005   COL4                  CHAR(2)              FOR SBCS DATA,
 00006   COL5                  VARCHAR(30)          NOT NULL,
 00007   COL6                  DECIMAL(2,5)
 00008    FIELDPROC PROGNAME
 00009      ('STRING1','STRING2'),
 00010   COL7                  FLOAT                NOT NULL,
 00011   COL8                  DATE,
 00012   COL9                  TIME,
 00013   PRIMARY KEY (COL3,             COL2,
 00014            COL5)
 00015  )
 00016  IN DSNDB04.DEMO;
 ***** *************************** bottom of data ****************************

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Setup Exit  Exec  Rfind Rchan -    +    Outpu             Canc
```

The free-mode editor is an adapted version of the Software AG Editor. It is almost identical to the interactive **ISQL - Input** screen. However, no SELECT statements can be issued from free mode.

For further details, please refer to the relevant *Software AG Editor* documentation.

Create Tablespace Function

≫ **To invoke theCreate Tablespace function**

1   On the **Catalog Maintenance** screen, enter the code CR.

In the **Object** field, enter TS and press ENTER.

The first **Create Tablespace** syntax input screen is displayed:

```
 16:08:09                   ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                                - Create Tablespace -

 >>-- CREATE TABLESPACE ----- TS1_____ -------- IN ----- _____ ----------->
                              tablespace-name            database-name


         +- VCAT ---- _____ ------------------------------------------+
 >- USING -+        catalog-name                                        +->
         +- STOGROUP- _____ - PRIQTY ____ - SECQTY ___ - ERASE ___ -+
                     stogroup-name     integer       integer   ( YES or NO )


 >--- FREEPAGE -------- ___ ----- PCTFREE -- __ ------------ COMPRESS ___ --->
                        integer            integer                 ( YES or NO )

 >--- NUMPARTS -------- __ ----- _ ------------------------------------------>
                        integer    PART

 >--- SEGSIZE --------- __ -------------------------------------------------->
                        integer

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
      Help  Error Exit  Exec  Free                              Next  Canc
```

2   Once you have entered all necessary information, press PF11 (Next) to go to the next screen:

```
   16:08:09                ***** NATURAL TOOLS FOR DB2 *****            2009-10-30
                              - Create Tablespace -


  >---+-----------------------------------------------------------------+-->< 
      !                                                                 !
      +--- BUFFERPOOL ------ _____ ------------------------------------+
      !                      bufferpool-name                           !
      +--- LOCKSIZE -+------ _____ --------------------------+-------+
      !             !( ANY, TABLE, TABLESPACE )               !       !
      !             +------ ____ ---+--------------------+--+          !
      !              ( ROW or PAGE )!                    !             !
      !                             +- LOCKMAX -- _____ --+           !
      !                                   ( SYSTEM or integer )        !
      +--- CLOSE ----------- ___ ---------------------------------------+
      !                ( YES or NO )                                    !
      +--- DSETPASS -------- _____ ----------------------------------+
                            password



   Command ===>
   Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
        Help  Error Exit  Exec  Free                         Prev        Canc
```

On the second **Create Tablespace** syntax input screen, you can now specify additional buffer pool names as well as the LOCKSIZE option with the LOCKMAX clause.

If you enter an S in the **bufferpool-name** field and press ENTER, a window is displayed, in which you can specify additional buffer pool names.

Refer to the relevant DB2 literature by IBM for further details on the COMPRESS, LOCKSIZE and LOCKMAX clauses.

Alter Table Function

The following example illustrates the use of the **Alter Table** syntax input screen.

≫ **To invoke the Alter Table function**

1   On the **Catalog Maintenance** screen, enter the code AL.

     In the **Object** field, enter TB and press ENTER.

     The **Alter Table** screen is displayed, where you can specify the following:

```
11:01:47                  ***** NATURAL TOOLS FOR DB2 *****            2009-10-30
                               - Alter Table -

>>-- ALTER TABLE ------------- _____ . _____ ---------------->
                                     <creator.>table-name
>-+- ALTER _____ -- SET DATA TYPE - VARCHAR - ( _____ ) --+>
  !            column-name                                      length     !
>-+- ADD _____ _____  ( _____ ) - _ -- __ - __ -->
  !         column-name        format           length    S/M/B  NN  UK/PK
  !    +--<
  !    +>- _ ------ _ ---------- _ -------------- _ --------------- _ -------+>
  !    field-proc default   check-constr   reference-constr   GENERATED-Clause!
  !                                                                          !
>-+- VALIDPROC --------------- _____ ------------------------------------+>
  !                            program-name or NULL                         !
  +- AUDIT ------------------- _____ --------------------------------------+
  !                            ( NONE, CHANGES, ALL )                        !
  +- DATA CAPTURE ------------ _____ --------------------------------------+
                               ( NONE, CHANGES )



  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec  Free                                Next  Canc
```

2   If you enter an S in the **field-proc** input field and press ENTER, a window is displayed, in which you can specify a field procedure to be executed for this column:

```
 11:05:47                  ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                                  - Alter Table -

>>-- ALTER TABLE ------------- _____ . _____ ---------------->
                                    <creator.>table-name
>-+- ALTER _____ -- SET DATA TYPE - VARCHAR - ( _____ ) --+>
  !             column-name                                    length       !
>-+- ADD _____ _____  ( _____ ) - _ -- __ - __ -->
  !          column-name         format        length    S/M/B  NN  UK/PK
  !     +--<              +----------------------+
  !     +>- S ------ _ -- !          1 / 0       ! --------------- _ -------+>
  !   field-proc default ! --- FIELDPROC ----    ! -constr   GENERATED-Clause!
  !                      !     _____          !                          !
>-+- VALIDPROC --------- !     program-name      ! -------------------------+>
  !                      !  ( _____ ,     !                          !
  +- AUDIT ------------- !    _____ ,     ! -------------------------+
  !                      !    _____ )     !                          !
  +- DATA CAPTURE ------ !    (constants,)       ! -------------------------+
                         !                       !
                         !                       !
                         +----------------------+
Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit                                                      Canc
```

3   If you enter an S in the **default** field and press ENTER, a window is displayed, in which you can specify a default value other than the system default value for this column:

```
 11:07:31                    ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                                   - Alter Table -


+----------------------------------------------------------------------------------+
! >--- _ - WITH DEFAULT ------------------------------------------------------->  !
! >-+-----------------------+-+------------------------------------+-+---+->< !
!   +- _____  ( -+ +-- _ - USER --------------------+ + ) +    !
!      cast-function-name      +-- _ - CURRENT SQLID -------------+         !
!                              +-- _ - NULL --------------------+           !
!                              _____            !
!                              _____            !
!                              _____            !
!                              _____            !
!                              _____            !
!                                          constant                        !
!                                                                          !
!                                                                          !
!                                                                          !
+----------------------------------------------------------------------------------+


Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                    Exit                                                    Canc
```

One of the following types of default values can be specified:

- USER: an execution-time value of the special register USER.

- CURRENT SQLID: the SQL authorization ID.

- NULL: the null value.

- constant: a constant which names the default value for the column.

For further information on default values, refer to the relevant DB2 literature by IBM.

4   If you enter an S in the **check-constraint** field and press ENTER, a window is displayed, in which you can specify a check constraint for this column:

```
 11:09:02                 ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - Alter Table -

>>-- ALTER TABLE ------------- _____ . _____ ---------------->
                               <creator.>table-name
>-+- ALTER _____ -- SET DATA TYPE - VARCHAR - ( _____ ) --+>
  !            column-name                                      length      !
>-+- ADD _____ _____ ( _____ ) - _ -- __ - __ -->
  !          column-name        format        length    S/M/B  NN  UK/PK
  !    +--<
  !    +>- _ ------ _ ---------- S -------------- _ --------------- _ -------+>
  !    field-proc default   check-constr  reference-constr  GENERATED-Clause!
+-------------------------------------------------------------------------+
! >-+----------------------+- CHECK  ( _____ !
! !                        !           _____ !
!  +- CONSTRAINT - _____ -+         _____ !
! !           constraint-name          _____ !
! !                                                                         !
! !                                    _____ !
! !                                    _____ !
! !                                                                         !
+-------------------------------------------------------------------------+
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                  Exit                                                     Canc
```

You must specify a column check condition. A check condition is a search condition with
various restrictions which are described in detail in the relevant DB2 literature by IBM. In
addition, you may specify a name for the check constraint.

5   If you enter an S in the **reference-constraint** field and press ENTER, a window is displayed, in
    which you can specify a references clause, which identifies this column as a foreign key of a
    referential constraint:

```
 11:10:36                  ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - Alter Table -

>>-- ALTER TABLE ------------- _____ . _____ ---------------->
                                    <creator.>table-name
>-+- ALTER _____ -- SET DATA TYPE - VARCHAR - ( _____ ) --+>
  !            column-name                                    length      !
>-+- ADD _____ _____ ( _____ ) - _ -- __ - __ -->
  !          column-name       format         length   S/M/B  NN  UK/PK
  !     +--<
  !     +>- _ ------ _ ---------- _ -------------- S --------------- _ -------+>
  !    field-proc default  check-constr   reference-constr   GENERATED-Clause!
  !         +-----------------------------------------------------+       !
>-+- VALID ! >--- REFERENCES ---- _____ . _____ -->  ! ------+>
  !       !                            <creator.>table-name          !      !
  +- AUDIT ! >-+------------------------------------+--------------->  ! ------+
  !       !    +- ON DELETE --+-- _ - RESTRICT --+                     !      !
  +- DATA  !                  +-- _ - CASCADE ---+                     ! ------+
          !                   +-- _ - SET NULL --+                     !
          !                   +-- _ - NO ACTION -+                     !
          !                                                            !
Command == !                                                          !
Enter-PF1- !                                                     ! -PF12---
          +--------------------------------------------------------+  Canc
```

You must specify the name (with an optional creator name) of the parent table to be referenced. In addition, you must specify the action to be taken when a row in the referenced table is deleted. The following options are provided:

- RESTRICT or NO ACTION prevents the deletion of the parent row until all dependent rows are deleted.

- CASCADE deletes all dependent rows, too.

- SET NULL sets to null all columns of the foreign key in each dependent row that can contain null values.

6   Once you have entered your column definitions, press PF11 (Next).

A screen is invoked in which you can add or drop primary and/or foreign keys:

```
 11:14:42                 ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - Alter Table -

>--+--- ADD ------ PRIMARY KEY ------------------ _ -- (column-name) ---+
   !                                                                    !
   +--- DROP ----- PRIMARY KEY ------------------ _ --------------------+-->


>--+->- ADD ------ FOREIGN KEY --- _____ ------ _ -- (column-name) -->
   !                              constraint-name
   ! >- REFERENCES ---->  _____ . _____ ----------------->
   !                          <creator.>table-name
   ! >-+-------------------------+------ ON DELETE -+- S - RESTRICT -+-+-->
   !   +--- _ --- (column-name) ---+                +- _ - CASCADE --+ !
   !                                                +- _ - SET NULL -+ !
   !                                                +- _ - NO ACTION + !
   +->- DROP ----- FOREIGN KEY --- _____ ----------------------------+
                                  constraint-name




 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Exec  Free                          Prev  Next  Canc
```

7   Once you have entered the required information for adding and/or dropping primary and/or
     foreign keys, press PF11 (Next). A screen is invoked, in which you can specify a RESTRICT ON
     DROP clause, add or drop a CHECK constraint, and/or drop any constraint:

```
  12:20:24                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                                 - Alter Table -


  >---+-- ADD --- _ --+----- RESTRICT ON DROP --------------------------------->
      !              !
      +-- DROP -- _ --+


  >------ ADD CHECK ----------- _ --------------------------------------------->
                              check-condition

  >------ DROP CHECK  --------- _____ --------------------------------------->
                              constraint-name


  >------ DROP CONSTRAINT ----- _____ --------------------------------------->
                              constraint-name



  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec  Free                          Prev        Canc
```

Alter Tablespace Function

The following example illustrates the use of the **Alter Tablespace** syntax input screen.

> **To invoke the Alter Tablespace function**

1   On the **Catalog Maintenance** screen, enter the code AL.

In the **Object** field, enter TS and press ENTER.

The **Alter Tablespace** screen is displayed, where you can specify the following:

```
 12:20:24                  ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                               - Alter Tablespace -

 >>------------- ALTER TABLESPACE -- _____ . _____ -------------------->
                                <database-name.>tablespace-name


              +-->- BUFFERPOOL ------- _____ ----------------+
              !                   bufferpool-name             !
   >---------+-->- CLOSE ------------ ___ ------------------+-------------->
              !                      ( YES or NO )           !
              +-->- DSETPASS --------- _____ ---------------+
              !                        password               !
              +-->- PART ------------- __ --------------------+
              !                       integer                 !
              +-->- FREEPAGE --------- ___ ------------------+
              !                       integer                 !
              +-->- PCTFREE ---------- __ --------------------+
              !                       integer                 !
              +-->- COMPRESS --------- ___ ------------------+
                                     ( YES or NO )

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec  Free                             Next  Canc
```

2   If you enter an S in the **bufferpool-name** field and press ENTER, a window is displayed, in
    which you can specify additional buffer pool names:

```
12:20:24                    ***** NATURAL TOOLS FOR DB2 *****                2009-10-30
                              - Alter Tablespace -
                                               +--------------------------------+
 >>------------- ALTER TABLESPACE -- _____ !                                 ↵
 !
                                  <database-na !     Valid values for          ↵
 !
                                             !      bufferpool-name:            ↵
 !
        +-->- BUFFERPOOL ------- S_____   ! --------------------------- ↵
 !
            !                   bufferpool-n !                                  ↵
 !
  >---------+-->- CLOSE ------------ ___ --- !  -  4KB buffer pools -           ↵
 !
            !                   ( YES or NO  !  BP0, BP1, BP2, ..., BP49        ↵
 !
        +-->- DSETPASS --------- _____ !                                     ↵
 !
            !                     passwor !  - 32KB buffer pools -             ↵
 !
        +-->- PART ------------- __ ---- !  BP32K, BP32K1, ..., BP32K9  ↵
 !
            !                   integer   !                                    ↵
 !
        +-->- FREEPAGE --------- ___ --- !  _____   Selection               ↵
 !
            !                   integer   !                                    ↵
 !
    +-->- PCTFREE ---------- __ -----+--------------------------------+
            !                   integer                              !
            +-->- COMPRESS --------- ___ -------------------+
                              ( YES or NO )

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
              Exit                                                        Canc
```

3   Once you are back in the first **Alter Tablespace** syntax input screen, press PF11 (Next) to go
     to the next screen:

```
 12:20:24                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - Alter Tablespace -


                            +- VCAT ----- _____ --+
                 +-->- USING -+         catalog-name  +--------------+
                 !            +- STOGROUP - _____ --+             !
                 !                        stogroup-name             !
    >---------+-->- PRIQTY ------------- ____ --------------------+------>-<
                 !                            integer             !
                 +-->- SECQTY ------------- ___ ----------------------+
                 !                            integer             !
                 +-->- ERASE ------------- ___ ----------------------+
                 !                          (YES or NO)            !
                 +-->- LOCKMAX ----------- _____ ------------------+
                 !                          (SYSTEM or integer)      !
                 +-->- LOCKSIZE ---+------- ____ --- LOCKMAX - _____ -+
                                   ! (PAGE or ROW)  (SYSTEM or integer)!
                                   +------- _____ --------------+
                                     (ANY, TABLE or TABLESPACE)

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec  Free                          Prev        Canc
```

4   On the second **Alter Tablespace** syntax input screen, you can now specify the LOCKMAX and
     LOCKSIZE options.

     Refer to the relevant DB2 literature by IBM for further details on the COMPRESS, LOCKSIZE and
     LOCKMAX clauses.

SQL Skeleton Members

SQL skeleton members are provided for processing the following SQL statements that are not
supported by the **Catalog Maintenance** function:

- CREATE AUXILIARY TABLE
- CREATE DISTINCT TYPE
- CREATE TRIGGER
- GRANT ALTERIN
- REVOKE ALTERIN

An SQL skeleton member is a Natural text object that contains an SQL skeleton that complies with
the DB2 SQL syntax rules as described in the relevant IBM literature. The replaceable items in the

SQL skeleton shown in lower-case characters must be filled with user input so that the skeleton becomes a valid SQL statement that can be executed in free mode (see *Free Mode*) or ISQL (see **Interactive SQL**). The skeleton text objects are delivered in the Natural system library SYSDB2, along with example SQL text objects.

7 Interactive SQL

The **Interactive SQL** function of the **Natural Tools for DB2** enables you to execute SQL statements dynamically.

Invoking the Interactive SQL Function

≫ **To invoke the Interactive SQL function**

■   On the **Natural Tools for DB2 Main Menu**, enter function code I.

The **Interactive SQL** screen is displayed:

```
 16:21:04                  ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                                  - Interactive SQL -




                        Code    Function
                        ----    -------------------------
                          I     SQL Input Member
                          O     Data Output Member
                          ?     Help
                          .     Exit
                        ----    -------------------------
                   Code.. _     Library .. SAG_____
                                Member ... _____




 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit                                                    Canc
```

The following functions are available:

Code	Description
I	Displays SQL members (text objects) in the interactive SQL input screen.
O	Displays output members (text objects) in the interactive SQL output screen.

The following parameters can be specified:

Parameter	Description
Library	Specifies the name of the current Natural library which contains the specified input/output members (text objects). Specification of libraries whose names begin with SYS is not allowed. The library name is preset with your Natural user ID.
Member	If a valid member name is specified, the corresponding member is displayed. If a value is specified followed by an asterisk (*), all input/output members in the current library whose names begin with this value are listed. If asterisk notation is specified only, a selection list of all input/output members in the current library is displayed. If the **Member** field is left blank, the empty SQL input/output screen is displayed.

SQL Input Members

≫ **To invoke the SQL Input Member function**

■   On the **Interactive SQL** screen, enter function code I and press ENTER.

Depending on what member (text object) name you have specified, different screens are displayed.

These screens are explained in the following sections.

ISQL Input Screen

If you leave the **Member** field blank, the empty **ISQL - Input** screen is invoked:

```
 16:21:56                  ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
  ISQL - Input         SAG                      S 01- --------------Columns 001 072
  ====>                                                    Scroll ===>  PAGE
  ***** **************************** top of data ****************************
  '''''
  '''''
  '''''
  '''''
  '''''
  '''''
  '''''
  '''''
  '''''
  '''''
  '''''
  '''''
  '''''
  '''''
  '''''
  '''''
  ***** *************************** bottom of data ***************************

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Setup Exit  Exec  Rfind Rchan -     +     Outpu               Canc
```

The **ISQL - Input** screen is a free-mode editor (see *Editing within the Natural Tools for DB2*) which provides a functionality similar to the one of the Software AG Editor. Using the editor you can enter or edit SQL statements via editor main and line commands. You can execute the SQL statements immediately from within the editor by pressing PF4 (Exec), or you can save them as an SQL member (text object) in a Natural library for later execution.

For information on the PF keys available, see *PF Key Settings*.

> **Note:** The PRINT command is not available in the SQL input screen.

Apart from the editor main and line commands, SQLCODE maintenance commands are also available to maintain SQL members in a Natural library; see *Global Maintenenance Commands*. With these maintenance commands, input members can be listed, retrieved, saved in a Natural library, copied, and purged. They are entered in the command line of the input screen.

You can also obtain a list of the available maintenance commands by entering the help character, that is, a question mark (?), in the command line of the input screen. A window is displayed from which the desired command can be selected. The window can be scrolled forwards by pressing PF8, or backwards by pressing PF7.

```
 12:22:12               ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
  ISQL - Input        SAG                    S 01- --------------Columns 001 072
  ====> ?                                              Scroll ===>  PAGE
  ***** ********************* +--------------------------------+*************
                             !                                !
                             !   _  List <*,member>           !
                             !   _  READ <member>             !
                             !   _  SAVE <member>             !
                             !   _  COPY <member>             !
                             !   _  Purge <member>            !
                             !   _  LIBrary <library>         !
                             !   _  SELect <TB,CO> name1 name2 !
                             !                                !
                             +--------------------------------+



  ***** *************************** bottom of data ***************************

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Setup Exit  Exec  Rfind Rchan -     +     Outpu            Canc
```

To assist you in coding your SQL member, existing DB2 tables and columns can be listed using the SELECT command. From the list, you can include table and column names into the editor.

The SELECT command is available for table and column selection:

Command	Description
SELECT TABLE [creator.]name	Selects all tables with the specified creator (optional) and name. For both *creator* and *name*, you can specify a value followed by an asterisk (*), and all tables whose names begin with this value are selected. . If you specify asterisk notation only, all existing tables are selected. If you specify a table name without a creator, all tables with the specified name are selected, regardless of their creator.
SELECT COLUMN creator.name	Selects all columns of the table *creator.name*. Since the table must be uniquely identified, asterisk notation cannot be used.

Sample Input Screen with Table Listing Window

```
12: +-----------------------------------------------------------------+
 ISQ ! Tab:                                                          !
==== ! SYSIBM.*                                                      !
 *** !   Table Name                        Creator                  !
 ''' ! _ SYSDATABASE                       SYSIBM                    !
 ''' ! _ SYSDATATYPES                      SYSIBM                    !
 ''' ! _ SYSDBAUTH                         SYSIBM                    !
 ''' ! _ SYSDBRM                           SYSIBM                    !
 ''' ! _ SYSDUMMY1                         SYSIBM                    !
 ''' ! _ SYSDUMMYA                         SYSIBM                    !
 ''' ! _ SYSDUMMYE                         SYSIBM                    !
 ''' ! _ SYSDUMMYU                         SYSIBM                    !
 ''' ! _ SYSFIELDS                         SYSIBM                    !
 ''' ! _ SYSFOREIGNKEYS                    SYSIBM                    !
 ''' ! _ SYSINDEXES                        SYSIBM                    !
 ''' ! _ SYSINDEXES_HIST                   SYSIBM                    !
 ''' ! _ SYSINDEXPART                      SYSIBM                    !
 ''' ! _ SYSINDEXPART_HIST                 SYSIBM                    !
 ''' ! _ SYSINDEXSTATS                     SYSIBM                    !
 ''' ! _ SYSINDEXSTATS_HIST                SYSIBM                    !
 *** ! _ SYSJARCLASS_SOURCE                SYSIBM                    !
     ! _ SYSJARCONTENTS                    SYSIBM                    !
Ente !                                                              !
     +-----------------------------------------------------------------+
```

From the table list, you can select a table for display of its columns by marking it with `C` in front of the table name. The columns of a table are listed together with their type and length. A creator or table name longer than 32 characters will be truncated. This will be indicated by a > symbol at the end of the creator or table name.

Sample Input Screen with Column Listing Window

```
12:27:08             ** +----------------------------------------------------+
ISQL - Input      GGS ! Tab: SYSIBM.SYSTABLES                               !
=====>                !                                                     !
***** ***************** !  Column Name                         Type    Len  !
A     SELECT          ! M NAME                                 VARCHAR 128  !
00002 SYSIBM.SYSTABLES  ! M CREATOR                              VARCHAR 128  !
***** ***************** ! M TYPE                                 CHAR    1    !
                      ! M DBNAME                               VARCHAR 24   !
                      ! M TSNAME                               VARCHAR 24   !
                      ! _ DBID                                 SMALLINT 2   !
                      ! _ OBID                                 SMALLINT 2   !
                      ! _ COLCOUNT                             SMALLINT 2   !
                      ! _ EDPROC                               VARCHAR 24   !
                      ! _ VALPROC                              VARCHAR 24   !
                      ! _ CLUSTERTYPE                          CHAR    1    !
                      ! _ CLUSTERRID                           INTEGER 4    !
                      ! _ CARD                                 INTEGER 4    !
                      ! _ NPAGES                               INTEGER 4    !
                      ! _ PCTPAGES                             SMALLINT 2   !
                      ! _ IBMREQD                              CHAR    1    !
                      ! _ REMARKS                              VARCHAR 762  !
                      ! _ PARENTS                              SMALLINT 2   !
Enter-PF1---PF2---PF3---P !                                                 !
     Help  Setup Exit  E +----------------------------------------------------+
```

If you want to copy table or column names from a selection list into the editor, mark the corresponding table or column with M as shown on the previous screen. The table or column names are copied either after or before the line marked with an A or a B respectively, or to the top of the displayed data.

Sample Input Screen with Copied Column Names

```
 12:29:44               ** +----------------------------------------------+
 ISQL - Input       GGS ! Tab: SYSIBM.SYSTABLES                          !
 =====>                  !                                                !
 ***** ****************** !   Column Name                  Type     Len  !
 A     SELECT           ! _ NAME                         VARCHAR  128  !
 00002 NAME             ! _ CREATOR                      VARCHAR  128  !
 00003 , CREATOR        ! _ TYPE                         CHAR       1  !
 00004 , TYPE           ! _ DBNAME                       VARCHAR   24  !
 00005 , DBNAME         ! _ TSNAME                       VARCHAR   24  !
 00006 , TSNAME         ! _ DBID                         SMALLINT   2  !
 00007 SYSIBM.SYSTABLES ! _ OBID                         SMALLINT   2  !
 ***** ****************** ! _ COLCOUNT                     SMALLINT   2  !
                        ! _ EDPROC                       VARCHAR   24  !
                        ! _ VALPROC                      VARCHAR   24  !
                        ! _ CLUSTERTYPE                  CHAR       1  !
                        ! _ CLUSTERRID                   INTEGER    4  !
                        ! _ CARD                         INTEGER    4  !
                        ! _ NPAGES                       INTEGER    4  !
                        ! _ PCTPAGES                     SMALLINT   2  !
                        ! _ IBMREQD                      CHAR       1  !
                        ! _ REMARKS                      VARCHAR  762  !
                        ! _ PARENTS                      SMALLINT   2  !
 Enter-PF1---PF2---PF3---P !                                              !
      Help  Setup Exit  E +----------------------------------------------+
```

Fixed Mode with Interactive SQL

All fixed-mode input screens from the *Catalog Maintenance* part of the **Natural Tools for DB2** are available as help maps within the *Interactive SQL* part.

To invoke this help facility, enter the name of the SQL statement you want to create in the command line of your **ISQL - Input** screen, for example, CREATE TABLE or CR TB for the CREATE TABLE command.

The same command abbreviations apply as with the **Catalog Maintenance** function.

If you enter CREATE TABLE or CR TB, the **Create Table** screen is invoked:

```
  01:22:12                   ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - Create Table -                              1  / 9

 >>- CREATE TABLE - SAG_____ . DEMOTABLE_____ --------------------------->
                    <creator.>table-name
 >+--- LIKE ------- _____ . _____ +----------------------+-+>
  !                <creator.>table/view-name   +- _ - INCLUDING IDENTITY + +
  !                                                                         !
  +( COL1_____   CHAR_____ ( 20_____ ) _ - __ - _ - __ - _ , +
  +- COL2_____   INTEGER_____ ( _____ ) _ - NN - _ - 2_ - _ , +
  +- COL3_____   SMALLINT_____ ( _____ ) _ - NN - _ - 1_ - _ , +
  +- COL4_____   CHAR_____ ( 2_____ ) S - __ - _ - __ - _ , +
  +- COL5_____   VARCHAR_____ ( 30_____ ) _ - NN - _ - 3_ - _ , +
  +- COL6_____   DECIMAL_____ ( 2,5_____ ) _ - __ - X - __ - _ , +
  +- COL7_____   FLOAT_____ ( _____ ) _ - NN - _ - __ - _ , +
  +- COL8_____   DATE_____ ( _____ ) _ - __ - _ - __ - _ , +
  +- COL9_____   TIME_____ ( _____ ) _ - __ - _ - __ - _ , +
  +- _____   _____ ( _____ ) _ - __ - _ - __ - _ , +
        column-name            format          length    S/M  NN  fld  PK/ R/C
                                                           B       proc UK  D/G
 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Error Exit  Exec  Free  --    -     +    ++             Next  Canc
```

If you have entered data for a complete SQL statement, you can generate an SQL statement from the entered data and include it into the **ISQL - Input** screen.

Using PF4 (Incl), you include the generated SQLCODE and remain on the **Create Table** screen.

Using PF5 (IBack), you include the generated SQLCODE and return to the **ISQL - Input** screen.

Retrieve an SQL Member

If you specify a unique member (text object) name in the **Member** field of the **Interactive SQL** screen, the corresponding SQL member is listed on the input screen. If no member exists with the specified name, a corresponding message is returned.

Sample SQL Member Listed in Input Screen

```
 01:03:23                 ***** NATURAL TOOLS FOR DB2 *****             2009-10-30
  ISQL - Input         SAG(TESTSEQ)           S 01- --------------Columns 001 072
  ====>                                                      Scroll ===>  PAGE
  ***** **************************** top of data ****************************
  00001 CREATE TABLE DEMOTABLE
  00002   (COL1                   CHAR(8),
  00003    COL2                   INTEGER
  00004    ) IN DATABASE DEMO;
  00005 INSERT INTO DEMOTABLE
  00006    VALUES ('AAAAA',1);
  00007 * INSERT INTO DEMOTABLE
  00008 *   VALUES ('BBBBB',2);
  00009 SELECT FROM DEMOTABLE;
  00010 DROP TABLE DEMOTABLE;
  ***** *************************** bottom of data ****************************




 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Setup Exit  Exec  Rfind Rchan -     +     Outpu             Canc
```

Listed SQL members can be purged, modified, executed, or saved.

An asterisk (*) in front of a statement line turns this line into a comment line, which means that the corresponding SQLCODE is not considered for execution.

List of SQL Members

If you specify a value followed by an asterisk (*) in the **Member** field of the **Interactive SQL** screen, a list of all SQL input members (text objects) in the current library whose names begin with this value is displayed.

If you specify an asterisk (*), a list of all SQL input members in the current library is displayed.

Sample SQL Input Member Selection List

```
15:06:14                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                            Select Member

      C       Member      Type         User      Date         Time
      -       --------    -----------  --------  ----------   --------
      _       CRAXTB      SQL          SAG       2009-10-30   13:48:53
      _       CRDITY      SQL          SAG       2009-10-30   13:39:14
      _       CRPRQE      SQL          SAG       2009-10-30   13:54:21
      _       CRTB        SQL          SAG       2009-10-30   13:48:14
      _       CRTRIG      SQL          SAG       2009-10-30   13:53:01
      _       CRTRIG2     SQL          SAG       2009-10-30   13:14:10
      _       DRPRQE      SQL          SAG       2009-10-30   13:55:04
      _       DRPRQE2     SQL          SAG       2009-10-30   13:50:30
      _       GGSDTYPE    SQL          SAG       2009-10-30   13:52:10
      _       GRSHPR      SQL          SAG       2009-10-30   13:28:01
      _       RESHPR      SQL          SAG       2009-10-30   13:31:05
      _       SELPROCS    SQL          SAG       2009-10-30   13:09:05
      _       SELTABS     SQL          SAG       2009-10-30   13:56:22




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Cont             Exit                                                >     Canc
```

From the input screen selection list, SQL members can be selected for display by marking them with an S.

If the list has been invoked by a PURGE command, members can be purged by marking them with a P.

By pressing PF11 (>), you can switch from the default view of the **Select Member** screen as shown above to the extended view with the first line of each member displayed in the **Description** column:

```
 15:09:17              ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              Select Member

     C       Member      Description (first line of member)
     -       --------    ----------------------------------------------------
     _       CRAXTB      CREATE AUXILIARY TABLE aux-table-name
     _       CRDITY      CREATE DISTINCT TYPE distinct-type-name
     _       CRPRQE      * ALL PROCEDURES FROM QARNDB31(10,110), WHICH HAVE 'C
     _       CRTB        CREATE TABLE NEWTYPE
     _       CRTRIG      CREATE TRIGGER trigger-name NO CASCADE BEFORE|
     _       CRTRIG2     CREATE TRIGGER trigger-name (NO CASCADE BEFORE|
     _       DRPRQE      * ALL PROCEDURES FROM QARNDB31(10,110), WHICH HAVE 'C
     _       DRPRQE2     DROP PROCEDURE CALLN2 RESTRICT;
     _       GGSDTYPE    SELECT COLTYPE,LENGTH,LENGTH2,DATATYPEID,SOURCETYPEID
     _       GRSHPR      GRANT ALTERIN [, CREATEIN] [, DROPIN]
     _       RESHPR      REVOKE ALTERIN [, CREATEIN] [, DROPIN]
     _       SELPROCS    SELECT * FROM SYSIBM.SYSPROCEDURES
     _       SELTABS     SELECT * FROM SYSIBM.SYSTABLES




Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Cont           Exit                                        <           Canc
```

The first line of a member can be the first line of an SQL statement or a comment line which provides more information on the member.

Data Output Members

≫ **To invoke the Data Output Member function**

■   On the **Interactive SQL** screen, enter function code 0 and press ENTER.

Depending on what member (text object) name you have specified, different screens are displayed.

These screens are explained in the following sections.

Data Output Screen

If you leave **Member** field of the **Interactive SQL** screen blank, the empty **ISQL - Output** screen is invoked.

```
 15:19:15                 ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
  ISQL - Output        SAG                      S 02- --------------Columns 001 072
  ====>                                                      Scroll ===>  PAGE
  ***** ***************************** top of data *****************************
  ***** ************************** bottom of data *****************************









 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind Rchan -      +           <     >    Canc
```

From the data output screen you have access to output data members only. Output members consist of data retrieved from the database as a result of executed SQL statements. These data can be browsed and saved for later use as output members on the Natural system file FUSER. In addition to the data retrieved from the database, output members also contain DB2 status information, and the executed SQL member.

If you execute an SQL statement, the results are automatically shown on the output screen. Thus, you can enter the interactive SQL output screen also by executing an SQL statement from the input screen. From the output screen you can return to the input screen by pressing PF3 (Exit).

For information on the other PF keys available, see *PF Key Settings*.

The maintenance commands available for output members can be displayed and selected in a window, too; see *Global Maintenenance Commands*. The window is invoked by entering the help character, that is, a question mark (?), in the command line of the output screen.

```
 15:57:59                  ***** NATURAL TOOLS FOR DB2 *****                2009-10-30
  ISQL - Output       SAG                          S 02- --------------Columns 001 072
  ====> ?                                                      Scroll ===>   PAGE
  ***** ********************* +--------------------------------+*************
  ***** ********************* !                                !*************
                             !   _  List <*,member>           !
                             !   _  READ <member>             !
                             !   _  SAve <member>             !
                             !   _  Purge <member>            !
                             !   _  LIBrary <library>         !
                             !                                !
                             +--------------------------------+




 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit        Rfind Rchan -     +           <     >    Canc
```

Apart from the maintenance commands, only browse commands are available (see *Editing within the Natural Tools for DB2*), since output members cannot be modified. Both browse and maintenance commands are entered in the command line of the output screen.

If an output member is too large to fit on your terminal screen, you can use the FIX ON *n* command to keep the first *n* characters on the screen when scrolling to the left or to the right.

Retrieve an Output Member

If you specify a unique member name in the **Member** field of the **Interactive SQL** screen, the corresponding output member is listed on the output screen. If no member exists with the specified name, a corresponding message is returned.

Sample Output Member Listed in Output Screen

```
 16:27:12                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
  ISQL - Output        SAG(TESTSEQO)         S 02- --------------Columns 001 072
  ====>                                                      Scroll ===>  PAGE
  ***** ***************************** top of data *****************************
  00001 CREATE TABLE DEMOTABLE
  00002   (COL1                  CHAR(8),
  00003    COL2                  INTEGER
  00004   ) IN DATABASE DEMO
  00005 ------------------------------------------------------------------------
  00006 STATEMENT WAS SUCCESSFUL, SQLCODE = 0
  00007 ------------------------------------------------------------------------
  00008 INSERT INTO DEMOTABLE
  00009   VALUES ('AAAAA',1)
  00010 ------------------------------------------------------------------------
  00011 STATEMENT WAS SUCCESSFUL, SQLCODE = 0
  00012 ------------------------------------------------------------------------
  00013 SELECT FROM DEMOTABLE
  00014 ------------------------------------------------------------------------
  00015 COL1          COL2
  00016 ------------------------------------------------------------------------
  00017 AAAAA            1

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind Rchan -     +          <     >     Canc
```

List of Output Members

If you specify a value followed by an asterisk (*) in the **Member** field of the **Interactive SQL** screen, a list of all data output members in the current library whose names begin with this value is displayed.

If you specify asterisk notation only, a list of all data output members in the current library is displayed.

Sample Data Output Member Selection List

```
 16:24:02                   ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                                   Select Member


       C     Member      Type         User      Date        Time
       -     --------    -----------  --------  --------    -----------
       _     AAAA        SQL-RESULT   SAG       2009-10-30  13:54:54
       _     ADEMVIEW    SQL-RESULT   SAG       2009-10-30  14:01:09
       _     AIRCRAFT    SQL-RESULT   SAG       2009-10-30  10:01:32
       _     BBBB        SQL-RESULT   SAG       2009-10-30  15:25:14
       _     BSP1        SQL-RESULT   SAG       2009-10-30  14:57:11
```

From the output member selection list, output members can be selected for display by marking them with an S.

If the list has been invoked by a PURGE command, members can be purged by marking them with a P.

Processing SQL Statements

SQL input members (text objects) can only be accessed from the **ISQL - Input** screen. They are executed from the input screen against DB2 by pressing PF4 (Exec).

After execution, the data output screen appears which contains the results of the executed SQL member.

If an SQL member consists of more than one SQL statements, the individual statements must be separated by a semicolon. They can be executed one by one or all together at the same time.

To choose the form of execution, a window is provided which can be invoked by pressing PF2 (Setup).

```
 16:29:12              ***** NATURAL TOOLS FOR DB2 *****          2009-10-30
 ISQL - Input        SAG(TESTSEQ)           S 01- --------------Columns 001 072
 ====>                            +----------------------------------------+
 ***** ***************************** !                                      !
 00001 CREATE TABLE DEMOTABLE        !  _  Execute statements one by one    !
 00002   (COL1                CHAR(8 !  X  Execute all statements together  !
 00003    COL2                INTEGE !                                      !
 00004   ) IN DATABASE DEMO;         !  _  Optional Commit/Rollback         !
 00005 INSERT INTO DEMOTABLE         !  X  Automatic Commit/Rollback        !
 00006   VALUES ('AAAAA',1);         !                                      !
 00007 * INSERT INTO DEMOTABLE       !  _  Ignore positive SQLCODEs         !
 00008 *   VALUES ('BBBBB',2);       !                                      !
 00009 SELECT FROM DEMOTABLE;        ! Text for NULL values    : <NULL>__   !
 00010 DROP TABLE DEMOTABLE;         ! Sql termination character : ;        !
 ***** *************************** b ! Maximum length of columns :    _____ !
                                     ! Maximum number of rows    : _____  !
                                     ! DB2 cost limit            : _____  !
                                     !                                      !
                                     ! Database type(DB2,CNX)    : DB2      !
                                     ! Header Line every 15___ Data Lines   !
                                     ! Record Length Data Session: _250     !
                                     !                                      !
 Enter-PF1---PF2---PF3---PF4---PF5---PF+----------------------------------------+
       Help  Setup Exit  Exec  Rfind Rchan -     +      Outpu           Canc
```

Below is information on the options provided:

- Execute Statements One By One
- Execute All Statements Together
- Automatic Commit/Rollback
- Optional Commit/Rollback
- Text For NULL Values
- SQL Termination Character
- Maximum Length of Columns
- Maximum Number of Rows
- DB2 Cost Limit
- Header Line Every n Data Lines

Execute Statements One By One

After each SQL statement the output screen is shown. From the output screen, you can either execute the next SQL statement from the input screen by pressing PF4 (Next), or skip the remaining SQL statements and return to the input screen immediately by pressing PF3 (Exit).

Execute All Statements Together

All statements are executed immediately one after the other. The output screen shows the results of all statements together.

Statements containing cursor names, host variables, or parameter markers cannot be executed with interactive SQL. Also not executed are statements available as embedded SQL only; that is, statements whose functions are automatically performed by Natural.

These statements are:

CLOSE
CONNECT
DECLARE
DELETE WHERE CURRENT OF CURSOR
DESCRIBE
EXECUTE
FETCH
INCLUDE
OPEN
PREPARE
SELECT INTO
SET *host-variable*
SET CURRENT PACKAGESET
UPDATE WHERE CURRENT OF CURSOR
WHENEVER

Automatic Commit/Rollback

If you select **Automatic Commit/Rollback**, each modification of the database is automatically either committed or rolled back, depending on whether all the SQL statements involved execute successfully. If so, an `SQL COMMIT WORK` command is executed; if not, an `SQL ROLLBACK` command backs out all database modifications since the last commit point.

Optional Commit/Rollback

If you select **Optional Commit/Rollback**, a window is invoked after each SQL statement, offering you the option to either commit or roll back the resulting database modifications shown on the screen.

> **Note:** Since under CICS and IMS TM each terminal I/O results in a `SYNCPOINT`, the optional commit/rollback feature only applies in a TSO environment.

In all environments, you can include `SQL COMMIT` and `ROLLBACK` commands in your input member, too. Under CICS and IMS TM, however, these commands are translated into the corresponding TP-monitor calls.

Text For NULL Values

The text that is to be shown for `NULL` values can be specified here; the default string is `---`.

SQL Termination Character

If you enter multiple SQL statement, they need to be separated. The default statement termination character is the semi-colon (;).

Maximum Length of Columns

Limits the length for a single column to *n* characters. This limit only applies to character data. `DATE`, `TIME`, or `NUMERIC` columns are not truncated. The value `0` indicates that no limit exists.

Maximum Number of Rows

Limits the number of rows returned by one `SELECT` statement. The value `0` indicates that no limit exists.

DB2 Cost Limit

Sets a limit for the DB2 cost estimate. SELECT statements which exceed this limit are not executed. The value 0 indicates that no limit exists.

Header Line Every n Data Lines

For SELECT statements, you can specify that every *n* data lines a header line is inserted with the names of the selected columns. If *n* is set to 0, only one header line is displayed at the top of the data.

Record Length Data Session

The record length (*n*) for the output session can be specified. If the specified record length is smaller than the record length of the output data, the output records are truncated accordingly. The truncation of records is indicated by a greater than character (>) as the leftmost character in the first line beneath each header line. The default value for *n* is 250 bytes.

PF-Key Settings

The following PF-key settings apply to the **ISQL - Input** screen:

Key	Setting	Function
PF2	Setup	Invokes a window with further processing options.
PF4	Exec	Executes the SQL member (text object) currently on the input screen.
PF5	Rfind	Repeats the last executed FIND command.
PF6	Rchan	Repeats the last executed CHANGE command.
PF7	-	Scrolls the display one page backward.
PF8	+	Scrolls the display one page forward.
PF9	Outpu	Invokes the output member (text object) selection list directly from within the input screen.

Apart from PF2 (Setup), PF4 (Exec), and PF9 (Outpu), the same PF-key settings apply to the **ISQL - Output** screen, too. In addition, the following PF-key settings are available:

Key	Setting	Function
PF4	Next	Executes the next SQL statement if an SQL member consists of more than one statement, and if you have chosen to execute them one after the other. If not, the setting for PF4 is left blank.
PF10	<	Scrolls the display of the output screen to the left.
PF11	>	Scrolls the display of the output screen to the right.

Unloading Interactive SQL Results

Results from interactive SQL are unloaded and written to a data set referred to by DD name CMWKF01 in batch mode using the UNLDDATA command.

CMWKF01 should be of variable record format; the record length depends on the size of the SQL output member (text object) and can range from 250 to 4000 bytes.

≫ To unload results from interactive SQL

1   Logon to the Natural system library SYSDB2.

2   In the command line, enter the command UNLDDATA and press ENTER.

The **Unload SQL Results** menu is displayed:

```
  16:53:20                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                                - Unload SQL Results -




                           Code Function
                           ---- ------------------
                            U   Unload SQL Results
                            .   Exit
                           ---- ------------------
                     Code .. _   Library .. _____
                                 Member ... _____




  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                     Exit                                                    Canc
```

The following function is available:

Code	Description
U	Unloads results from interactive SQL execution.

The following parameters apply:

Parameter	Description
Library	Specifies the name of the Natural library from which the specified output members are to be unloaded. You cannot specify libraries whose names begin with SYS. This parameter must be specified.
Member	Specifies the name(s) of the output member(s) to be unloaded. This parameter must be specified.

8 Retrieval of System Tables

⚠️   **Important:** Before you use the **Retrieval of System Tables** function, refer to *LISTSQL and Explain Functions* in the section *Special Requirements for Natural Tools for DB2* in *Installing Natural for DB2 on z/OS*.

The DB2 system tables provide information on the contents of your DB2 system. The **Retrieval of System Tables** function enables you to:

- display information on DB2 objects without coding SQL queries;
- easily access related objects, such as indexes of a table.

The DB2 objects supported by the **Retrieval of System Tables** function are database, tablespace, table, index, column, plan, check constraints, statistic tables, package, and DBRM (database request module), as well as access rights to and relationships between these objects.

DB2 objects are presented in one of the following two ways:

- As selection lists, where all objects are of the same type, and where commands can be issued to display related objects.
- You can list databases, tables, plans, and packages by name. From the database listings, you can invoke listings of the tablespaces or tables of a database. From the table listing, you can invoke listings of the columns and indexes of a table. From the plan listing, you can invoke listings of the DBRMs of a plan, of the package list of a plan, of the tables and indexes used by a plan, and of the systems which are enabled or disabled for a plan. From the package listing, you can invoke listings of the tables and indexes used in a package and of the systems which are enabled or disabled for a package. From the database, table, plan, or package listings, you can also investigate who is authorized to access a DB2 object. In addition, the **User Authorization** menu enables you to list all existing access rights by user ID.
- As reports, which merely contain information on different types of DB2 objects, and where only browse commands can be issued.

The most important browse commands can also be issued via PF keys; see *Editing within the Natural Tools for DB2*.

This section covers the following topics:

Invoking the Retrieval of System Tables Function

≫ **To invoke the Retrieval of System Tables function**

- On the **Natural Tools for DB2 Main Menu**, enter function code R.

  The **Retrieval of System Tables** screen is displayed:

```
 16:31:56              ***** NATURAL TOOLS FOR DB2 *****              2006-05-25
                        - Retrieval of System Tables -

                     Code   Function              Parameter

                      D     List Databases        Database
                      K     List Packages         Collection, Name
                      P     List Plans            Plan
                      T     List Tables           Tbreator, Tbname
                      U     User Authorizations
                      S     Statistic Tables
                      ?     Help
                      .     Exit

            Code .. _     Database Name ....... _____
                          Package Collection .. _____
                          Package Name ........ _____
                           Plan Name .......... _____
                          Table Creator ....... _____
                          Table Name .......... _____

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Setup Exit                                                    Canc
```

With PF2 (Setup) the maximum length of one column and the number of fixed characters when scrolling left may be specified. The default values for both parameters may be changed in the CONFIG subprogram in library SYSDB2.

When a column value is longer than the maximum length, it will be truncated and marked with a greater than sign (>) in the case of strings truncated at the right end or a less than sign (<) in the case of numbers truncated at the left end.

Note, that for further commands on a line, for example, the line command I, only the visible value can be taken as input. This means that commands on lines will fail, when values for further processing are truncated.

```
 16:31:56                 ***** NATURAL TOOLS FOR DB2 *****              2007-10-05
                          - Retrieval of System Tables -

                    Code   Function            Parameter
                              +------Retrieval of System Tables------+
                     D     List Dat !                                !
                     K     List Pac ! Maximum length of columns ... ____8  !
                     P     List Pla ! Number of fixed characters .. ____0  !
                     T     List Tab !                                !
                     U     User Aut !                                !
                     S     Statisti +------------------------------------+
                     ?     Help
                     .     Exit

                Code .. _    Database Name ....... _____
                             Package Collection .. _____
                             Package Name ........ _____
                              Plan Name ........... _____
                             Table Creator ....... _____
                             Table Name .......... _____

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Setup Exit                                                    Canc
```

The following functions are available:

Code	Description
D	Lists databases defined in the DB2 catalog.
K	Lists packages defined in the DB2 catalog.
P	Lists plans defined in the DB2 catalog.
S	Statistic tables.
T	Lists tables defined in the DB2 catalog.
U	Provides information on which user(s) can access which DB2 objects.

The following parameters must be specified as selection criteria:

Parameter	Description
Database Name	The name of the database to be listed. Asterisk notation (*) for range specification is possible. The Database Name parameter is relevant to the **List Databases** function only.
Package Collection	The collection of the package to be listed. Asterisk notation (*) for range specification is possible. The Package Collection parameter is relevant to the **List Packages** function only.

Parameter	Description
Package Name	The name of the package to be listed. Asterisk notation (*) for range specification is possible. The Package Name parameter is relevant to the **List Packages** function only.
Plan Name	The name of the plan to be listed. Asterisk notation (*) for range specification is possible. The Plan Name parameter is relevant to the **List Plans** function only.
Table Creator	The name of the creator of the table(s) to be listed. Asterisk notation (*) for range specification is possible. The Table Creator parameter is relevant to the **List Tables** function only.
Table Name	The name of the table to be listed. Asterisk notation (*) for range specification is possible. The Table Name parameter is relevant to the **List Tables** function only.

List Databases

≫ **To invoke the List Databases function**

1 On the **Retrieval of System Tables** screen, enter function code D.

2 Specify the name of the database(s) to be listed.

   ■ If a value followed by an asterisk is specified, all databases defined in the DB2 catalog whose names begin with this value are listed.

   ■ If asterisk notation is specified only, all databases defined in the DB2 catalog are listed.

```
 16:32:24              ***** NATURAL TOOLS FOR DB2 *****            2007-10-05
 DATABASES *                                 S 01     Row 0 of 25 Columns 001 059
 ====>                                                      Scroll ===>  PAGE
   DATABASE CREATOR    STOGROUP BPOOL    DBID CREATEDBY ROSHARE TIMESTAMP GR
** ***************************** top of data ******************************
 __ DEMO      DEFAULT    SYSDEFLT BP0      269 DEFAULT            0001-01-0>
 __ DEMODB    SAG2       SYSDEFLT BP0      273 SAG2               0001-01-0>D8
 __ DEVELOP   SAG        DEVELOP  BP0      260 SAG                0001-01-0>DB
 __ ECHDB01   SAG2       SYSDEFLT BP0      272 SAG2               0001-01-0>
 __ EFGDB     SAG        SYSDEFLT BP0      263 SAG                0001-01-0>
 __ HBUTST    SAG2       SYSDEFLT BP0      275 SAG2               0001-01-0>
 __ PLANTAB   SAG2       SYSDEFLT BP0      270 SAG2               0001-01-0>
 __ Predict   SAG2       SYSDEFLT BP0      262 SAG2               0001-01-0>
 __ QA        SAG2       SYSDEFLT BP0      265 SAG2               0001-01-0>
 __ SAGDB04   SYSIBM     SYSDEFLT BP0        4 SYSIBM             0001-01-0>
 __ SAGDB06   SYSIBM                         6 SYSIBM             0001-01-0>
 __ SAGDB07   SAG1       SYSDEFLT BP0        7 SAG1               0001-01-0>
 __ SAGDDF    SAG1       SYSDEFLT BP0      257 SAG1               0001-01-0>
 __ SAGRLST   SAG1       SYSDEFLT BP0      256 SAG1               0001-01-0>
 __ SAG8D22A  SAG1       SAG8G220 BP0      258 SAG1               0001-01-0>
 __ SAG8D22P  SAG1       SAG8G220 BP0      259 SAG1               0001-01-0>


 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help       Exit       Rfind            -     +           <     >    Canc
```

The following line commands are available on the database listing screen. Line commands are entered in front of the desired database(s):

Command	Description
I	Displays information on a database.
S	Selects a database to be used with main commands (see below).
U	Unselects a database.
AU	Displays information on access rights to a database.
TB	Displays all tables defined in a database.
TS	Displays all tablespaces defined in a database.

The listings of tables or tablespaces displayed as a result of the TB or TS command can be used for further processing, whereas the contents of the screens displayed as a result of the AU or I command are for information purposes only.

A list of all line commands available with the **List Database** function can be invoked as a window by entering the help character, that is, a question mark (?), in front of any of the listed databases.

The commands AU, TB, and TS can also be used as main commands. Main commands are entered in the command line of the database list screen and apply to all databases previously selected with the line command S.

A further main command is the INFO command, which is the equivalent of the I line command, but displays information on all previously selected databases. Instead of being displayed, all information resulting from the I or INFO commands can also be marked for printing. Even if already displayed, information can be printed by issuing the PRINT command.

```
 16:32:24               ***** NATURAL TOOLS FOR DB2 *****          2007-10-05
  DATABASES *                              S 01    Row 0 of 25 Columns 001 059
  ====>                                                      Scroll ===>  PAGE
    DATABASE CREATOR   STOGROUP BPOOL   DBID CREATEDBY ROSHARE TIMESTAMP GR
 ** **** +------------------------------------------------------------+ **********
  I_ DEMO !                                                      ! 01-01-0>
  __ DEMO !                   Select what to display             ! 01-01-0>D8
  __ DEVE !                                                      ! 01-01-0>DB
  __ ECHD !                                                      ! 01-01-0>
  __ EFGD !               _ authorizations for database          ! 01-01-0>
  __ HBUT !               _ tablespaces in database              ! 01-01-0>
  __ PLAN !               _ tables      in database              ! 01-01-0>
  __ PRED !                                                      ! 01-01-0>
  __ QA   !                                                      ! 01-01-0>
  __ SAGD !                                                      ! 01-01-0>
  __ SAGD !                   Mark  _ to print output            ! 01-01-0>
  __ SAGD !                                                      ! 01-01-0>
  __ SAGD +------------------------------------------------------------+ 01-01-0>
  __ SAGRLST  SAG1        SYSDEFLT BP0        256  SAG1            0001-01-0>
  __ SAG8D22A SAG1        SAG8G220 BP0        258  SAG1            0001-01-0>
  __ SAG8D22P SAG1        SAG8G220 BP0        259  SAG1            0001-01-0>


 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit        Rfind       -     +            <     >    Canc
```

A list of all main commands available with the **List Database** function can be invoked as a window by entering the help character, that is, a question mark (?), in the command line of the database list screen.

List Tablespaces

The function to list tablespaces is not part of the **Retrieval of System Tables** main menu.

≫ **To list tablespaces**

■   Issue the "TS" command on the database listing screen only.

A tablespace listing screen is displayed, for example:

```
 16:35:07                ***** NATURAL TOOLS FOR DB2 *****              2006-05-25
 TABLESPACES IN DATABASE DB2DEMO            S 02     Row 0 of 2 Columns 032 075
 ====>                                                      Scroll ===>  PAGE
    DATABASE NAME       CREATOR  BPOOL   PGSIZE PARTITIONS NTABLES SEGSIZE LO
 ** ***************************** top of data ******************************
 __ DB2DEMO  AUTOMOBI    SAG     BPO          4          0       1       0 A
 __ DB2DEMO  EMPLOYEE    SAG     BPO          4          0       1       0 A
 ** *************************** bottom of data ****************************
```

The following line commands are available on the tablespace listing screen. Line commands are entered in front of the desired tablespace(s):

Command	Description
I	Displays information on a tablespace.
S	Selects a tablespace to be used with main commands.
U	Unselects a tablespace.
PT	Displays all partitions of a tablespace.
TB	Displays all tables defined in a tablespace.

The **listings of tables** displayed as a result of the TB command can be used for further processing, whereas the listings resulting from the I and PT commands are for information purposes only.

A list of all line commands available on the tablespace listing screen can be invoked as a window by entering the help character, that is, a question mark (?), in front of any of the listed tablespaces.

The commands PT and TB can also be used as a main commands entered on the command line of the tablespace listing screen. Main commands apply to all tablespaces previously selected with the line command S.

A further main command is the INFO command, which is the equivalent of the I line command, but displays information on all previously selected tablespaces. Instead of being displayed, all information resulting from the I or INFO commands can also be marked for printing. Even if already displayed, information can be printed by issuing the PRINT command.

```
 16:35:07            ***** NATURAL TOOLS FOR DB2 *****           2006-05-25
 TABLESPACES IN DATABASE DB2DEMO             S 02     Row 0 of 2 Columns 032 075
 ====>                                                  Scroll ===>  PAGE
   DATABASE NAME        CREATOR  BPOOL   PGSIZE PARTITIONS NTABLES SEGSIZE LO
** **** +------------------------------------------------------------+ **********
 __ DB2D !                                                    !      0 A
 __ DB2D !                Select what to display              !      0 A
** **** !                                                    ! **********
        !                                                    !
        !                                                    !
        !              _ partitions of tablespace            !
        !              _ tables     in tablespace            !
        !                                                    !
        !                                                    !
        !                                                    !
        !              Mark  _ to print output               !
        !                                                    !
        +----------------------------------------------------+



 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit        Rfind       -     +         <     >    Canc
```

A list of all main commands available on the tablespace listing screen can be invoked as a window by entering the help character, that is, a question mark (?), in the command line of the screen.

List Plans

≫ **To invoke the List Plans function**

■   On the **Retrieval of System Tables** screen, enter function code P.

The name of the plan(s) to be listed must be specified.

▪ If a value followed by an asterisk is specified, all plans defined in the DB2 catalog whose names begin with this value are listed.

▪ If asterisk notation is specified only, all plans defined in the DB2 catalog are listed.

Press Enter.

```
  16:37:59                 ***** NATURAL TOOLS FOR DB2 *****              2007-10-05
   PLAN *                                   S 01     Row 0 of 80 Columns 023 075
   ====>                                                    Scroll ===>  PAGE
      PLAN    CREATOR    VALIDATE ISO ACQUIRE REL VALID OPER EXPLAIN   PLSIZE
  ** ***************************** top of data *******************************
   __ CAFPLAN  SAG3        R        S   U      C   Y     Y     N         2472
   __ SAGEDCL  SAG1        R        S   U      C   Y     Y     N         1992
   __ SAGESPCS SAG1        R        S   U      C   Y     Y     N         1992
   __ SAGESPRR SAG1        R        R   U      C   Y     Y     N         1992
   __ SAGTIA22 SAG1        R        S   U      C   Y     Y     N         1992
   __ SAG8BH22 SAG1        R        S   U      C   Y     Y     N         2296
   __ SAG8CC22 SAG1        R        S   U      C   Y     Y     N         4376
   __ SAG8IC22 SAG1        R        S   U      C   Y     Y     N         4264
   __ SAG8SC22 SAG1        R        S   U      C   Y     Y     N         2296
   __ SAGPLA   SAG         R        S   U      C   Y     Y     N         2648
   __ TREPH01  SAG4        B        S   U      C   A     Y     N         2168
   __ TREPLANC SAG2        R        S   U      C   N     Y     N         4560
   __ TREPLANG SAG2        R        S   U      C   N     Y     N         8976
   __ TREPLANO SAG2        R        S   U      C   N     Y     N         8976
   __ TREPLANT SAG2        R        S   U      C   Y     Y     N         2472
   __ TREPLAN1 SAG2        R        S   U      C   N     Y     N         3248

  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help        Exit        Rfind         -     +           <     >     Canc
```

Commands Allowed on Plans

The following line commands are available on the plan listing screen. Line commands are entered in front of the desired plan(s):

Command	Description
I	Displays information on a plan.
S	Selects a plan to be used with main commands.
U	Unselects a plan.
AU	Displays information on access rights to a plan.
DR	Displays all DBRMs contained in a plan.
IX	Displays all indexes used by a plan.
PK	Displays the package list of a plan.
SY	Displays the systems enabled or disabled for a plan.
TB	Displays tables used in a plan.

The listing displayed as a result of the DR, IX, PK, or TB command can be used for further processing, whereas the contents of the screens displayed as a result of the I, AU, or SY command are for information purposes only.

A list of all line commands available with the **List Plans** function can be invoked as a window by entering the help character "?" in front of any of the listed plans.

The commands AU, DR, IX, PK, SY, and TB can also be used as main commands, which are entered on the command line of the plan listing screen and apply to all plans previously selected with the line command S.

The INFO main command, which is the equivalent of the I line command, displays information on the DBRMs and their SQL statements contained in the plans previously selected. As with the **List Database** function, information resulting from the I or INFO commands can be printed, too.

```
 16:37:59               ***** NATURAL TOOLS FOR DB2 *****              2007-10-05
  PLAN *                                    S 01      Row 0 of 80 Columns 023 075
  ====>                                                      Scroll ===>  PAGE
     PLAN     CREATOR    VALIDATE ISO ACQUIRE REL VALID OPER EXPLAIN   PLSIZE
  ** **** +---------------------------------------------------------------+ **********
  I_ CAFP !                                                              !    2472
  __ SAGE !                 Select what to display                       !    1992
  __ SAGE !                                                              !    1992
  __ SAGE !           _ DBRMs of plan                                    !    1992
  __ SAGT !           _ package list of plan                            !    1992
  __ SAG8 !           _ systems enabled or disabled for plan             !    2296
  __ SAG8 !           _ tables  referenced in plan                      !    4376
  __ SAG8 !           _ indexes used in plan                            !    4264
  __ SAG8 !           _ authorizations for plan                         !    2296
  __ SAGP !                                                              !    2648
  __ TREP !             Mark  _ to print output                         !    2168
  __ TREP !                                                              !    4560
  __ TREP +---------------------------------------------------------------+    8976
  __ TREPLAN0 SAG2       R       S   U       C    N     Y    N               8976
  __ TREPLANT SAG2       R       S   U       C    Y     Y    N               2472
  __ TREPLAN1 SAG2       R       S   U       C    N     Y    N               3248


 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit        Rfind           -     +          <     >    Canc
```

A list of all main commands available with the **List Plans** function can be invoked as a window by entering the help character, that is, a question mark (?), in the command line of the plan list screen.

DBRMs of Plan

If you issue the `DR` command on the plan listing screen, a list of all DBRMs bound into the selected plan(s) is displayed.

```
 16:40:56                  ***** NATURAL TOOLS FOR DB2 *****                2007-10-05
  DBRMS OF PLAN SAGTEST                          S 02      Row 0 of 3 Columns 033 075
  ====>                                                      Scroll ===>  PAGE
    PLAN     DBRM     TIMESTAMP    CREATOR  TIME     DATE     PDS NAME QUOTE CO
 ** ***************************** top of data *****************************
 __ SAGTEST  TEST1    148C251A1>    SAG      16:24:10 07-10-05 DB2.V42.>N     N
 __ SAGTEST  TEST2    148C251A1>    SAG      16:24:42 07-10-05 DB2.V42.>N     N
 __ SAGTEST  TEST3    148C251A1>    SAG      16:25:15 07-10-05 DB2.V42.>N     N
 ** *************************** bottom of data ***************************
```

Commands Allowed on DBRMs

The following line commands are available on the DBRM listing screen. Line commands are entered in front of the desired DBRM(s):

Command	Description
I	Displays information on a DBRM.
S	Selects a DBRM to be used with main commands.
U	Unselects a DBRM.

A list of all line commands available on the DBRM listing screen can be invoked as a window by entering the help character, that is, a question mark (?), in front of any of the listed DBRMs.

The only main command that applies to DBRMs is the `INFO` command, which is the equivalent of the `I` line command, but displays information on all previously selected DBRMs. Instead of being displayed, all information resulting from the `I` or `INFO` commands can also be marked for printing. Even if already displayed, information can be printed by issuing the `PRINT` command.

```
 16:40:56              ***** NATURAL TOOLS FOR DB2 *****           2007-10-05
  DBRMS OF PLAN SAGTEST                         S 02      Row 0 of 3 Columns 033 075
  ====>                                                   Scroll ===>  PAGE
     PLAN     DBRM     TIMESTAMP    CREATOR  TIME      DATE     PDS NAME QUOTE CO
 ** **** +-------------------------------------------------------------+***********
 I_ SAGT !                                                             ! .>N     N
 __ SAGT !               Select what to display                       ! .>N     N
 __ SAGT !                                                             ! .>N     N
 ** **** !                                                             !***********
        !                                                             !
        !                 _ Plans referencing DBRM                    !
        !                 _ SQL statements of DBRM                     !
        !                                                             !
        !                                                             !
        !                                                             !
        !                 Mark  _ to print output                     !
        !                                                             !
        +-------------------------------------------------------------+


 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit        Rfind       -     +          <     >     Canc
```

Indexes Used in Plan

If you issue the IX command on either the **plan listing screen** or the **table listing screen**, a list of all indexes used in the selected plan(s) or table(s) is displayed.

```
 16:40:56              ***** NATURAL TOOLS FOR DB2 *****           2007-10-05
  INDEXES OF PLAN SAGTEST                       S 02      Row 0 of 3 Columns 033 075
  ====>                                                   Scroll ===>  PAGE
     CREATOR  INDEX NAME    CREATOR  TABLE NAME COLCNT UNIQ CLSTRNG CLSTRD -RATI
 ** ***************************** top of data ******************************
 __ SAGCRE   XDEPT1        SAGCRE   DEPT           1 P   N       Y        10
 __ SAGCRE   XEMP1         SAGCRE   EMP            1 P   Y       Y        10
 __ SAGCRE   XEMP2         SAGCRE   EMP            1 D   N       N         4
 ** ***************************** bottom of data ***************************
```

Commands Allowed on Indexes

The following line commands are available on the index listing screen. Line commands are entered in front of the desired index(es):

Command	Description
I	Displays information on an index.
S	Selects an index to be used with main commands.
U	Unselects an index.
CO	Displays all columns of an index.
PT	Displays the partitions of an index.

The listings of columns displayed as a result of the CO or PT command cannot be used for further processing. Like the display resulting from the I command, they are for information purposes only.

A list of all line commands available on the index listing screen can be invoked as a window by entering the help character "?" in front of any of the listed indexes.

The commands CO and PT can be used as main commands, too, and entered in the command line of the index listing screen. If so, all columns of all indexes previously selected with the line command S are displayed.

A further main command is the INFO command, which is the equivalent of the line command I, but displays information on all previously selected indexes. Instead of being displayed, all information resulting from the I or INFO commands can also be marked for printing. Even if already displayed, information can be printed by issuing the PRINT command.

```
 16:40:56              ***** NATURAL TOOLS FOR DB2 *****              2007-10-05
  INDEXES OF PLAN SAGTEST                      S 02      Row 0 of 3 Columns 033 075
  ====>                                                    Scroll ===>  PAGE
     CREATOR  INDEX NAME    CREATOR  TABLE NAME COLCNT UNIQ CLSTRNG CLSTRD -RATI
 ** **** +---------------------------------------------------------------+ **********
  I_ SAGC !                                                              !        10
  __ SAGC !                   Select what to display                     !        10
  __ SAGC !                                                              !         4
 ** **** !                                                               ! **********
         !                   _ columns  of index                        !
         !                   _ portions of index                        !
         !                   _ plans using index                        !
         !                   _ packages using index                     !
         !                                                               !
         !                                                               !
         !                   Mark  _ to print output                    !
         !                                                               !
         +---------------------------------------------------------------+


  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help        Exit        Rfind         -     +          <     >     Canc
```

A list of all main commands available on the index listing screen can be invoked as a window by entering the help character "?" in the command line of the screen.

Package List of Plan

If you issue the PK command on the plan listing screen, a list of all entries in the package list of the selected plan(s) is displayed.

```
 16:40:56                ***** NATURAL TOOLS FOR DB2 *****              2007-10-05
  PACKAGE LIST FOR PLAN SAGTEST                 S 02      Row 0 of 3 Columns 033 075
  ====>                                                     Scroll ===>  PAGE
     PLANNAME LOCATION COLLID      NAME     SEQNO   TIMESTAMP IBM
 ** ********************** top of data ***********************
  __ SAGTEST            SAGCOLLE>   *           1 2007-10-0>N
  __ SAGTEST            SAG_STAT>   *           2 2007-10-0>N
 ** *************************** bottom of data ***************
```

Commands Allowed on Package List Entries

The following line commands are available on the package list screen. Line commands are entered in front of the desired package list entry:

Command	Description
I	Displays information on a package list entry.
S	Selects a package list entry to be used with main commands.
U	Unselects a package list entry.
PK	Displays all packages of a package list entry.

The **listing of packages** as a result of the PK command can be used for further processing, whereas the display resulting from the I command is for information purposes only.

A list of all line commands available with a package list can be invoked as a window by entering the help character, that is, a question mark (?), in front of any of the listed entries.

The command PK can also be used as main command, which is entered in the command line of the above screen and applies to all package list entries previously selected with the line command S.

List Packages

≫ **To invoke the List Packages function**

■ On the **Retrieval of System Tables** screen, enter function code K.

The collection and name of the package(s) to be listed can be specified.

If a value followed by an asterisk is specified, all packages defined in the DB2 catalog whose collections/names begin with this value are listed.

If asterisk notation is specified only, all packages defined in the DB2 catalog are listed.

Press Enter.

```
 11:06:11              ***** NATURAL TOOLS FOR DB2 *****            2007-10-05
  PACKAGE *.*                              S 01    Row 34 of 65 Columns 041 075
  ====>                                                   Scroll ===>  PAGE
     COLLID    NAME       CONTOKEN CONTOKEN (HEX) OWNER     CREATOR  QUALIFIER
  __ SAGQCATV SAGQVPLN    ?  l?F   148C409316C673>SAG        SAG      SAG
  __ SAGQCATV SAGQVPPA    ?k  ? ?? 149270680F77E0>SAG        SAG      SAG
  __ SAGQCATV SAGQVRAS    ?   ??=? 148C409B09097E>SAG        SAG      SAG
  __ SAGQCATV SAGQVREL    ?   ??y0 148C409C06DFA8>SAG        SAG      SAG
  __ SAGQCATV SAGQVREV    ? ? ?v?  148CDFAD16A51F>SAG        SAG      SAG
  __ SAGQCATV SAGQVRIL    ?   s ?B 148C40A20329C2>SAG        SAG      SAG
  __ SAGQCATV SAGQVROO    ? ?  A y 148CDFAF03C18E>SAG        SAG      SAG
  __ SAGQCATV SAGQVSCA    ?   u??S 148C40A409DEE2>SAG        SAG      SAG
  __ SAGQCATV SAGQVSQL    ?    ??? 148C40AB001D3F>SAG        SAG      SAG
  __ SAGQCATV SAGQVSTM    ?   ? 7q 148C40AD078CF7>SAG        SAG      SAG
  __ SAGQCATV SAGQVSTO    ?   ? ?  148C40B409681E>SAG        SAG      SAG
  __ SAGQCATV SAGQVTAB    ?   ? +U 148C40B61F024E>SAG        SAG      SAG
  __ SAGQCATV SAGQVTAS    ?   ?  d 148C40B80874FF>SAG        SAG      SAG
  __ SAGQCATV SAGQVTBA    ?   ?  ? 148C40BB1854EC>SAG        SAG      SAG
  __ SAGQCATV SAGQVTBC    ?   ?d ? 148C40BD1684EC>SAG        SAG      SAG
  __ SAGQCATV SAGQVTBP    ?   ?    148C40BF07AE9D>SAG        SAG      SAG
  __ SAGQCATV SAGQVTBS    ?      ?? 148C40CA034928>SAG       SAG      SAG

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind        -     +          <     >     Canc
```

Commands Allowed on Packages

The following line commands are available on the package listing screen. Line commands are entered in front of the desired package(s):

Command	Description
I	Displays information on a package.
S	Selects a package to be used with main commands.
U	Unselects a package.
AU	Displays information on access rights to a package.
IX	Displays all indexes used by a package.
SY	Displays all systems enabled or disabled for a package.
TB	Displays all tables used by a package.

The listings of **indexes** or **tables** displayed as a result of the IX or TB command can be used for further processing, whereas the displays resulting from the AU, SY, or I command are for information purposes only.

A list of all line commands available with the **List Packages** function can be invoked as a window by entering the help character, that is, a question mark (?), in front of any of the listed packages.

The commands AU, IX, SY, and TB can also be used as main commands, which are entered in the command line of the table listing screen and apply to all tables previously selected with the line command S.

The INFO main command, which is the equivalent of the I line command, displays information on all tables previously selected. All information resulting from the I or INFO commands can also be printed.

```
 11:06:11                  ***** NATURAL TOOLS FOR DB2 *****              2007-10-05
  PACKAGE *.*                                    S 01    Row 34 of 65 Columns 041 075
  ====>                                                          Scroll ===>  PAGE
     COLLID    NAME        CONTOKEN CONTOKEN (HEX) OWNER    CREATOR  QUALIFIER
  i_ SAGQ +-----------------------------------------------------------+ G
  __ SAGQ !                                                           ! G
  __ SAGQ !                   Select what to display                  ! G
  __ SAGQ !                                                           ! G
  __ SAGQ !           _ systems enabled or disabled for package       ! G
  __ SAGQ !           _ tables   referenced in package                ! G
  __ SAGQ !           _ indexes used in package                       ! G
  __ SAGQ !           _ statements of package                         ! G
  __ SAGQ !           _ authorizations on package                     ! G
  __ SAGQ !                                                           ! G
  __ SAGQ !                                                           ! G
  __ SAGQ !                 Mark  _ to print output                   ! G
  __ SAGQ !                                                           ! G
  __ SAGQ +-----------------------------------------------------------+ G
  __ SAGQCATV SAGQVTBC   ?    ?d ? 148C40BD1684EC>SAG       SAG      SAG
  __ SAGQCATV SAGQVTBP   ?    ?    148C40BF07AE9D>SAG       SAG      SAG
  __ SAGQCATV SAGQVTBS   ?     ?? 148C40CA034928>SAG        SAG      SAG

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind         -     +            <     >     Canc
```

A list of all main commands available with the **List Packages** function can be invoked as a window by entering the help character, that is, a question mark (?), in the command line of the packages list screen.

List Tables

≫ **To invoke the List Tables function**

■ On the **Retrieval of System Tables** screen, enter function code T.

The creator and name of the table(s) to be listed can be specified.

▪ If a value followed by an asterisk is specified, all tables defined in the DB2 catalog whose creator/name begins with this value are listed.

▪ If asterisk notation is specified only, all tables defined in the DB2 catalog are listed.

Press Enter.

```
 16:42:58              ***** NATURAL TOOLS FOR DB2 *****          2007-10-05
 TABLE SAG*.*                              S 01   Row 34 of 361 Columns 036 075
 ====>                                                    Scroll ===>  PAGE
    CREATOR  TABLE NAME   TYPE COLCOUNT KEYCOLS RECLEN DATABASE TSNAME    ↵
C
 ** ***************************** top of data *****************************
 __ SAGCRE   ACT          T        3       1      38 SAG8D22A ACT
 __ SAGCRE   DEPT         T        4       1      59 SAG8D22A SAG8S2
 __ SAGCRE   EACT         T        5       0      54 SAG8D22A SAG8S2
 __ SAGCRE   EDEPT        T        6       0      75 SAG8D22A SAG8S2
 __ SAGCRE   EEMP         T       16       0     123 SAG8D22A SAG8S2
 __ SAGCRE   EEPA         T        8       0      52 SAG8D22A SAG8S2
 __ SAGCRE   EMP          T       14       1     107 SAG8D22A SAG8S2
 __ SAGCRE   EMPPROJACT   T        6       0      36 SAG8D22A EMPPRO
 __ SAGCRE   EPROJ        T       10       0      86 SAG8D22A SAG8S2
 __ SAGCRE   EPROJACT     T        7       0      45 SAG8D22A SAG8S2
 __ SAGCRE   PROJ         T        8       1      70 SAG8D22A PROJ
 __ SAGCRE   PROJACT      T        5       3      29 SAG8D22A PROJAC
 __ SAGCRE   TCONA        T        5       0    4056 SAG8D22P SAG8S2
 __ SAGCRE   TDSPTXT      T        3       0      91 SAG8D22P SAG8S2
 __ SAGCRE   TOPTVAL      T       11       0     354 SAG8D22P SAG8S2
 __ SAGCRE   VACT         V        3       0       0 SAG8D22A ACT

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind         -     +          <     >     Canc
```

Commands Allowed on Tables

The following line commands are available on the table listing screen. Line commands are entered in front of the desired table(s):

Command	Description
I	Displays information on a table.
S	Selects a table to be used with main commands.
U	Unselects a table.
AU	Displays information on access rights to a table.
CO	Displays all columns of a table.
IX	Displays all indexes on a table.
CC	Checks constraints.

The **listings of indexes** displayed as a result of the IX command can be used for further processing, whereas the listings of columns resulting from the CO command, as well as the displays resulting from the AU or I command, are for information purposes only.

A list of all line commands available with the **List Tables** function can be invoked as a window by entering the help character, that is, a question mark (?), in front of any of the listed tables.

The commands `AU`, `CO`, and `IX` can also be used as main commands, which are entered in the command line of the table listing screen and apply to all tables previously selected with the line command `S`.

The `INFO` main command, which is the equivalent of the `I` line command, displays information on all tables previously selected. All information resulting from the `I` or `INFO` commands can also be printed.

```
 16:42:58                 ***** NATURAL TOOLS FOR DB2 *****              2007-10-05
  TABLE SAG*.*                                  S 01   Row 34 of 361 Columns 036 075
  ====>                                                       Scroll ===>  PAGE
     CREA +---------------------------------------------------------+        C
 ** **** !                                                         ! **********
 I_ SAGC !                                                         !
 __ SAGC !                     Select what to display              ! S2
 __ SAGC !                                                         ! S2
 __ SAGC !  _ columns  of table/view    _ referential constraints  ! S2
 __ SAGC !  _ synonyms of table/view    _ authorized  users        ! S2
 __ SAGC !  _ plans    using table/view                            ! S2
 __ SAGC !  _ packages using table/view _ indexes  of table        ! S2
 __ SAGC !  _ views    using table/view _ columns  of indexes      ! RO
 __ SAGC !  _ base tables of view       _ plans using indexes      ! S2
 __ SAGC !  _ definition  of view       _ packages using indexes   ! S2
 __ SAGC !  _ check conditions of table                            !
 __ SAGC !                                                         ! AC
 __ SAGCR!              Mark  _ to print output                    ! S2
 __ SAGCR+---------------------------------------------------------+ S2
 __ SAGCRE   TOPTVAL     T           11        0    354 SAG8D22P SAG8S2
 __ SAGCRE   VACT        V            3        0      0 SAG8D22A ACT

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind         -      +         <     >    Canc
```

A list of all main commands available with the **List Tables** function can be invoked as a window by entering the help character, that is, a question mark (?), in the command line of the table listing screen.

User Authorizations

≫ **To invoke the User Authorization function**

■ On the **Retrieval of System Tables** screen, enter function code `U` and press Enter.

The **Retrieval of User Authorizations** menu is displayed:

```
 16:44:51                 ***** NATURAL TOOLS FOR DB2 *****               2007-10-05
                          - Retrieval of User Authorizations -



                    Code Function                    Parameter

                     C   Column   Authorizations Grantee
                     D   Database Authorizations Grantee
                     K   Package  Authorizations Grantee
                     P   Plan     Authorizations Grantee
                     R   Resource Authorizations Grantee
                     T   Table    Authorizations Grantee
                     U   User     Authorizations Grantee
                     ?   Help
                     .   Exit

                Code .. _    Grantee .. _____


 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help        Exit                                                   Canc
```

The following functions are available:

Code	Description
C	Displays the columns which can be accessed by the specified grantee.
D	Displays the databases which can be accessed by the specified grantee.
K	Displays the packages which can be accessed by the specified grantee.
P	Displays the plans which can be accessed by the specified grantee.
R	Displays the resources which can be accessed by the specified grantee.
T	Displays the tables which can be accessed by the specified grantee.
U	Displays the system privileges of the specified grantee.

The following parameter must be specified:

Parameter	Description
Grantee	A list of all existing DB2 objects of the specified object type to which the specified grantee has access is displayed.

List Statistic Tables

≫ **To invoke the List Statistic Tables function**

■   On the **Retrieval of System Tables** screen, enter function code S and press Enter.

The **Retrieval of Statistic Tables** menu is displayed:

```
 16:38:47               ***** NATURAL TOOLS FOR DB2 *****            2007-10-05
                         - Retrieval of Statistic Tables -


                       Code Function              Parameter

                        C   List SYSCOLSTATS      Creator, Name
                        D   List SYSCOLDISTSTATS  Creator, Name
                        I   List SYSINDEXSTATS    Index Owner, Name
                        T   List SYSTABSTATS      Creator, Name
                        ?   Help
                        .   Exit

               Code .. _   Index Owner ......... _____
                           Index Name .......... _____
                           Table Creator ....... _____
                           Table Name .......... _____



  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12--
        Help        Exit                                                   Canc
```

The following functions are available:

Code	Description
C	Displays the partitioned statistics for columns in a partitioned table space.
D	Displays the distribution of the values of the first column of a partitioned index.
I	Displays the statistics for a partitioned index.
T	Displays the statistics for a partitioned table space.

The following parameters must be specified:

Parameter	Description
Table Creator	The name of the creator of the table for which the statistics are to be displayed.
Table Name	The name of the table for which the statistics are to be displayed.
Index Owner	The name of the owner of the index for which the index statistics are to be displayed.
Index Name	The name of the index for which the index statistics are to be displayed.

9 Environment Setting

The **Environment Setting** facility of the **Natural Tools for DB2** allows you to issue special SQL statements interactively.

For details on the SQL statements described in this section, see the relevant DB2 literature by IBM.

Invoking the Environment Setting Facility

≫ **To invoke the Environment Setting facility**

■ On the **Natural Tools for DB2 Main Menu**, enter function code S and press Enter.

The **Environment Setting** screen is displayed.

```
 15:01:49                  ***** NATURAL TOOLS FOR DB2 *****              2009-10-07
                              - Environment Setting -


        Code Function                       Code Function SET CURRENT

         CO  CONNECT                         SS  SQLID
         RE  RELEASE (connection)            SP  PACKAGESET
         SC  SET CONNECTION                  SD  DEGREE
         SY  SET ENCRYPTION PASSWORD         SU  RULES
         SR  Display SPECIAL REGISTER        SO  OPTIMIZATION HINT
         ?   Help                            SL  LOCALE LC_CTYPE
         .   Exit                            SA  PATH
                                             SE  PRECISION
                                             SM  MAINTAINED TABLE TYPES FOR OPT
                                             SB  PACKAGE PATH
   Code .. __                                SF  REFRESH AGE
                                             SH  SCHEMA
                                             SN  APPLICATION ENCODING SCHEME

 Command ===>
```

This screen offers you the following functions:

CO	Specifies and executes the SQL statement CONNECT.
RE	Specifies and executes the SQL statement RELEASE.
SC	Specifies and executes the SQL statement SET CONNECTION.
SS	Specifies and executes the SQL statement SET CURRENT SQLID.
SP	Specifies and executes the SQL statement SET CURRENT PACKAGESET.
SD	Specifies and executes the SQL statement SET CURRENT DEGREE.
SU	Specifies and executes the SQL statement SET CURRENT RULES.

SO	Specifies and executes the SQL statement `SET CURRENT OPTIMIZATION HINT`.
SL	Specifies and executes the SQL statement `SET CURRENT LOCALE LC_CTYPE`.
SA	Specifies and executes the SQL statement `SET CURRENT PATH`.
SE	Specifies and executes the SQL statement `SET CURRENT PRECISION`.
SM	Specifies and executes the SQL statement `SET CURRENT MAINTAINED TABLE TYPE FOR OPTIMIZATION`.
SB	Specifies and executes the SQL statement `SET CURRENT PACKAGE PATH`.
SF	Specifies and executes the SQL statement `SET CURRENT REFRESH AGE`.
SH	Specifies and executes the SQL statement `SET CURRENT SCHEMA`.
SN	Specifies and executes the SQL statement `SET CURRENT APPLICATION ENCODING SCHEME`.
SY	Specifies and executes the SQL statement `SET ENCRYPTION PASSWORD`.
SR	Displays the current values of the supported special registers.

Connect

≫ **To invoke the Connect function**

■ On the **Environment Setting** screen, enter function code `CO` and press Enter.

The **Connect** screen is displayed:

```
 14:23:29                    ***** NATURAL TOOLS FOR DB2 *****               2006-04-13
                                      - Connect -




   >>---- CONNECT ---+-- _ --------------------------------+------------------->< 
                     !                                     !
                     !                                     !
                     +-- _ --- TO ---- _____  --+
                     !                  (location name)   !
                     !                                     !
                     +-- _ --- RESET ---------------------+


     Current Server Version _____



  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec                                               Canc
```

The **Connect** function connects the current application to a designated server. This server is the current server, which is displayed in the **Current Server Version** field.

On the **Connect** screen, you identify the current server by specifying a location name. The identified server must be known to the local DB2 subsystem.

Release

≫ **To invoke the Release function**

■   On the **Environment Setting** screen, enter function code RE and press Enter.

The **Release** screen is displayed:

```
 14:24:29                   ***** NATURAL TOOLS FOR DB2 *****              2006-04-13
                                  - Release -




  >>--- RELEASE ------+-------- _____ ------+-------------------->><
                      !            location-name        !
                      !                                  !
                      +-- _ --- CURRENT --------------+
                      !                                  !
                      !-- _ --- ALL SQL --------------!
                      !                                  !
                      +-- _ --- ALL PRIVATE ----------+




  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec                                             Canc
```

The **Release** function places one or more connections in the release pending state.

Set Connection

〉 **To invoke the Set Connection function**

■   On the **Environment Setting** screen, enter function code SC and press Enter.

The **Set Connection** screen is displayed:

```
 14:23:47                    ***** NATURAL TOOLS FOR DB2 *****               2006-04-13
                                  - Set Connection -




   >>--- SET CONNECTION -------- _____ --------------------------->><
                                     location-name






 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec                                             Canc
```

On the **Set Connection** screen, you identify a server by specifying a location name. The identified server must be known to the local DB2 subsystem.

Set Current SQLID

≫ **To invoke the Set Current SQLID function**

■ On the **Environment Setting** screen, enter function code SS and press Enter.

The **Set Current SQLID** screen is displayed:

```
 14:23:47                  ***** NATURAL TOOLS FOR DB2 *****            2006-04-13
                              - Set Current SQLID -




        >>--- SET CURRENT SQLID = ----- _____ -------------------->< 
                                       ( USER,
                                         string-constant)







 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec  Free                                       Canc
```

The **Set Current SQLID** function changes the value of the SQL authorization identifier. With SQL statements that use unqualified table names, DB2 uses the SQLID as an implicit table qualifier. This enables you to access identical tables with the same table name but with different creator names.

On the **Set Current SQLID** screen, you can replace the value of CURRENT SQLID by the value of the special register USER or by a string constant. The string constant can be up to 8 characters long.

In all supported TP-monitor environments, the SQLID can then be kept across terminal I/Os until its resetting or the end of the session.

Set Current Packageset

≫ **To invoke the Set Current Packageset function**

■   On the **Environment Setting** screen, enter function code SP and press Enter.

The **Set Current Packageset** screen is displayed:

```
09:39:07                    ***** NATURAL TOOLS FOR DB2 *****              2006-04-18

                              - Set Current Packageset -



>>--- SET CURRENT PACKAGESET = ----------------------------------------------->


 >-+-- _ - USER ------------------------------------------------------+->＜
   !                                                                  !
   +-- _____   !
                          (string-constant)                          !

         _____  -+
                        (string-constant cont.)

Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec                                            Canc
```

The `SET CURRENT PACKAGESET` statement assigns a value to the special register `CURRENT PACKAGESET`.

On the **Set Current Packageset** screen, you can replace the value of `CURRENT PACKAGESET` by the value of the special register `USER` or by a string constant of up to 18 characters.

Set Current Degree

≫ **To invoke the Set Current Degree function**

■ On the **Environment Setting** screen, enter function code `SD` and press Enter.

The **Set Current Degree** screen is displayed:

```
 14:23:58                  ***** NATURAL TOOLS FOR DB2 *****              2006-04-13
                               - Set Current Degree -




 >>--- SET CURRENT DEGREE ---------- ___ ------------------------------------->< 
                                    ( 1 or ANY )






 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec                                              Canc
```

CURRENT DEGREE specifies the degree of parallelism for the execution of queries that are dynamically prepared by the application process.

Set Current Rules

≫ **To invoke the Set Current Rules function**

■   On the **Environment Setting** screen, enter function code SU and press Enter.

The **Set Current Rules** screen is displayed:

```
 14:23:58                   ***** NATURAL TOOLS FOR DB2 *****              2006-04-13
                                  - Set Current Rules -




    >>--- SET CURRENT RULES ----------- ___ ------------------------------------->< 
                                     ( DB2 or STD )







 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec                                            Canc
```

CURRENT RULES specifies whether certain SQL statements are executed in accordance with DB2 rules or the rules of the SQL standard.

Set Current Optimization Hint

≫ **To invoke the Set Current Optimization Hint function**

■ On the **Environment Setting** screen, enter function code S0 and press Enter.

The **Set Current Optimization Hint** screen is displayed:

```
  09:41:43                  ***** NATURAL TOOLS FOR DB2 *****              2006-04-18

                           - Set Current Optimization Hint -



>>--- SET CURRENT OPTIMIZATION HINT -------------------------------------->


 >--- _____
                              (string-constant)

     _____  --->><
                              (string-constant cont.)

Command ===>

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec                                              Canc
```

CURRENT OPTIMIZATION HINT specifies the user-defined optimization hint that DB2 should use to generate the access path for dynamic statements.

Set Current Locale LC_CType

≫ **To invoke the Set Current Locale LC_CType function**

■ On the **Environment Setting** screen, enter function code SL and press Enter.

The **Set Current Locale LC_CType** screen is displayed:

```
 14:58:12                   ***** NATURAL TOOLS FOR DB2 *****              2006-04-13
                            - Set Current Locale LC_CType -




>>--- SET CURRENT LOCALE LC_CTYPE ------------------------------------------->

 >--------- _____ ---------->< 
                              (string-constant)




 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec                                             Canc
```

CURRENT LOCALE LC_CTYPE specifies the LC_CTYPE locale that will be used to execute SQL statements that use a built-in function that references a locale.

Set Current Path

≫ **To invoke the et Current Path function**

■   On the **Environment Setting** screen, enter function code SA and press Enter.

The **Set Current Path** screen is displayed:

```
09:42:09                    ***** NATURAL TOOLS FOR DB2 *****              2006-04-18

                                   - Set Current Path -




>>- SET CURRENT PATH ---------------------------------------------------------->


     +----------------------------<--( , )----------------------------+

     !                                                                !
>-++------------------------------ _ ------------------------------++->< 

     !                     (schema-name<,schema-name,...>)            !

     !                                                                !
     +- _ -------------------- SYSTEM PATH --------------------------+

     !                                                                !
     +- _ ----------------------- USER -----------------------------+

     !                                                                !
     +- _ -------------------- CURRENT PATH -------------------------+

     !                                                                !
     +- _ --------------- CURRENT PACKAGE PATH ----------------------+

Command===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
     Help  Error Exit  Exec                                             Canc
```

CURRENT PATH specifies the SQL path used to resolve unqualified data type names and function names in dynamically prepared SQL statements.

Set Current Precision

≫ **To invoke the Set Current Precision function**

■ On the **Environment Setting** screen, enter function code `SE` and press Enter.

The **Set Current Precision** screen is displayed:

```
 15:01:17                    ***** NATURAL TOOLS FOR DB2 *****              2006-04-13
                                  - Set Current Precision -




>>--- SET CURRENT PRECISION ------- DEC15 -------------------------------->< 
                            (DEC15,DEC31,15,31,
                        D15.1 - D15.9,D31.1 - D31.9)






   Command ===>
   Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec                                              Canc
```

`CURRENT PRECISION` specifies the rules to be used when both operands in a decimal operation have precisions of 15 or less.

Set Current Maintained Types for Optimization

≫ **To invoke the Set Current Maintained Types function**

■ On the **Environment Setting** screen, enter function code `SM` and press Enter.

The **Set Current Maintained Types** for Optimization screen is displayed:

```
09:36:51                    ***** NATURAL TOOLS FOR DB2 *****              2006-04-18

                           - Set Current Maintained Types -


>>--- SET CURRENT MAINTAINED TYPES --- SYSTEM ----------------------------->< 

                           ( ALL, NONE, SYSTEM or USER )







 Command ===>

  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec                                            Canc
```

CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION specifies a value that identifies the types of objects that can be considered to optimize the processing of dynamic SQL queries. This register contains a keyword representing table types.

Set Current Package Path

≫ **To invoke the Set Current Package Path function**

■   On the **Environment Setting** screen, enter function code SB and press Enter.

   The **Set Current Package Path** screen is displayed:

```
 09:37:22                    ***** NATURAL TOOLS FOR DB2 *****              2006-04-18
                             - Set Current Package Path -


 >> - SET CURRENT PACKAGE PATH ------------------------------------------------------>


     +-------------------------------< --( , )--------------------------------+
     !                                                                        !
  > -++----------------------------- _ ----------------------------------++->< 
     !                    (collection-id< ,collection-id,...> )             !
     !                                                                      !
     +- _ ----------------------- USER -----------------------------------+
     !                                                                    !
     +- _ -------------------- CURRENT PATH ------------------------------+
     !                                                                    !
     +- _ ----------------- CURRENT PACKAGE PATH -----------------------+


 Command ===>

  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec                                              Canc
```

CURRENT PACKAGE PATH specifies a value that identifies the path used to resolve references to
packages that are used to execute SQL statements.

Set Current Refresh Age

≫ **To invoke the Set Current Refresh Age function**

■ On the **Environment Setting** screen, enter function code SF and press Enter.

The **Set Current Refresh Age** screen is displayed:

```
09:37:40                    ***** NATURAL TOOLS FOR DB2 *****              2006-04-18

                              - Set Current Refresh Age -




>> --- SET CURRENT REFRESH AGE ----- _____  ---------------->< 

                          ( 0 or ANY/99999999999999.000000 )




Command ===>

  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help  Error Exit  Exec                                              Canc
```

CURRENT REFRESH AGE specifies a timestamp duration value with a data type of DECIMAL.

Set Current Schema

≫ **To invoke the Set Current Schema function**

■ On the **Environment Setting** screen, enter function code SH and press Enter.

The **Set Current Schema** screen is displayed:

```
09:38:01                    ***** NATURAL TOOLS FOR DB2 *****                2006-04-18
                                   - Set Current Schema -



>>- SET CURRENT SCHEMA --------------------------------------------------->


 >--+- _____ -+-->< 
    !                            (schema-name)                        !

    !                                                                 !

    +- _ ------------------------ USER ----------------------------+

    !                                                                 !

    +- _ ------------------------ DEFAULT -------------------------+

    !                                                                 !

    +- _____ -+
                            (string-constant)




Command ===>

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec                                           Canc
```

The CURRENT SCHEMA, or equivalently CURRENT_SCHEMA, special register specifies the schema name used to qualify unqualified database object references in dynamically prepared SQL statements.

Set Current Application Encoding Scheme

 ❯ **To invoke the Set Current Application Encoding Scheme function**

■   On the **Environment Setting** screen, enter function code SN and press Enter.

   The **Set Current Application Encoding Scheme** screen is displayed:

```
09:38:21                    ***** NATURAL TOOLS FOR DB2 *****                2006-04-18

                         - Set Current Application Encoding Scheme -




 >>--- SET CURRENT APPLICATION ENCODING SCHEME ----------------------------->


  >------------------------------- _____ ------------------------------->< 

                             ( ASCII, EBCDIC, UNICODE

                                or 1 - 65533)




 Command ===>

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec                                            Canc
```

CURRENT APPLICATION ENCODING SCHEME specifies which encoding scheme is to be used for
dynamic statements. It allows an application to indicate the encoding scheme that is used to
process data.

Set Encryption Password

❯ **To invoke the Set Encryption Password function**

■ On the **Environment Setting** screen, enter function code SY and press Enter.

The Set Encryption Password screen is displayed:

```
09:36:13                    ***** NATURAL TOOLS FOR DB2 *****              2006-04-18

                                - Set Encryption Password -




>>--- SET ENCRYPTION PASSWORD ------------------------------------------------->


 >---- _____
                              (password-string-constant)

      _____ ---->
                              (password-string-constant cont.)


 >-+-----------------------------------------------------------------------+->< 
   !                                                                       !
   +--- WITH HINT --- _____ ----------------+
                              (hint-string-constant)




Command ===>

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Exec                                            Canc
```

The **Set Encryption Password** function sets the value of the encryption password and, option-
ally, the password hint.

Display Special Registers

≫ **To invoke the Display Special Registers function**

■ On the **Environment Setting** screen, enter function code SR and press Enter.

The **Display Special Registers** screen is displayed:

```
15:18:07                ***** NATURAL TOOLS FOR DB2 *****              2006-04-13
                           - Display Special Registers -
 Current
  +Client_Acctng .........
  +Client_ApplName .......
  +Client_UserID .........
  +Client_WrkStnName .....
   Appl.Encoding Scheme .. EBCDIC
   Date .................. 13.04.2006
   Degree ............... 1
   LC_CType .............
  +Maintained Types ...... SYSTEM

   Member ............... DB28
  +Optimization Hint .....

  +Package Path .........




Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Updat                                      Next  Canc
```

When you press PF11, the next screen of Special Register values is displayed.

```
15:31:20                   ***** NATURAL TOOLS FOR DB2 *****               2006-04-13
                            - Display Special Registers -
 Current
  +PackageSet ............

  +Path ................. "SYSIBM","SYSFUN","SYSPROC","GGS"


   Precision ............. DEC15
   Refresh Age ...........
   Rules ................. DB2
  +Schema ................ GGS

   Server ................ DAEFDB28
   SQLID ................. GGS
   Time .................. 15.31.20
   TimeStamp ............. 2006-04-13-15.31.20.948481
   TimeZone .............. 10000
   User .................. GGS

Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Error Exit  Updat                                Prev        Canc
```

When you press PF10, the previous screen of Special Register values is displayed.

The **Display Special Registers** screens show you the current values of the Special Registers of DB2 supported by Natural for DB2.

Fields, which are prefixed with a plus sign (+), may contain more data than displayed on the screen. You can display the full contents either when you position the cursor on the field (description or data) and press ENTER, or when you enter the abreviation of the field (which are the capital letters of the description) prefoxed by the plus sign (+) in the command line. For example, +PS shows a window with the full value of the **Current Package Set**.

10   Explain PLAN_TABLE

⚠️ **Important:** Before you use the **Explain PLAN_TABLE** function, refer to *LISTSQL and Explain Functions* in the section *Special Requirements for Natural Tools for DB2* in *Installing Natural for DB2 on z/OS*.

The **Explain PLAN_TABLE** facility of the **Natural Tools for DB2** interprets the results of SQL `EXPLAIN` commands from your `PLAN_TABLE`. The information contained in your `PLAN_TABLE` is represented in so-called explanations.

Explanations of a `PLAN_TABLE` describe the access paths chosen by DB2 to execute SQL statements.

An SQL statement is executed by DB2 in one or more steps. For each execution step, one row is inserted into the `PLAN_TABLE`. All rows together describing the access path for one SQL statement are called an explanation.

The explanations are identified in the `PLAN_TABLE` by a combination of either plan name, DBRM (database request module) name, and query number or collection name, package name, and query number.

EXPLAIN Modes

DB2 provides three ways to explain SQL statements:

- Dynamic EXPLAIN
- Bind Plan EXPLAIN
- Bind Package EXPLAIN

Depending on the way the identifications of the explanations differ.

Dynamic EXPLAIN

Executes an SQL `EXPLAIN` command dynamically, where the explanation is inserted into the `PLAN_TABLE` of your current SQLID.

The `EXPLAIN` command can be issued within the **Catalog Maintenance** and **Interactive SQL** facilities of the **Natural Tools for DB2**. In addition, the Natural `LISTSQL` command can be used to extract SQL statements from cataloged Natural programs, and to issue the SQL `EXPLAIN` command for the extracted SQL statements.

If you issue the SQL `EXPLAIN` command dynamically, you should specify a query number to help identify the explanation in the `PLAN_TABLE`. The same query number should be used for related statements.

Depending on the method with which the DBRM used by the dynamic SQL processor is bound into the plan, DB2 uses two different ways to identify rows in the `PLAN_TABLE`:

- Dynamic Mode
- Package Mode

Dynamic Mode

The DBRM is bound directly into the plan.

When an explanation is inserted, the plan name, the DBRM name, and the query number are determined by DB2 as follows:

Parameter	Description
plan name	is left blank;
DBRM name	is the name of the DBRM used by the dynamic SQL processor;
query number	is equal to the query number you specified with the EXPLAIN command (the default query number is 1).

This explanation mode is called dynamic mode.

Package Mode

The DBRM is bound as package into the plan.

When an explanation is inserted, the collection name, the package name, and the query number are determined by DB2 as follows:

Parameter	Description
collection name	is the name of the collection that contains the package;
package name	is the name of the package used by the dynamic SQL processor;
query number	is equal to the query number you specified with the EXPLAIN command (the default query number is 1).

This explanation mode is called package mode.

Bind Plan EXPLAIN

Binds an application plan with the option **EXPLAIN YES**, where the explanation is inserted into the PLAN_TABLE of the owner of the plan. When an explanation is inserted, the plan name, the DBRM name, and the query number are determined by DB2 as follows:

Parameter	Description
`plan name`	is the name of the plan;
`DBRM name`	is the name of the DBRM that contains the SQL statement;
`query number`	is equal to the statement number (`stmtno`), which is generated by the DB2 precompiler.

Bind Package EXPLAIN

Binds a package with the option **EXPLAIN YES**, where the explanation is inserted into the `PLAN_TABLE` of the owner of the package. When an explanation is inserted, the collection name, the package name, and the query number are determined by DB2 as follows:

Parameter	Description
`collection name`	is the name of the collection that contains the package;
`package name`	is the name of the package that contains the SQL statement;
`query number`	is equal to the statement number (`stmtno`), which is generated by the DB2 precompiler.

Invoking the EXPLAIN_TABLE Function

Explanations can be selected by either plan name, DBRM name, and query number or collection name, package name, and query number. If you issue an `EXPLAIN` command various times, it is possible that multiple explanations are identified by a given combination of these selection fields. Thus, you can select either all explanations or only the most recent one. A list with all selected explanations is displayed, from which you can select individual rows for a more detailed description.

The individual rows of a `PLAN_TABLE` are displayed one row per line. Rows that describe the same SQL statement are shown together as one explanation. Different explanations, are separated by empty lines. You can browse through the list and select a detailed report for individual explanations. If rows have been inserted into your `PLAN_TABLE` as a result of a Natural system command `LISTSQL`, the names of the Natural library and program are also displayed.

≫ **To invoke the Explain PLAN_TABLE facility**

■ On the **Natural Tools for DB2 Main Menu**, enter function code `X`.

The **Explain PLAN_TABLE** screen is displayed:

```
16:45:35              ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                           - Explain PLAN_TABLE -



                   Code Function

                    L    List PLAN_TABLE - Latest  Explanations
                    A    List PLAN_TABLE - All     Explanations
                    D    Delete from PLAN_TABLE
                    ?    Help
                    .    Exit


             Code .. _    Mode ........ DYNAMIC_ ( Dynamic, Plan, Package )
                          Plan ........ _____
                          Collection .. _____
                          DBRM/Package  _____
                          Queryno ..... _____ - _____




  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Setup Exit                                                     Canc
```

With PF2 (Setup) the maximum length of one column and the number of fixed characters when
scrolling left may be specified. The default values for both parameters may be changed in the
CONFIG subprogram in library SYSDB2.

When a column value is longer than the maximum length, it will be truncated and marked
with a greater than symbol (>), which means that strings are truncated at the right end, or a
or a less than symbol (<), which means that numbers are truncated at the left end. Note, that
for further commands on a line, for example, the line command I, only the visible value can
be taken as input. This means that commands on lines will fail, when values for further pro-
cessing are truncated.

```
16:45:35                 ***** NATURAL TOOLS FOR DB2 *****            2009-10-30
                             - Explain PLAN_TABLE -


                  Code Function    +---------Explain PLAN_TABLE----------+
                                    !                                     !
                  L   List PLAN_T ! Maximum length of columns ... ___12  !
                  A   List PLAN_T ! Number of fixed characters .. ____0  !
                  D   Delete from !                                      !
                  ?   Help        !                                      !
                  .   Exit        +-------------------------------------+


              Code .. _    Mode ........ DYNAMIC_ ( Dynamic, Plan, Package )
                           Plan ........ _____
                           Collection .. _____
                           DBRM/Package  _____
                           Queryno ..... _____ - _____




  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help  Setup Exit                                                    Canc
```

The following functions are available:

Code	Description
L	The **List PLAN_TABLE - Latest Explanations** function lists the last explanation for any combination of the parameters described below.
A	The **List PLAN_TABLE - All Explanations** function lists all explanations for any combination of the parameters described below.
D	The **Delete from PLAN_TABLE** function deletes the specified explanations from your PLAN_TABLE.

The following parameters can be specified:

Parameter	Description
Mode	Specifies the explanation mode (Dynamic, Plan, or Package).
Plan *plan-name*	Specifies a valid plan name. The parameter Plan is only required in Plan mode.
Collection *collection-name*	Specifies a valid collection name. The parameter Collection is only required in Package mode.
DBRM/Package *dbrm/package-name*	In Plan mode, specifies a valid DBRM name. In Package mode, specifies a valid package name.

Parameter	Description
	In dynamic mode, specifies the DBRM used by the dynamic SQL processor.
	If a value followed by an asterisk (*) is specified, all DBRMs/packages of the specified plan/collection whose names start with the specified value are considered.
	If asterisk notation is specified only, all DBRMs/packages of the specified plan/collection are considered.
	The `DBRM/Package` parameter is used to limit the display to individual DBRMs/packages.
`Queryno no.1 - no.2`	This parameter specifies a valid range of query numbers, where the following rules apply: ■ If no query number is specified, all query numbers are displayed; ■ If only the first query number is specified, only this query number is displayed; ■ If only the second query number is specified, all query numbers up to and including the second query number are displayed; ■ If both query numbers are specified, all query numbers between and including the first and the second query number are displayed.

List PLAN_TABLE - Latest Explanations

This function only lists the most recent explanation for any specified combination of either plan name, DBRM name, and query number or package name, collection name and query number.

List PLAN_TABLE - All Explanations

This function lists all explanations for any combination of either plan name, DBRM name, and query number or package name, collection name and query number. The query number parameters are interpreted as above.

Sample Listing of Explanations

```
11:04:04             ***** NATURAL TOOLS FOR DB2 *****              2007-09-05
 Plan TESTPLAN                                  S 01    Row 0 of 152 Columns 032 075
 ====>                                                       Scroll ===>  PAGE
   DBRM         QNO    ME ACC    MA IO   PRE SORTN SORTC TCREATOR TABLENAME
 ** **************************** top of data ****************************
 __ TEST         722       I     1 -        ---- ----  SAGCRE   DEPT
 __ TEST         722    1  I     1 -        ---- ----  SAGCRE   EMP
 __ TEST         722    3        -          ---- --0-
 __ TEST         722       I     1 -        ---- ----  SAGCRE   DEPT
 __ TEST         722       I     1 Y        ---- ----  SAGCRE   EMP
 __ TEST         722       I     1 -        ---- ----  SAGCRE   DEPT
 __
 __ TEST         761       I     1 -        ---- ----  SAGCRE   EMP
 __ TEST         761    1  I     1 -        ---- ----  SAGCRE   DEPT
 __ TEST         761    3        -          ---- --0-
 __ TEST         761       I     1 -        ---- ----  SAGCRE   EMP
 __ TEST         761       I     1 Y        ---- ----  SAGCRE   DEPT
 __
 __ TEST         793       I     1 -        ---- ----  SAGCRE   DEPT
 __ TEST         793    1  I     1 -        ---- ----  SAGCRE   EMP
 __ TEST         793    1  I     1 -        ---- ----  SAGCRE   EMP


 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help       Exit       Rfind        -     +         <     >    Canc
```

Commands Available

The following line commands are available within listings of the **Explain PLAN_TABLE** facility. Line commands are entered in front of any of the rows of the desired explanation(s).

Command	Description
I	Displays a window where additional information about an explanation can be selected
S	Selects an explanation to be used with the INFO command described below.
U	Unselects an explanation for use with the INFO command.

A list of the line commands available can be invoked as a window by entering the help character, that is a question mark (?), in front of any of the listed rows.

Apart from the line commands, the INFO command can be specified, too. The INFO command must be entered in the command line of the listing screen and is the equivalent of the line command I. INFO displays a window where additional information can be selected on all explanations previously selected by the line command S.

In Plan mode, the following window is displayed, where you can select which additional information you want to be displayed or printed.

```
16:48:24                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
 Plan TESTPLAN                               S 01    Row 0 of 82 Columns 048 100
 ====>                                                     Scroll ===>  PAGE
    DBRM    QNO   ME ACC MA IO    PRE SORTN SORTC TCREATOR TABLENAME
 ** **** +-----------------------------------------------------------+***********
 __ TEST !                                                           !
 __ TEST !                     Select what to display                !
 __ TEST !                                                           !
 __ TEST !                      _ information about plan             !
 __ TEST !                      _ statements of plan                 !
 __ TEST !                      _ data from PLAN_TABLE               !
 __      !                      _ evaluation of PLAN_TABLE           !
 __ TEST !                      _ catalog statistics                 !
 __ TEST !                      _ columns of used indexes            !
 __ TEST !                                                           !
 __ TEST !                     Mark  _ to print output               !
 __ TEST !                                                           !
 __      +-----------------------------------------------------------+
 __ TEST    793      I   1  -            ---- ---- SAGCRE   DEPT
 __ TEST    793  1   I   1  -            ---- ---- SAGCRE   EMP
 __ TEST    793  1   I   1  -            ---- ---- SAGCRE   EMP

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind        -     +           <     >     Canc
```

Accordingly, the following window is displayed in Package mode:

```
16:48:24                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
 Package TESTPACK                            S 01    Row 0 of 82 Columns 048 100
 ====>                                                     Scroll ===>  PAGE
    DBRM +-----------------------------------------------------------+
 ** **** !                                                           ! **********
 __ TEST !                                                           ! ES
 __ TEST !                     Select what to display                ! ES
 __ TEST !                                                           ! ES
 __ TEST !                      _ information about package          ! ES
 __ TEST !                      _ statements of package              ! ES
 __ TEST !                      _ data from PLAN_TABLE               ! ES
 __      !                      _ evaluation of PLAN_TABLE           ! ES
 __ TEST !                      _ catalog statistics                 ! ES
 __ TEST !                      _ columns of used indexes            ! ES
 __ TEST !                                                           ! ES
 __ TEST !                     Mark  _ to print output               ! ES
 ** **** +-----------------------------------------------------------+ **********




 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit        Rfind        -     +           <     >     Canc
```

Browsing of data displayed is performed with browse commands, of which the most important can also be issued via PF keys; see *Editing within the Natural Tools for DB2*.

Option	Description
Information about plan/package	If a plan/package name has been specified, this option includes information from the DB2 catalog, such as date and time of the bind, as well as several bind options. In Dynamic mode, this option is not available.
Statements of plan/package	If a plan/package name has been specified, this option provides information on the explained SQL statements contained in this package. This information is taken from the DB2 catalog. In Dynamic mode, this option is not available.
Data from PLAN_TABLE	This option provides information from the PLAN_TABLE about the selected rows.
Evaluation of PLAN_TABLE	This option provides a description of the PLAN_TABLE. For each execution step, it describes: the locks chosen by DB2, whether a join operation is performed, whether the data is sorted and why the sort is performed, the access path in detail.
Catalog statistics	This option provides statistical information from the DB2 catalog.
Columns of used indexes	This option provides the columns of used indexes including catalog statistics on this columns.

Delete from PLAN_TABLE

The **Delete from PLAN_TABLE** function is also used to select PLAN_TABLE explanations depending on the specified combination of either plan name, DBRM name, and query number or collection name, package name, and query number. This time, however, the selected PLAN_TABLE explanations are not displayed but deleted.

The **Delete from PLAN_TABLE** function is useful to delete old data before either binding or re-binding a plan, or before executing an SQL EXPLAIN command.

To prevent PLAN_TABLE explanations from being deleted unintentionally, you are prompted for confirmation:

```
 16:50:23                 ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                            - Delete from PLAN_TABLE -



        The SQL Command

              DELETE FROM PLAN_TABLE
                 WHERE APPLNAME = ' '
                   AND COLLID = 'OLD'
                   AND PROGNAME LIKE 'ANY%'
                   AND QUERYNO BETWEEN 1 AND 2


        will be executed.



        Press PF5 to delete the data from the PLAN_TABLE or
              PF3 to return to the menu without deleting data

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help          Exit          Del                                    Canc
```

Apart from the **global PF-key settings**, with the **Delete from PLAN_TABLE** function of the **Explain PLAN_TABLE** facility, PF5 (Del) is used to confirm the deletion of previously selected explanations.

Explain PLAN_TABLE Facility for Mass and Batch Processing

An adapted version of the **Explain PLAN_TABLE** facility is also available for online mass processing and for batch mode execution.

EXPLAINB for Mass Processing

For online mass processing, a modified version of the **Explain PLAN_TABLE** facility is available.

≫ **To invoke the modified version of the Explain PLAN_TABLE facility**

1   Logon to the Natural system library SYSDB2.

2   In the command line, enter the command EXPLAINB and press ENTER.

    The following screen is displayed:

```
 16:45:35              ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                           - Explain PLAN_TABLE -




                Code Function

                 L    List PLAN_TABLE - Latest Explanations
                 A    List PLAN_TABLE - All    Explanations
                 O    Output Options
                 .    Exit

            Code .. _     Mode .......... DYNAMIC_ ( Dynamic, Plan, Package )
                          Plan .......... _____
                          Collection .... _____
                          DBRM/Package .. _____
                          Queryno ....... _____ - _____



 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help        Exit                                                    Canc
```

In addition to function codes L (**List PLAN_TABLE - Latest Entries** function) and A (**List PLAN_TABLE - All Entries** function), function code O (**Output Options**) is available.

The **Output Options** function enables you to restrict the output of information on PLAN_TABLE entries. The various options are listed in a window invoked by entering function code O on the above **Explain PLAN_TABLE** menu. The window is similar to the one invoked by the online I or INFO commands.

```
 16:53:20                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                          - Explain PLAN_TABLE -

         +-------------------------------------------------------+
         !                                                       !
         !                                                       !
         !                    Select what to display             !
         !                                                       !
         !            _ information about plan/package           !
         !            _ statements of plan/package               !
         !            _ data from PLAN_TABLE                      !
         !            _ evaluation of PLAN_TABLE                  !
         !            _ catalog statistics                       !
         !            _ columns of used indexes                  ! kage )
         !                                                       !
         !                                                       !
         +-------------------------------------------------------+
                      Queryno ....... _____ - _____



 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                   Exit                                                    Canc
```

If the **Output Options** function has been selected, only information covered by the options marked for output are printed.

If function code 0 has not been selected, all information on PLAN_TABLE entries covered by the options listed in the above window are printed.

In both cases, you are prompted for a printer.

EXPLAINB in Batch Mode

Apart from being used for online mass processing, the functionality of EXPLAINB is especially intended for batch processing. If EXPLAINB is used in batch mode, output is sent to a data set referred to by DD name CMPRT01 (logical printer 1).

11 File Server Statistics

If a file server has been installed, the file server statistics part of the **Natural Tools for DB2** is used to display statistics on the use of the file server.

≫ **To invoke the File Server Statistics function**

■ On the **Natural Tools for DB2 Main Menu**, enter function code F and press ENTER

The **File Server - Generation Statistics** screen is displayed:

```
  16:53:20                 ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                      - File Server - Generation Statistics -


     File Server Dataset Name ......: SAG.N2122.FSERV

     Enqueue Resource Name .........: FSERVV609

     Total Number of File Server Blocks ..........: 1000

     File Server Block Size ......................: 4080

     Number of Space Map Blocks ..................: 2

     Number of Global Directory Blocks ...........: 1
                              Entries ..........: 203
     User Space Allocation Quantities Primary ....: 50
                                      Secondary ..: 10
     Total Number of Blocks permitted per User ...: 200

  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help        Exit                                      Dire       Next  Canc
```

This screen provides information on parameters that must be specified when generating the file server.

If the file server storage medium is the Software AG Editor buffer pool, the **File Server - Generation Statistics** screen looks as follows:

```
 16:53:20                    ***** NATURAL TOOLS FOR DB2 *****                2009-10-30
                        - File Server - Generation Statistics -


   File Server Dataset Name ......: STORAGE MEDIUM IS EDITOR BUFFER POOL

   Enqueue Resource Name .........:

   Total Number of File Server Blocks ..........: 0

   File Server Block Size .....................: 4088

   Number of Space Map Blocks ..................: 0

   Number of Global Directory Blocks ...........: 0
                             Entries ..........: 0
   User Space Allocation Quantities Primary ....: 20
                                    Secondary ..: 10
   Total Number of Blocks permitted per User ...: 100

 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit                                              Next  Canc
```

If you press PF11 (Next), a second screen is displayed, the **File Server - User Statistics** screen, showing statistics that have been kept since the file server was installed - **Statistics since Generation** -, and statistics about the current Natural session - **Current Session Statistics**.

```
16:53:20              ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                       - File Server - User Statistics -

   Statistics since Generation:

   Active Users - Maximum Number: 3          Current Number: 1
   Maximum Number of used Blocks for single User ...........: 200
                                 for all Users .............: 200
   Number of Block Allocations PRIMARY .....................: 13
                               SECONDARY ...................: 17
   Number of free Blocks ...................................: 997
   Number of INIT SESSION Calls ............................: 65

   Current Session Statistics:

   Total Number of Blocks .................. ...: 0
                 Free Blocks ..................: 0
                 Secondary  Allocations ......: 0
   VSAM I/O Buffer inside DB2AREA ........... ..: YES   (Yes/No)

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help      Exit                               Dire  Prev        Canc
```

If you press PF10 (Prev), you are returned to the **File Server - Generation Statistics** screen.

Statistics are updated, each time you press ENTER, PF10, or PF11.

If the file server storage medium is the Software AG Editor buffer pool, the **File Server - User Statistics** screen looks as follows:

```
 16:53:20                   ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                          - File Server - User Statistics -

     Statistics since Generation:

     Active Users - Maximum Number: 3         Current Number: 0
     Maximum Number of used Blocks for single User ..........: 0
                                   for all Users ............: 0
     Number of Block Allocations PRIMARY ....................: 0
                                 SECONDARY ..................: 0
     Number of free Blocks ..................................: 0
     Number of INIT SESSION Calls ...........................: 0

     Current Session Statistics:

     Total Number of Blocks .................. ...: 20
                    Free Blocks .................: 20
                    Secondary  Allocations ......: 0
     VSAM I/O Buffer inside DB2AREA ........... ..: YES   (Yes/No)

  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
       Help       Exit                                      Prev        Canc
```

Note that the section **Statistics since Generation** could not be provided by this display.

For file server VSAM files, Natural for DB2 also provides file server directory display and maintenance.

If you press PF9 (Dire), the active directory entries are listed showing the session identifiers and their allocated file server blocks. The display looks like the following:

```
12:47:40              ***** NATURAL TOOLS FOR DB2 *****            2009-11-03
User XYZ               - File Server - Directory Entries -          TID TCD4

C  No  Tpsessid Birth       1st Block Last Block    Blocks Comment
------------------------------------------------------------------------------
_    0 Free Chn                   826        964       597 Checked
_    1 TCKK     pre NDB43         902        951        50 Checked
_    2 TCLB     pre NDB43          50         99        50 Checked
_    3 TCR0     pre NDB43         301        250        50 Checked
_    4 TCR7     pre NDB43         251        350        50 Checked
_    5 TCDW     pre NDB43         604        503        50 Checked
_    6 TCEX     pre NDB43         504        653        50 Checked
_    7 TCBW     2009-09-25        957        374        50 Checked
_    8 TC42     2009-10-15        357        993        50 Checked
_    9 - free -                     0          0         0 Empty Chain
_   10 - free -                     0          0         0 Empty Chain




Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Cont  Help        Exit  List  Pos   --    -     +     ++   Delet Fresh Canc  ↵
 ↵
```

Birth denotes a rough creation date of the file server session, if it was created by Natural for DB2 Version 4.3. If the file server session was created by an earlier version of Natural for DB2, the birthday of the file server session appears as `pre NDB43`.

The **Directory Entries** screen provides the functionality to scroll through the directory entries and to position a particular entry to the top of the screen.

In addition, **Directory Entries** allows you to list all file server block numbers of a directory entry (PF4, line command L) or to delete a directory entry from the file server (PF10, line command D). You should only delete directory entries, if you are sure the associated Natural session is no longer alive, otherwise the deletion could destroy the file server structure.

Directory Entries reflects the file server sessions at one particular point in time. By pressing PF11, the display will be refreshed from the file at another (actual) point in time.

12 Issuing DB2 Commands from Natural

The **DB2 Command** part of the **Natural Tools for DB2** enables you to issue DB2 commands from a Natural environment.

A file is maintained for each user on the system file `FUSER`. This file is stored under the object name `DB2$CMD` in the Natural library of the current user.

You can select a command and submit it, save the command file and save and/or print the output report.

Invoking the DB2 Command Part

≫ **To invoke the Interactive SQL function**

■ On the **Natural Tools for DB2 Main Menu**, enter function code `D` and press ENTER.

The **Execute DB2 Command** screen is displayed:

```
  16:07:56                 ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                             - Execute DB2 Command -




                        Code    Function

                          C     Display Commands
                          O     Display Output
                          ?     Help
                          .     Exit

                  Code .. _      Library .. DBA_____





  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
        Help        Exit                                                    Canc
```

The following functions are available:

Code	Description
C	Displays your command file. If you have not saved a command file yet, a default file is displayed.
O	If an output file exists, the output report is displayed.

The following parameter can be specified:

Parameter	Description
Library	You can enter a user name or library. The default is the currrent user ID.

Displaying the Command File

≫ **To display the command file**

■   On the **Execute DB2 Command** menu, enter function code C and press ENTER.

The **DB2 Commands** screen is displayed:

```
 16:12:11                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                              - DB2 Commands -

   Mark the line of the command you want to execute with 'S' and press PF4

   Cmd 1     _      -DISPLAY THREAD (*)........................................
   Cmd 2     _      -DISPLAY LOCATION..........................................
   Cmd 3     _      -DISPLAY DATABASE(*) LIMIT(2500)............................
                    ...........................................................
   Cmd 4     _      -DISPLAY PROCEDURE (*)......................................
                    ...........................................................
   Cmd 5     _      -DISPLAY DATABASE(DSNDB04) LIMIT (*)........................
                    ...........................................................
   Cmd 6     _      ...........................................................
                    ...........................................................
   Cmd 7     _      ...........................................................
                    ...........................................................
   Cmd 8     _      ...........................................................
                    ...........................................................
                    ...........................................................

  Command ===>
  Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                   Exit  Subm  Save                                 Next  Canc
```

Use PF11 (Next) to scroll to the next page.

You can modify the command file. Save your modifications with PF5 (Save).

≫ **To execute a command**

■ Mark the command with an S and press PF4 (Subm).

The results are displayed on the **DB2 Commands Output** screen:

```
 16:13:23                ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                            - DB2 Commands Output -


  Command:             -DISPLAY DATABASE(DSNDB04) LIMIT (*)
  Return Code 1:      00000000           Return Code 2:    00000000
  Length of Output:  00001AFB


  DSNT360I - **********************************
  DSNT361I - *  DISPLAY DATABASE SUMMARY
             *    GLOBAL
  DSNT360I - **********************************
  DSNT362I -     DATABASE = DSNDB04   STATUS = RW
                 DBD LENGTH = 72674
  DSNT397I -
  NAME     TYPE PART STATUS              PHYERRLO PHYERRHI CATALOG  PIECE
  -------- ---- ---- ------------------ -------- -------- -------- -----
  ADRESSE  TS        RW
  ALIASRBY TS        RW
  ALLDATA0 TS        RW



 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                   Exit        Save  --    -     +     ++                  Canc
```

≫ **To save the command file**

1 Press PF5 (Save).

The output file is stored under the object name DB2$OUT in the Natural library of the current user.

2 Press PF3 (Exit) to return to the command file.

You can submit further commands.

Displaying the Output Report

> **To display the last output record**

■ On the **Execute DB2 Command** menu, enter function code 0 and press ENTER.

The **DB2 Commands Output** screen is displayed:

```
 16:13:57                  ***** NATURAL TOOLS FOR DB2 *****              2009-10-30
                             - DB2 Commands Output -

  Command:              -DISPLAY DATABASE(*) LIMIT(2500)
  Return Code 1:     00000000           Return Code 2:   00000000
  Length of Output:  00007468

  DSNT360I - **********************************
  DSNT361I - *  DISPLAY DATABASE SUMMARY        *
            *    GLOBAL                         *
  DSNT360I - **********************************
  DSNT362I -     DATABASE = DSNDB01  STATUS = RW
                 DBD LENGTH = 8000
  DSNT397I -
  NAME     TYPE PART STATUS              PHYERRLO PHYERRHI CATALOG  PIECE
  -------- ---- ---- ----------------- -------- -------- -------- -----
  DBD01    TS        RW
  SPT01    TS        RW
  SCT02    TS        RW


 Command ===>
 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                   Exit        Print --    -     +     ++                  Canc
```

To print the output record, press PF5 (Print).

13 Using Natural System Commands for DB2

The following Natural system commands have been incorporated into the **Natural Tools for DB2**:

Natural System Command	Explanation
LISTSQL	Lists Natural DML statements and their corresponding SQL statements.
LISTSQLB	Provides explanations of SQL statements for a specific object.
SQLERR	Provides information of the SQLCA on a DB2 error.
SQLDIAG	Provides diagnostic information about the last SQL statement (other than a GET DIAGNOSTICS statement) that was executed.
LISTDBRM	Displays either a list of DBRMs (database request modules) for a particular Natural program or a list of Natural programs that reference a particular DBRM.

For a description of these commands, follow the links leading to the Natural *System Commands* documentation.

14 Generating Natural Data Definition Modules (DDMs)

To enable Natural to access a DB2 table, a logical Natural data definition module (DDM) of the table must be generated. This is done either with Predict (see the relevant Predict documentation for details) or with the Natural utility SYSDDM; see also *SYSDDM Utility* in the Natural *Editors* documentation.

If you do not have Predict installed, use the SYSDDM function **SQL Services** to generate Natural DDMs from DB2 tables. This function is invoked from the main menu of SYSDDM and is described on the following pages.

For further information on Natural DDMs, see *Data Definition Modules - DDMs* in the Natural *Programming Guide*.

SQL Services (NDB)

The **SQL Services (NDB)** function of the Natural SYSDDM utility (see *Using SYSDDM Maintenance and Service Functions* in the Natural *Editors* documentation) is used to access DB2 tables. You access the catalog of the DB2 server to which you are connected, for example, by using the **Environment Setting** function as described in *Natural Tools for DB2*, or by entering the name of a server in the **Server Name** field on the **SQL Services Menu**. The name of the DB2 server to which you are connected is then displayed in the top left-hand corner of the screen **SQL Services Menu**. You can access any DB2 server that is located on either a mainframe (z/OS) or a UNIX platform if the servers have been connected via DRDA (Distributed Relational Database Architecture). For further details on connecting DB2 servers and for information on binding the application package (SYSDDM uses I/O module NDBIOMO) to access data on remote servers, refer to the relevant IBM literature.

The **SQL Services** function determines whether you are connected to a mainframe DB2 (z/OS) or a UNIX DB2, access the appropriate DB2 catalog and performs the functions listed below.

> **Note:** If you use SYSDDM **SQL Services** in a CICS environment without file server, specify CONVERS=ON in the NTDB2 macro; otherwise you might get SQLCODE -518.

- Using SQL Services
- Select SQL Table from a List
- Generate DDM from an SQL Table
- List Columns of an SQL Table

■ Making a User Exit Routine Available

Using SQL Services

≫ **To invoke the SQL Services function**

1   In the command line, enter the Natural system command `SYSDDM` and press Enter.

Or:

1. From the Natural main menu, choose **Maintenance and Transfer Utilities** to display the **Maintenance and Transfer Utilities** menu.

2. From the **Maintenance and Transfer Utilities** menu, choose **Maintain DDMs**.

The menu of the `SYSDDM` utility is displayed. The fields and functions provided on the `SYSDDM` utility menu are explained in the section *Using SYSDDM Maintenance and Service Functions*.

2   In the **Code** field of the Natural `SYSDDM` utility **Menu**, enter code `B` and press Enter.

The **SQL Services Menu** is displayed.

```
11:31:39              ***** NATURAL SYSDDM UTILITY *****            2009-11-27
 Server DAEFDB29              - SQL Services: Menu -



                   Code  Function

                    S    Select SQL Table from a List
                    G    Generate DDM from an SQL Table
                    L    List Columns of an SQL Table
                    ?    Help
                    .    Exit

             Code ... _
         Table name ... _____
         Creator ...... _____
         Replace ...... N (Y,N)         DDM Name with Creator .. Y (Y/N)
         Server name .. DAEFDB29_____
         Remark ....... O (Overwrite/SQL/Comment)




Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
     Help      Exit                                                       Canc
```

The functions available on this screen are described in the corresponding sections.

Select SQL Table from a List

This function is used to select a DB2 table from a list for further processing.

≫ To invoke the Select SQL Table from a List function

■ On the **SQL Services Menu**, enter Function Code S.

   ■ If you enter the function code only, you obtain a list of all tables defined to the DB2 catalog.

   ■ If you do not want a list of all tables but would like only a certain range of tables to be listed, you can, in addition to the function code, specify a value in the **Table Name** and/or **Creator** fields. You can use asterisk notation (*) or the greater-than character (>) for a start value.

Press ENTER.

The **Select SQL Table From A List** screen is invoked displaying a list of all DB2 tables requested. On the list, you can mark a DB2 table with a function code:

Code	Function	Description
G	**Generate DDM from an SQL Table**	This function can be used to generate a Natural DDM from a DB2 table, based on the definitions in the DB2 catalog.
L	**List Columns of an SQL Table**	This function lists all columns of a specific DB2 table.

Generate DDM from an SQL Table

This function is used to generate a Natural DDM from a DB2 table, based on the definitions in the DB2 catalog.

The following topics are covered below:

- Invoking the Generate DDM from an SQL Table function
- Assigning Default Values - Generating DDMs in Batch
- DBID/FNR Assignment
- Long Field Generation
- Length Indicator for Variable Length Fields: VARBINARY, VARCHAR, LONG VARCHAR, VARGRAPHIC, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB
- Null Values

■ Locator Field for LOB Column

Invoking the Generate DDM from an SQL Table function

≫ To invoke the function

■ On the **SQL Services Menu**, enter function code G along with the name and creator of the table for which you wish a DDM to be generated.

  ■ If you do not know the table name/creator, you can use the function **Select SQL Table from a List** to choose the table you want.

  ■ If you do not want the creator of the table to be part of the DDM name, enter an N (No) in the field **DDM Name with Creator**. The default setting is is Y (Yes).

  ■ If you wish to generate a DDM for a table for which a DDM already exists and you want the existing one to be replaced by the newly generated one, enter a Y (Yes) in the **Replace** field.

  By default, **Replace** is set to N (No) to prevent an existing DDM from being replaced accidentally.

  ■ In the **Remark** field you can specify the contents of the DDM **Remark** column. Enter:

O (Overwrite)	for SQL column remarks if defined, overwritten by field information generated by Natural if available. This is the default setting;
S (SQL)	for SQL column remarks if defined and blank otherwise;
C (Comment)	for field information generated by Natural if available. SQL column remarks will be copied to a separate DDM comment line.

  By default, **Remark** is set to O (Overwrite).

  ■ To define or alter a default value for the fields **Code**, **Table Name**, **Creator**, **Replace**, **DDM Name with Creator** or **Remark** use user exit NDBDDM-2 and its data area NDBDDM-L provided in library SYSDB2. See *Making a User Exit Routine Available*. For detailed information on how to handle NDBDDM-2, refer to the remarks in its source.

  ⚠ **Important:** Since the specification of any special characters as part of a field or DDM name does not comply with Natural naming conventions, any special characters allowed within DB2 must be avoided. DB2 delimited identifiers must be avoided, too.

Assigning Default Values - Generating DDMs in Batch

To avoid user interaction popup windows during DDM field generation, the user exit `NDBDDM-1` and its data area `NDBDDM-L` provided in library `SYSDB2` can be used. For detailed information on how to handle `NDBDDM-1`, refer to the remarks in its source. See also *Making a User Exit Routine Available*.

DBID/FNR Assignment

When the **Generate DDM from an SQL Table** function is invoked for a table for which a DDM is to be generated for the first time, the **DBID/FNR Assignment** screen is displayed. If a DDM is to be generated for a table for which a DDM already exists, the existing DBID and FNR are used and the **DBID/FNR Assignment** screen is suppressed.

On the **DBID/FNR Assignment** screen, enter one of the database IDs (DBIDs) chosen at Natural installation time, and the file number (FNR) to be assigned to the DB2 table. Natural requires these specifications for identification purposes only.

The range of DBIDs which is reserved for DB2 tables is specified in the `NTDB` macro of the Natural parameter module (see the Natural *Parameter Reference* documentation) for the database type DB2. Any DBID not within this range is not accepted. The FNR can be any valid file number within the database (between 1 and 65535).

After a valid DBID and FNR have been assigned, a DDM is automatically generated from the specified table.

Long Field Generation

The maximum field length supported by Natural is 1 GB-1 (1073741823 bytes). If a DB2 table contains a column which is longer than 253 bytes or if a DB2 column is defined as a DB2 LOB field, the pop-up window Long Field Generation will be invoked automatically. A DB2 LOB field may be defined as a simple Natural variable with a maximum length of 1GB-1, or as a dynamic Natural variable.

A field which is longer than 253 bytes and which is not a DB2 LOB field may be defined as a simple Natural field with a maximum length of 1GB-1, or as an array. In the DDM, such an array is represented as a multiple-value variable.

If, for example, a DB2 column has a length of 2000 bytes, you can specify an array element length of 200 bytes, and you receive a multiple-value field with 10 occurrences, each occurrence with a length of 200 bytes.

Since generated long fields are not multiple-value fields in the sense of Natural, the Natural C* notation makes no sense here and is therefore not supported.

When such a generated long field is defined in a Natural view to be referenced by Natural SQL statements (that is, by host variables which represent multiple-value fields), both when defined

and when referenced, the specified range of occurrences (index range) must always start with oc-currence 1. If not, a Natural syntax error is returned.

Example:

```
UPDATE table SET varchar = #arr(*)
SELECT ... INTO #arr(1:5)
```

> **Note:** When such a generated long field is updated with the Natural DML `UPDATE` statement, care must be taken to update each occurrence appropriately.

Length Indicator for Variable Length Fields: VARBINARY, VARCHAR, LONG VARCHAR, VARGRAPHIC, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB

For each of the column types listed above, an additional length indicator field (format/length I2 or I4 for `LOB` fields) is generated in the DDM. The length is always measured in number of characters, not in bytes. To obtain the number of bytes of a `VARGRAPHIC`, `LONG VARGRAPHIC` or `DBCLOB` field, the length must be multiplied by 2.

The name of a length indicator field begins with `L@` followed by the name of the corresponding field. The value of the length indicator field can be checked or updated by a Natural program.

If the length indicator field is not part of the Natural view and if the corresponding field is a re-defined long field, the length of this field with `UPDATE` and `STORE` operations is calculated without trailing blanks.

Null Values

With Natural, it is possible to distinguish between a null value and the actual value zero (0) or blank in a DB2 column.

When a Natural DDM is generated from the DB2 catalog, an additional `NULL` indicator field is generated for each column which can be `NULL`; that is, which has neither `NOT NULL` nor `NOT NULL WITH DEFAULT` specified.

The name of the `NULL` indicator field begins with `N@` followed by the name of the corresponding field.

When the column is read from the database, the corresponding indicator field contains either zero (0) (if the column contains a value, including the value `0` or blank) or `-1` (if the column contains no value).

Example:

The column `NULLCOL CHAR(6)` in a DB2 table definition would result in the following view fields:

```
NULLCOL          A 6.0
N@NULLCOL     I 2.0
```

When the field `NULLCOL` is read from the database, the additional field `N@NULLCOL` contains:

- `0` (zero) if `NULLCOL` contains a value (including the value `0` or blank),

- `-1` (minus one) if `NULLCOL` contains no value.

A null value can be stored in a database field by entering `-1` as input for the corresponding `NULL` indicator field.

> **Note:** If a column is `NULL`, an implicit `RESET` is performed on the corresponding Natural field.

Locator Field for LOB Column

For each `LOB` column, an additional locator field will be generated in the I4 format.

A `LOB` locator may be used to reference a `LOB` value in the DB2 database server, when a `LOB` value is not needed locally in a program.

List Columns of an SQL Table

This function lists all columns of a specific DB2 table.

≫ **To invoke the List Columns function**

- On the **SQL Services Menu**, enter function code `L` along with the name and creator of the table whose columns you wish to be listed, and press Enter.

  The **List Columns** screen for this table is invoked, which lists all columns of the specified table and displays the following information for each column:

Variable	Content	
Name	The DB2 name of the column.	
Type	The column type.	
Length	The length (or precision if type is `DECIMAL`) of the column as defined in the DB2 catalog.	
Scale	The decimal scale of the column (only applicable if type is `DECIMAL`).	
Update	Y	The column can be updated.
	N	The column cannot be updated.
Nulls	Y	The column can contain null values.
	N	The column cannot contain null values.

Variable	Content
Not	A column whose scale length or whose type is not supported by Natural is marked with an asterisk (*). For such a column, a view field cannot be generated. The maximum scale length supported is 7 bytes. The following SQL types are supported: BIGINT, BINARY, VARBINARY, DECFLOAT, XML CHAR, VARCHAR, LONG VARCHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DECIMAL, INTEGER, SMALLINT, DATE, TIME, TIMESTAMP, FLOAT, ROWID, BLOB, CLOB and DBCLOB.

The data types DATE, TIME, TIMESTAMP, FLOAT and ROWID are converted into numeric or alphanumeric fields of various lengths: DATE is converted into A10, TIME into A8, TIMESTAMP into A26, FLOAT into F8 and ROWID into A40. DATE and TIME could be mapped alternatively to Natural DATE and Natural TIME respectively.

For DB2, Natural provides a DB2 TIMESTAMP column as an alphanumeric field (A26) in the format *YYYY-MM-DD-HH.II.SS.MMMMMM*. Alternatively, you can generate the Natural TIME field (data format T) as DB2 TIMESTAMP data type if the DBTSTI option of the COMPOPT system command is set to ON (see the *System Commands* documentation).

You can use the Natural subprogram NDBSTMP to compute TIMESTAMP (A26) fields.

Making a User Exit Routine Available

You can customize the **Generating Natural Data Definition Modules (DDMs)** map with user exit routine NDBDDM-1 or NDBDDM-2.

≫ **To make user exit routine** NDBDDM-1 **or** NDBDDM-1 **available**

1   Catalog the NDBDDM-*num* source object under the name NDBDDMU*num* in library SYSDB2.

   📄   **Note:** The names of the source object and the cataloged object of the user exit routine must be different to ensure that the overwriting of the source object during an update installation does not affect the cataloged object.

2   Copy NDBDDMU*num* to steplib SYSLIBS.

A subprogram used by SYSDDM searches for NDBDDMU*num* in steplib SYSLIBS.

15 Dynamic and Static SQL Support

This section describes the dynamic and static SQL support provided by Natural.

Related Documentation

- For a list of error messages that may be issued during static generation, see *Static Generation Messages and Codes Issued under NDB* in the Natural *Messages and Codes* documentation.

- For information on Static SQL with Natural Security, see *Integration with Natural Security*.

SQL Support - General Information

The SQL support of Natural combines the flexibility of dynamic SQL support with the high performance of static SQL support.

In contrast to static SQL support, the Natural dynamic SQL support does not require any special consideration with regard to the operation of the SQL interface. All SQL statements required to execute an application request are generated automatically and can be executed immediately with the Natural RUN command. Before executing a program, you can look at the generated SQLCODE, using the LISTSQL command.

Access to DB2 through Natural has the same form whether dynamic or static SQL support is used. Thus, with static SQL support, the same SQL statements in a Natural program can be executed in either dynamic or static mode. An SQL statement can be coded within a Natural program and, for testing purposes, it can be executed using dynamic SQL. If the test is successful, the SQL statement remains unchanged and static SQL for this program can be generated.

Thus, during application development, the programmer works in dynamic mode and all SQL statements are executed dynamically, whereas static SQL is only created for applications that have been transferred to production status.
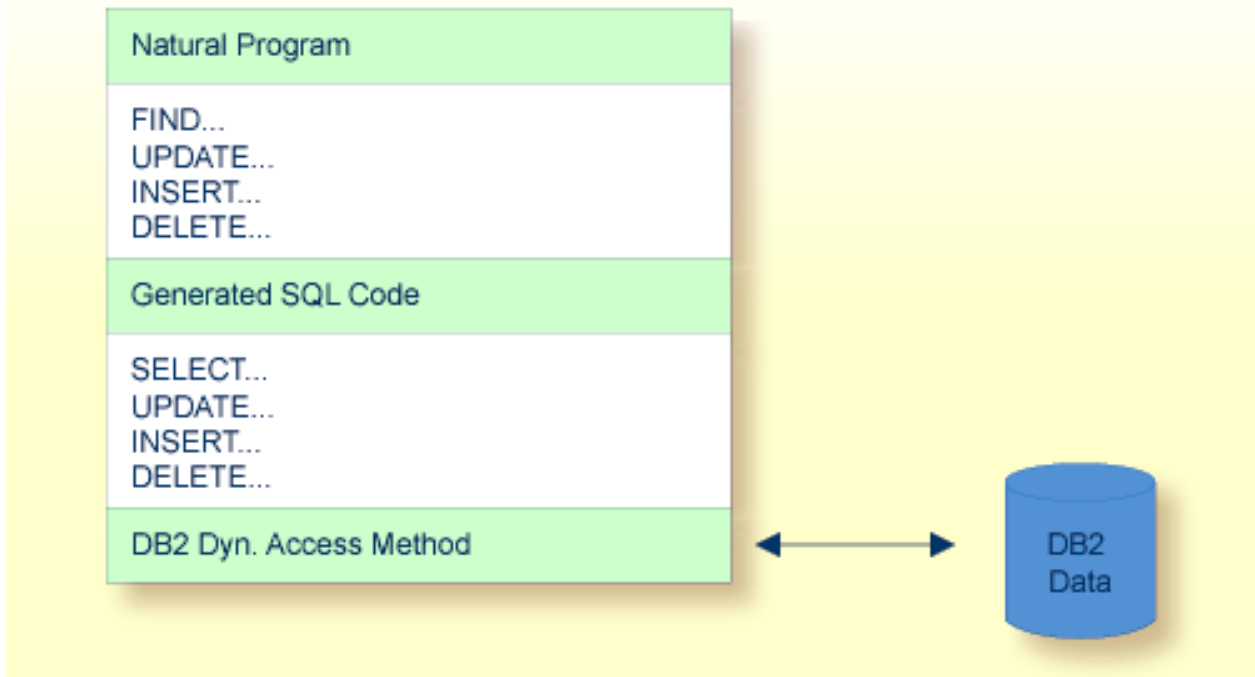
Internal Handling of Dynamic Statements

Natural automatically provides for the preparation and execution of each SQL statement and handles the opening and closing of cursors used for scanning a table.

The following topics are covered:

- I/O Module NDBIOMO for Dynamic SQL Statement Execution
- Statement Table
- Processing of SQL Statements Issued by Natural

I/O Module NDBIOMO for Dynamic SQL Statement Execution

As each dynamic execution of an SQL statement requires a statically defined `DECLARE STATEMENT` and `DECLARE CURSOR` statement, a special I/O module named `NDBIOMO` is provided which contains a fixed number of these statements and cursors. This number is specified during the generation of the `NDBIOMO` module in the course of the Natural for DB2 installation process.

Statement Table

If possible, an SQL statement is only prepared once and can then be executed several times if required. For this purpose, Natural internally maintains a table of all SQL statements that have been prepared and assigns each of these statements to a DECLAREd STATEMENT in the module NDBIOMO. In addition, this table maintains the cursors used by the SQL statements SELECT, FETCH, UPDATE (positioned), and DELETE (positioned).

Each SQL statement is uniquely identified by:

- the name of the Natural program that contains this SQL statement,

- the line number of the SQL statement in this program,

- the name of the Natural library into which this program was stowed,

- the time stamp when this program was stowed.

Once a statement has been prepared, it can be executed several times with different variable values, using the dynamic SQL statement EXECUTE USING DESCRIPTOR or OPEN CURSOR USING DESCRIPTOR.

When the full capacity of the statement table is reached, the entry for the next prepared statement overwrites the entry for a free statement whose latest execution is the least recent one.

When a new SELECT statement is requested, a free entry in the statement table with the corresponding cursor is assigned to it and all subsequent FETCH, UPDATE, and DELETE statements referring to this SELECT statement will use this cursor. Upon completion of the sequential scanning of the table, the cursor is released and free for another assignment. While the cursor is open, the entry in the statement table is marked as used and cannot be reused by another statement.

If the number of nested FIND (SELECT) statements reaches the number of entries available in the statement table, any further SQL statement is rejected at execution time and a Natural error message is returned.

The size of the statement table depends on the size specified for the module NDBIOMO. Since the statement table is contained in the DB2 buffer area, the setting of Natural profile parameter DB2SIZE (see also *Natural Parameter Modifications for Natural for DB2* in *Installing Natural for DB2 on z/OS* in the *Installation* documentation) may not be sufficient and may need to be increased.

Processing of SQL Statements Issued by Natural

The embedded SQL uses cursor logic to handle SELECT statements. The preparation and execution of a SELECT statement is done as follows:

1. The typical SELECT statement is prepared by a program flow which contains the following embedded SQL statements (note that *X* and *SQLOBJ* are SQL variables, not program labels):

```
DECLARE SQLOBJ STATEMENT
DECLARE X CURSOR FOR SQLOBJ
INCLUDE SQLDA (copy SQL control block)
```

Then, the following statement is moved into SQLSOURCE:

```
SELECT PERSONNEL_ID, NAME, AGE
 FROM EMPLOYEES
 WHERE NAME IN (?, ?)
    AND AGE BETWEEN ? AND ?
```

> **Note:** The question marks (?) above are parameter markers which indicate where values are to be inserted at execution time.

```
PREPARE SQLOBJ FROM SQLSOURCE
```

2. Then, the SELECT statement is executed as follows:

```
OPEN X USING DESCRIPTOR SQLDA
FETCH X USING DESCRIPTOR SQLDA
```

The descriptor SQLDA is used to indicate a variable list of program areas. When the OPEN statement is executed, it contains the address, length, and type of each value which replaces a parameter marker in the WHERE clause of the SELECT statement. When the FETCH statement is executed, it contains the address, length, and type of all program areas which receive fields read from the table.

When the FETCH statement is executed for the first time, it sets the Natural system variable *NUMBER to a non-zero value if at least one record is found that meets the search criteria. Then, all records satisfying the search criteria are read by repeated execution of the FETCH statement.

To help improve performance, especially when using distributed databases, the DB2-specific FOR FETCH ONLY clause can be used. This clause is generated and executed if rows are to be retrieved only; that is, if no updating is to take place.

3. Once all records have been read, the cursor is released by executing the following statement:

```
CLOSE X
```

Preparing Programs for Static Execution

This section describes how to prepare Natural programs for static execution.

The following topics are covered:

- Basic Principles
- Generation Procedure: CMD CREATE Command
- Precompilation of the Generated Assembler Program
- Modification Procedure: CMD MODIFY Command
- BIND of the Precompiled DBRM
- Impact of the DB2 MF parameter

For an explanation of the symbols used in this section to describe the syntax of Natural statements, see *Syntax Symbols* in the *Statements* documentation.

Basic Principles

Static SQL is generated in Natural batch mode for one or more Natural applications which can consist of one or more Natural object programs. The number of programs that can be modified for static execution in one run of the generation procedure is limited to 999.

During the generation procedure, the database access statements contained in the specified Natural objects are extracted, written to work files, and transformed into a temporary Assembler program. If no Natural program is found that contains SQL access or if any error occurs during static SQL generation, batch Natural terminates and condition code 40 is returned, which means that all further JCL steps must no longer be executed.

The Natural modules `NDBCHNK` and `NDBSTAT` must reside in a steplib of the generation step. Both are loaded dynamically during the execution of the generation step.

The temporary Assembler program is written to a temporary file (the Natural work file `CMWKF06`) and precompiled. The size of the workfile is proportional to the maximum number of programs, the number of SQL statements and the number of variables used in the SQL statements. During the precompilation step, a database request module (DBRM) is created, and after the precompilation step, the precompiler output is extracted from the Assembler program and written to the corresponding Natural objects, which means that the Natural objects are modified (prepared) for static execution. The temporary Assembler program is no longer used and deleted.

A static database request module is created by using either the sample job provided on the installation medium or an appropriate job created with the **Create DBRM** function.

Generation Procedure: CMD CREATE Command

The following topics are covered:

- Generating Static SQL for Natural Programs
- Static Name
- USING Clause

Generating Static SQL for Natural Programs

≫ To generate static SQL for Natural programs

1   Logon to the Natural system library `SYSDB2`.

Since a new `SYSDB2` library has been created when installing Natural for DB2, ensure that it contains all Predict interface programs necessary to run the static SQL generation. These programs are loaded into `SYSDB2` at Predict installation time (see the relevant *Predict* product documentation).

2   Specify the `CMD CREATE` command and the Natural input necessary for the static SQL generation process; the `CMD CREATE` command has the following syntax:

```
CMD CREATE DBRM static-name USING using-clause
{application-name,object-name,excluded-object}
:
:
```

The generation procedure reads but does not modify the specified Natural objects. If one of the specified programs was not found or had no SQL access, return code 4 is returned at the end of the generation step.

Static Name

If the `PREDICT DOCUMENTATION` option is to be used, a corresponding Predict static SQL entry must be available and the *static-name* must correspond to the name of this entry. In addition, the *static-name* must correspond to the name of the DBRM to be created during precompilation. The *static-name* can be up to 8 characters long and must conform to Assembler naming conventions.

USING Clause

The *using-clause* specifies the Natural objects to be contained in the DBRM. These objects can either be specified explicitly as INPUT DATA in the JCL or obtained as PREDICT DOCUMENTATION from Predict.

```
{ INPUT DATA       }  [ WITH { YES   } ]  [ FS { OFF } ]  [LIB           [ DCTODP { OFF } ] ]
{ PREDICT          }  [      { NO    } ]  [    { ON  } ]  lib-name               { ON  }
{ DOCUMENTATION    }  [ XREF { FORCE } ]                 ]
```

If the parameters to be specified do not fit in one line, specify the command identifier (CMD) and the various parameters in separate lines and use both the input delimiter (as specified with the Natural profile/session parameter ID - default is a comma (,) - and the continuation character indicator - as specified with the Natural profile/session parameter CF; default is a percent (%) - as shown in the following example:

Example:

```
CMD
CREATE,DBRM,static,USING,PREDICT,DOCUMENTATION,WITH,XREF,NO,%
LIB,library
```

Alternatively, you can also use abbreviations as shown in the following example:

Example:

```
CMD CRE DBRM static US IN DA W XR Y FS OFF LIB library
```

The sequence of the parameters USING, WITH, FS, and LIB is optional.

INPUT DATA

As input data, the applications and names of the Natural objects to be included in the DBRM must be specified in the subsequent lines of the job stream (*application-name*, *object-name*). A subset of these objects can also be excluded again (*excluded-objects*). Objects in libraries whose names begin with SYS can be used for static generation, too.

The applications and names of Natural objects must be separated by the input delimiter - as specified with the Natural profile parameter ID; default is a comma (,). If you wish to specify all objects whose names begin with a specific string of characters, use an *object-name* or *excluded-objects* name that ends with asterisk notation (*). To specify all objects in an application, use asterisk notation only.

Example:

```
LIB1,ABC*
  LIB2,A*,AB*
  LIB2,*
  :
  .
```

The specification of applications/objects must be terminated by a line that contains a period (.) only.

PREDICT DOCUMENTATION

Since Predict supports static SQL for DB2, you can also have Predict supply the input data for creating static SQL by using already existing `PREDICT DOCUMENTATION`.

WITH XREF Option

Since Predict Active References supports static SQL for DB2, the generated static DBRM can be documented in Predict, and the documentation can be used and updated with Natural.

`WITH XREF` is the option which enables you to store cross-reference data for a static SQL entry in Predict each time a static DBRM is created (`YES`). You can instead specify that no cross-reference data are stored (`NO`) or that a check is made to determine whether a Predict static SQL entry for this static DBRM already exists (`FORCE`). If so, cross-reference data are stored; if not, the creation of the static DBRM is not allowed. For more detailed information on Predict Active References, refer to the relevant Predict documentation.

When `WITH XREF (YES/FORCE)` is specified, `XREF` data are written for both the Predict static SQL entry (if defined in Predict) and each generated static Natural program. However, static generation with `WITH XREF (YES/FORCE)` is possible only if the corresponding Natural programs have been cataloged with `XREF ON`.

`WITH XREF FORCE` only applies to the `USING INPUT DATA` option.

> **Note:** If you do not use Predict, the `XREF` option must be omitted or set to `NO` and the module `NATXRF2` need not be linked to the Natural nucleus.

FS Option

If the `FS` (file server) option is set to `ON`, a second `SELECT` is generated for the **Natural file server for DB2**. `ON` is the default setting.

If the `FS` option is set to `OFF`, no second `SELECT` is generated, which results in less SQL statements being generated in your static DBRM and thus in a smaller DBRM.

LIB Option

With the `LIB` (library) option, a Predict library other than the default library (`*SYSSTA*`) can be specified to contain the Predict static SQL entry and `XREF` data. The name of the library can be up to eight characters long.

DCTODP Option

The `DCTODP` option is only relevant if you are doing a static generation for programs which have been catalogued with Natural parameter DC=','.

By the `DCTODP` parameter it can be determined whether the static generation will change decimal literals in SQL statements by replacing the decimal character comma ',' by the decimal character period '.' in the generated static SQL assembler program. This will be necessary since the DB2 precompiler does not support decimal point comma in decimal literals.

When `DCTODP OFF` is specified no conversion of decimal point comma to decimal point period will take place. `DCTODP OFF` is the default value.

When `DCTODP ON` is specified static generation will convert decimal point comma to decimal point period.

If `DCTODP ON` is used, you should be sure your programs are coded unambiguously when using the comma as separator in function calls and various SQL clauses like `IN` or `ORDER BY` so that the comma as separator in element lists can't be misinterpreted as decimal point of a decimal literal

For instance, if you code a SQL `IN` clause while using DC=',' like

Column-name `IN` (10,20,30,40).

Static generation with `DCTODP ON` change the `IN` clause to

Column-name `IN` (10,20 , 30,40).

`IN` list contains two values 10.20 and 30.40.

The same `IN` clause in static generation with `DCTODP OFF` (default value) will be changed to

Column-name `IN` (10,20 , 30,40).

`IN` list contains four values 10 , 20 , 30 and 40.

If you had coded

Column-name `IN` (10 , 20 , 30 , 40).

Static generation will always generate the `IN` clause with `DCTODP` either `ON` or `OFF` to

Column-name `IN` (10 , 20 , 30 , 40).

Precompilation of the Generated Assembler Program

In this step, the precompiler is invoked to precompile the generated temporary Assembler program. The precompiler output consists of the DBRM and a precompiled temporary Assembler program which contains all the database access statements transformed from SQL into Assembler statements.

Later, the DBRM serves as input for the `BIND` step and the Assembler program as input for the modification step.

Modification Procedure: CMD MODIFY Command

The modification procedure modifies the Natural objects involved by writing precompiler information into the object and by marking the object header with the *static-name* as specified with the `CMD CREATE` command.

In addition, any existing copies of these objects in the Natural global buffer pool (if available) are deleted and `XREF` data are written to Predict (if specified during the generation procedure).

≫ **To perform the modification procedure**

1   Logon to the Natural system library `SYSDB2`.

2   Specify the `CMD MODIFY` command which has the following syntax:

```
CMD MODIFY [XREF]
```

The input for the modify step is the precompiler output which must reside on a data set defined as the Natural work file `CMWKF01`.

The output consists of precompiler information which is written to the corresponding Natural objects. In addition, a message is returned telling you whether it was the first time an object was modified for static execution (modified) or whether it had been modified before (re-modified).

Assembler/Natural Cross-References

If you specify the `XREF` option of the `CMD MODIFY` command, an output listing is created on the work file `CMWKF02`, which contains the DBRM name and the Assembler statement number of each statically generated SQL statement together with the corresponding Natural source code line number, program name, library name, database ID and file number.

```
    ----------------------------------------------------------------  ....
     DBRMNAME STMTNO      LINE NATPROG  NATLIB   DB  FNR COMMENT       ....
    ----------------------------------------------------------------  ....
     TESTDBRM 000627      0390 TESTPROG SAG      010 042 INSERT        ....
              000641      0430                           INSERT        ....
              000652      0510                           SELECT        ....
              000674      0570                           SELECT        ....
              000698      0570                           SELECT   2ND  ....
              000728      0650                           UPD/DEL       ....
              000738      0650                           UPD/DEL  2ND  ....
              000751      0700                           SELECT        ....
              000775      0700                           SELECT   2ND  ....
```

Column	Explanation
DBRMNAME	Name of the DBRM which contains the static SQL statement.
STMTNO	Assembler statement number of the static SQL statement.
LINE	Corresponding Natural source code line number.
NATPROG	Name of the Natural program that contains the static SQL statement.
NATLIB	Name of the Natural library that contains the Natural program.
DB / FNR	Natural database ID and file number.
COMMENT	Type of SQL statement, where 2ND indicates that the corresponding statement is used for a reselection; see also the *Concept of the File Server*.

BIND of the Precompiled DBRM

We recommend that you execute the DB2 BIND command *after* the CMD MODIFY command.

The DB2 BIND command binds the DBRM into a DB2 package. You can bind one or more DB2 packages into a DB2 application plan. In addition to the packages of static DBRMs created with the CMD CREATE command, this application plan can also contain the package of the DBRM of the NDBIOMO module Natural provides for dynamic SQL execution.

A DBRM can be bound into any number of packages and the packages can be bound into any number of application plans where required. A plan is physically independent of the environment where the program is to be run. However, you can group your packages logically into plans which are to be used for either batch or online processing, where the same package can be part of both a batch plan and an online plan.

Unless you are using plan switching, only one plan can be executed per Natural session. Thus, you must ensure that the plan name specified in the BIND step is the same as the one used to execute Natural.

Impact of the DB2 MF parameter

The DB2 MF parameter setting of the Natural session that performs the static generation, determines whether the `DECLARE CURSOR` statements of the generated static program contain the `WITH ROWSET POSITIONING` clause.

The `WITH ROWSET POSITIONING` clause enables the static program to use multi-fetching.

Code	Explanation
MF>0	`WITH ROWSET POSITIONING` clause is added to the `DECLARE CURSOR` statements. Multi-fetching will be used with the value of the `MF` parameter at execution time.
MF=0	No `WITH ROWSET POSITIONING` clause is added to the `DECLARE CURSOR` statements. Multi-fetching is not possible, regardless how the `MF` parameter is set at execution time. The generated static program can only operate with single-fetching.

Execution of Natural in Static Mode

To be able to execute Natural in static mode, all users of Natural must have the DB2 `EXECUTE PLAN/PACKAGE` privilege for the plan created in the `BIND` step.

To execute static SQL, start Natural and execute the corresponding Natural program. Internally, the Natural runtime interface evaluates the precompiler data written to the Natural object and then performs the static accesses.

To the user there is no difference between dynamic and static execution.

Mixed Dynamic/Static Mode

It is possible to operate Natural in a mixed static and dynamic mode where for some programs static SQL is generated and for some not.

The mode in which a program is run is determined by the Natural object program itself. If a static DBRM is referenced in the executing program, all statements in this program are executed in static mode.

> **Note:** Natural programs which return a runtime error do not automatically execute in dynamic mode. Instead, either the error must be corrected or, as a temporary solution, the Natural program must be recataloged to be able to execute in dynamic mode.

Within the same Natural session, static and dynamic programs can be mixed without any further specifications. The decision which mode to use is made by each individual Natural program.

Messages and Codes

For a list of error messages that may be issued during static generation, refer to *Static Generation Messages and Codes Issued under NDB* in the Natural *Messages and Codes* documentation.

Application Plan Switching

This section describes how to switch application plans within the same Natural session in different TP-monitor environments or in batch mode.

The following topics are covered:

- Basic Principles of Plan Switching
- Plan Switching under CICS
- Plan Switching under Com-plete
- Plan Switching under IMS TM
- Plan Switching under TSO and in Batch Mode

Basic Principles of Plan Switching

When using application plan switching, you can switch to a different application plan within the same Natural session.

If a second application plan is to be used, this can be specified by executing the Natural program `NATPLAN`. `NATPLAN` is contained in the Natural system library `SYSDB2` and can be invoked either from within a Natural program or dynamically by entering the command `NATPLAN` at the `NEXT` prompt. The only input value required for `NATPLAN` is an eight-character plan name. If no plan name is specified, you are prompted by the system to do so.

Before executing `NATPLAN`, ensure that any open DB2 recovery units are closed.

Since the `NATPLAN` program is also provided in source form, user-written plan switching programs can be created using similar logic.

The actual switch from one plan to another differs in the various environments supported. The feature is available under Com-plete, CICS, and IMS TM MPP. When using the Call Attachment Facility (CAF) or Resource Recovery Services Attachment Facility (RRSAF), it is also available in TSO and batch environments.

In some of these environments, a transaction ID or code must be specified instead of a plan name.

Plan Switching under CICS

Under CICS, you have the option of using either plan switching by transaction ID (default) or dynamic plan selection exit routines. Thus, by setting the field `#SWITCH-BY- TRANSACTION-ID` in the `NATPLAN` program to either `TRUE` or `FALSE`, either the subroutine `CMTRNSET` or the desired plan name is written to temporary storage queue.

For more information on activating plan switching under CICS, see *Installation Steps Specific to CICS* in the *Installing Natural for DB2 on z/OS* documentation.

Below is information on:

- Plan Switching by CICS/DB2 Exit Routine

Plan Switching by CICS/DB2 Exit Routine

If `#SWITCH-BY-TRANSACTION-ID` is set to `FALSE`, the desired plan name is written to a temporary storage queue for a CICS/DB2 exit routine specified as PLANExit attribute of a DB2ENTRY or of the DB2CONN definition, the `NATPLAN` program must be invoked before the first DB2 access. Natural for DB2 provides `NDBUEXT` as CICS DB2 plan selection exit program. For additional information on CICS/DB2 exit routines, refer to the relevant IBM literature.

The name of the temporary storage queue is `PLANxxxx`, where `xxxx` is the CICS terminal identifier.

When running in a CICSplex environment, the CICS temporary storage queue `PLANxxxx` containing the plan name must be defined with `TYPE=SHARED` or `TYPE=REMOTE` in a CICS TST.

For each new DB2 unit of recovery, the appropriate plan selection exit routine is automatically invoked. This exit routine reads the temporary storage record and uses the contained plan name for plan selection.

When no temporary storage record exists for the Natural session, a default plan name, contained in the plan exit, can be used. If no plan name is specified by the exit, the name of the plan used is the same as the name of the static program (DBRM) issuing the SQL call. If no such plan name exists, an SQL error results.

Plan Switching under Com-plete

In Com-plete environments, plan switching is accomplished by using the Call Attachment Facility (CAF), which releases the thread in use and attaches another one that has a different plan name.

Once the DB2 connection has been established, the same plan name continues to be used until the plan is explicitly changed with IBM's call attachment language interface (`DSNALI`). For additional information on the CAF interface, refer to the relevant IBM literature.

Under Com-plete, the NATPLAN program first issues an `END TRANSACTION` statement and then invokes an Assembler routine by using `DB2SERV`.

The assembler routine performs the actual switching. It issues a `CLOSE` request to `DSNALI` to terminate the DB2 connection (if one exists). It then issues an `OPEN` request to re-establish the DB2 connection and to allocate the resources needed to execute the specified plan.

If `NATPLAN` has not been executed before the first SQL call, the default plan used is the one defined in the Com-plete startup parameters. Once a plan has been changed using `NDBPLAN`, it remains scheduled until another plan is scheduled by `NDBPLAN` or until the end of the Natural session.

Plan Switching under IMS TM

In an IMS MPP environment, the switch is accomplished by using direct or deferred message switching. As a different application plan is associated with each IMS application program, message switching from one transaction code to another automatically switches the application plan being used.

Since Natural applications can perform direct or deferred message switches by calling the appropriate supplied routines, use of the `NATPLAN` program for plan switching is optional.

`NATPLAN` calls the Assembler routine `CMDEFSW`, which sets the new transaction code to be used with the next following terminal I/O.

Plan Switching under TSO and in Batch Mode

In the TSO and batch environments, plan switching is accomplished by using the Call Attachment Facility (CAF) or the Resource Recovery Services Attachment Facility (RRSAF). Either facility releases the thread in use and attaches another one that has a different plan name.

Below is information on:

- Plan Selection with CAF
- Plan Selection with RRSAF

Plan Selection with CAF

Initial connection and plan setting can be done using the subparameters `DB2PLAN` and `DB2SSID` of the `NTDB2` macro or of the `DB2` profile parameter without using the `NATPLAN` program. However, the initial settings could be overwritten by using the `NATPLAN` program.

When using the Call Attachment Facility (CAF), plan selection is either implicit or explicit. If no DB2 connection has been made before the first SQL call, a plan name is selected by DB2. If so, the plan name used is the same as the name of the program (DBRM) issuing the SQL call.

Once the DB2 connection has been established, the plan name is retained until explicitly changed by IBM's call attachment language interface (`DSNALI`). For additional information on the CAF interface, refer to the relevant IBM literature.

Under TSO and in batch mode, the `NATPLAN` program first issues an `END TRANSACTION` statement and then invokes an Assembler routine by using `DB2SERV`.

> **Note:** Modify the `NATPLAN` program by setting the `#SSM` field to the current DB2 subsystem name; the default name is DB2.

The assembler routine performs the actual switching. It issues a `CLOSE` request to `DSNALI` to terminate a possible DB2 connection. It then issues an `OPEN` request to re-establish the DB2 connection and to allocate the resources needed to execute the specified plan.

If `NATPLAN` has not been executed before the first SQL call, plan selection is done by `DSNALI`. If so, the plan name used is the same as the name of the program issuing the SQL call. The subsystem ID used is the one specified during the DB2 installation. If no such plan name or subsystem ID exists, a Natural error message is returned.

If a static DBRM issues the SQL call, a plan name must exist with the same name as the one of the static DBRM.

If dynamic SQL is used, a DB2 plan must exist which contains a package with the DBRM of the `NDBIOMO` module. If the name of the DB2 plan has neither been defined in the `NATPLAN` program nor with the `DB2PLAN` keyword subparameter, the DB2 Call Attachment Facility (CAF) uses the name of the `NDBIOMO` DBRM as the default plan name.

> **Note:** To avoid any confusion concerning the chosen plan name and/or subsystem ID, always call `NATPLAN` before the first SQL call.

Plan Selection with RRSAF

Initial connection and plan setting can be done using the keyword subparameters `DB2COLL`, `DB2GROV`, `DB2PLAN`, `DB2SSID` and `DB2XID` of the `NTDB2` macro or of the `DB2` profile parameter without using the `NATPLAN` program. However, the initial settings can be overwritten by using the `NATPLAN` program.

When using the Resource Recovery Services Attachment Facility (RRSAF), plan selection is explicit.

RRSAF is used if IBM's `DSNRLI` interface module is linked to Natural. Once the DB2 connection has been established, the plan name is retained until explicitly changed with RRSAF. For additional information on RRSAF, refer to the relevant IBM literature.

The `NATPLAN` program performs the actual switching. It issues a `TERMINATE IDENTIFY` request to `DSNRLI` to terminate a possible DB2 connection. `NATPLAN` then issues an `IDENTIFY` request to re-establish the DB2 connection. This request is followed by `SIGNON` and `CREATE THREAD` requests.

In an RRSAF environment, up to four of the following parameters can be specified in `NATPLAN` where `#PLAN` is mandatory:

Parameter	Default Value	Format	Explanation
#PLAN	None	A8	Name of the plan used for SQL processing in the thread created (CREATE THREAD).
#SSM	DB2	A4	Subsystem ID of the DB2 server connected (IDENTIFY).
#COLLID	COLLID	A18	Only used if the first character of #PLAN is a question mark (?). Collection ID used for SQL processing in the thread created (CREATE THREAD).
#XID	1	I4	Indicates that a global transaction ID is used. If set to 0 (SIGNON), no global transaction ID is used.

Example of Plan Selection with RRSAF under TSO

The example below demonstrates plan selection under TSO by using RRSAF.

```
NATPLAN
<Enter>
Please enter new plan name   NDBPLAN4
          ,SUB SYSTEM ID DB27
          ,COLLECTION ID
       ,global XID (0/1) _____1
<Enter>
```

Example of Plan Selection with RRSAF in Batch Mode

The example below demonstrates plan selection in batch mode by using RRSAF:

```
NATPLAN NDBPLAN4,DB27, ,1
```

16   Using Natural Statements and System Variables

This section contains special considerations when using Natural native DML statements and Natural system variables with Natural SQL statements and DB2.

It mainly consists of information also contained in the Natural basic documentation set where each Natural statement and variable is described in detail.

For an explanation of the symbols used in this section to describe the syntax of Natural statements, see *Syntax Symbols* in the *Statements* documentation.

For information on logging SQL statements contained in a Natural program, refer to *DBLOG Trace Screen for SQL Statements* in the *DBLOG Utility* documentation.

DB2 Special Register Consideration

NDB refreshes the following DB2 special registers automatically to the values, which applied to the least previous executed transaction.

- CURRENT SQLID
- CURRENT SCHEMA
- CURRENT PATH
- CURRENT PACKAGE PATH

NDB refreshes the following DB2 special registers only automatically to the values, which applied to the least previous executed transaction, if the parameter REFRESH=ON is set.

- CURRENT PACKAGESET
- CURRENT SERVER

Those special registers are refreshed regardless whether the previously executed transaction was rolled back or was committed.

All other special registers are not implicitly manipulated by NDB.

Using Natural Native DML Statements

This section summarizes particular points you have to consider when using Natural data manipulation language (DML) statements with DB2. Any Natural statement not mentioned in this section can be used with DB2 without restriction.

Below is information on the following Natural DML statements:

- BACKOUT TRANSACTION

- DELETE
- END TRANSACTION
- FIND
- HISTOGRAM
- READ
- STORE
- UPDATE

BACKOUT TRANSACTION

The Natural native DML statement `BACKOUT TRANSACTION` undoes all database modifications made since the beginning of the last logical transaction. Logical transactions can start either after the beginning of a session or after the last `SYNCPOINT`, `END TRANSACTION`, or `BACKOUT TRANSACTION` statement.

How the statement is translated and which command is actually issued depends on the TP-monitor environment:

- If this command is executed within a Natural stored procedure or Natural user-defined function (UDF), Natural for DB2 executes the underlying rollback operation. This sets the caller into a must-rollback state. If this command is executed within a Natural stored procedure or UDF for Natural error processing (implicit `ROLLBACK`), Natural for DB2 does not execute the underlying rollback operation, thus allowing the caller to receive the original Natural error.

- Under CICS, the `BACKOUT TRANSACTION` statement is translated into an `EXEC CICS ROLLBACK` command. However, in pseudo-conversational mode, only changes made to the database since the last terminal I/O are undone. This is due to CICS-specific transaction processing, see *Natural for DB2 under CICS*.

  > **Note:** Be aware that with terminal input in database loops, Natural switches to conversational mode if no file server is used.

- In batch mode and under TSO, the `BACKOUT TRANSACTION` statement is translated into an SQL `ROLLBACK` command.

  > **Note:** If running in a DSNMTV01 environment, the `BACKOUT TRANSACTION` statement is ignored if the used PSB has been generated without the `CMPAT=YES` option.

- Under IMS TM, the `BACKOUT TRANSACTION` statement is translated into an IMS Rollback (`ROLB`) command. However, only changes made to the database since the last terminal I/O are undone. This is due to IMS TM-specific transaction processing, see *Natural for DB2 under IMS TM*.

As all cursors are closed when a logical unit of work ends, a `BACKOUT TRANSACTION` statement must not be placed within a database loop; instead, it has to be placed outside such a loop or after the outermost loop of nested loops.

If an external program written in another standard programming language is called from a Natural program, this external program must not contain its own ROLLBACK command if the Natural program issues database calls, too. The calling Natural program must issue the BACKOUT TRANSACTION statement for the external program.

If a program tries to backout updates which have already been committed, for example by a terminal I/O, a corresponding Natural error message (NAT3711) is returned.

DELETE

The Natural native DML statement DELETE is used to delete a row from a table which has been read with a preceding FIND, READ, or SELECT statement. It corresponds to the SQL statement DELETE WHERE CURRENT OF cursor-name, which means that only the row which was read last can be deleted.

Example:

```
FIND EMPLOYEES WITH NAME = 'SMITH'
     AND FIRST_NAME = 'ROGER'
DELETE
```

Natural would translate the above Natural statements into SQL and assign a cursor name (for example, CURSOR1) as follows:

```
DECLARE CURSOR1 CURSOR FOR
SELECT FROM EMPLOYEES
 WHERE NAME = 'SMITH' AND FIRST_NAME = 'ROGER' FOR UPDATE OF NAME
DELETE FROM EMPLOYEES
 WHERE CURRENT OF CURSOR1
```

Both the SELECT and the DELETE statement refer to the same cursor.

Natural translates a Natural native DML DELETE statement into a Natural SQL DELETE statement in the same way it translates a Natural native DML FIND statement into a Natural SQL SELECT statement.

A row read with a FIND SORTED BY cannot be deleted due to DB2 restrictions explained with the FIND statement. A row read with a READ LOGICAL cannot be deleted either.

DELETE when Using the File Server

If a row rolled out to the file server is to be deleted, Natural rereads automatically the original row from the database to compare it with its image stored in the file server. If the original row has not been modified in the meantime, the DELETE operation is performed. With the next terminal I/O, the transaction is terminated, and the row is deleted from the actual database.

If the DELETE operates on a scrollable cursor, the row on the file server is marked as DELETE hole and is deleted from the base table.

However, if any modification is detected, the row will not be deleted and Natural issues the NAT3703 error message for non-scrollable cursors.

If the `DELETE` operates on a scrollable cursor, Natural for DB2 simulates SQLCODE -224 THE RESULT TABLE DOES NOT AGREE WITH THE BASE TABLE USING for DB2 compliance.

If the `DELETE` operates on a scrollable cursor and the row has become a hole, Natural for DB2 simulates SQLCODE -222 AN UPDATE OR DELETE OPERATION WAS ATTEMPTED AGAINST A HOLE.

Since a `DELETE` statement requires that Natural rereads a single row, a unique index must be available for the respective table. All columns which comprise the unique index must be part of the corresponding Natural view.

END TRANSACTION

The Natural native DML statement `END TRANSACTION` indicates the end of a logical transaction and releases all DB2 data locked during the transaction. All data modifications are committed and made permanent.

How the statement is translated and which command is actually issued depends on the TP-monitor environment:

- If this command is executed from a Natural stored procedure or user defined function (UDF), Natural for DB2 does not execute the underlying commit operation. This allows the stored procedure or UDF to commit updates against non DB2 databases.

- Under CICS, the `END TRANSACTION` statement is translated into an `EXEC CICS SYNCPOINT` command. If the file server is used, an implicit end-of-transaction is issued after each terminal I/O. This is due to CICS-specific transaction processing in pseudo-conversational mode, see *Natural for DB2 under CICS*.

- In batch mode and under TSO, the `END TRANSACTION` statement is translated into an SQL `COMMIT WORK` command.

  > **Note:** If running in a DSNMTV01 environment the `END TRANSACTION` statement is ignored if the used PSB has been generated without the `CMPAT=YES` option.

- Under IMS TM, the `END TRANSACTION` statement is not translated into an IMS CHKP call, but is ignored. Due to IMS TM-specific transaction processing (see *Natural for DB2 under IMS TM*), an implicit end-of-transaction is issued after each terminal I/O.

Except when used in combination with the SQL `WITH HOLD` clause (see `SELECT - SQL` in *Using Natural SQL Statements*), an `END TRANSACTION` statement must not be placed within a database loop, since all cursors are closed when a logical unit of work ends. Instead, it has to be placed outside such a loop or after the outermost loop of nested loops.

If an external program written in another standard programming language is called from a Natural program, this external program must not contain its own `COMMIT` command if the Natural program issues database calls, too. The calling Natural program must issue the `END TRANSACTION` statement on behalf of the external program.

> **Note:** With DB2, the `END TRANSACTION` statement cannot be used to store transaction data.

FIND

The Natural native DML statement `FIND` corresponds to the Natural SQL statement `SELECT`.

Example:

Natural native DML statements:

```
FIND EMPLOYEES WITH NAME = 'BLACKMORE'
   AND AGE EQ 20 THRU 40
OBTAIN PERSONNEL_ID NAME AGE
```

Equivalent Natural SQL statement:

```
SELECT PERSONNEL_ID, NAME, AGE
  FROM EMPLOYEES
    WHERE NAME = 'BLACKMORE'
      AND AGE BETWEEN 20 AND 40
```

Natural internally translates a `FIND` statement into an SQL `SELECT` statement as described in *Processing of SQL Statements Issued by Natural* in the section *Internal Handling of Dynamic Statements*. The `SELECT` statement is executed by an `OPEN CURSOR` statement followed by a `FETCH` command. The `FETCH` command is executed repeatedly until either all records have been read or the program flow exits the `FIND` processing loop. A `CLOSE CURSOR` command ends the `SELECT` processing.

The `WITH` clause of a `FIND` statement is converted to the `WHERE` clause of the `SELECT` statement. The basic search criterion for a DB2 table can be specified in the same way as for an Adabas file. This implies that only database fields which are defined as descriptors can be used to construct basic search criteria and that descriptors cannot be compared with other fields of the Natural view (that is, database fields) but only with program variables or constants.

> **Note:** As each database field (column) of a DB2 table can be used for searching, any database field can be defined as a descriptor in a Natural DDM.

The `WHERE` clause of the `FIND` statement is evaluated by Natural *after* the rows have been selected via the `WITH` clause. Within the `WHERE` clause, non-descriptors can be used and database fields can be compared with other database fields.

> **Note:** DB2 does not have sub-, super-, or phonetic descriptors.

A `FIND NUMBER` statement is translated into a `SELECT` statement containing a `COUNT(*)` clause. The number of rows found is returned in the Natural system variable `*NUMBER` as described in the Natural *System Variables* documentation.

The `FIND UNIQUE` statement can be used to ensure that only one record is selected for processing. If the `FIND UNIQUE` statement is referenced by an `UPDATE` statement, a non-cursor (Searched) `UPDATE` operation is generated instead of a cursor-oriented (Positioned) `UPDATE` operation. Therefore, it can be used if you want to update a DB2 primary key. It is, however, recommended to use the Natural SQL Searched `UPDATE` statement to update a primary key.

In static mode, the `FIND NUMBER` and `FIND UNIQUE` statements are translated into a `SELECT SINGLE` statement as described in the section *Using Natural SQL Statements*.

The `FIND FIRST` statement cannot be used. The `PASSWORD`, `CIPHER`, `COUPLED` and `RETAIN` clauses cannot be used either.

The `SORTED BY` clause of a `FIND` statement is translated into the SQL `SELECT ... ORDER BY` clause, which follows the search criterion. Because this produces a read-only result table, a row read with a `FIND` statement that contains a `SORTED BY` clause cannot be updated or deleted.

A limit on the depth of nested database loops can be specified at installation time. If this limit is exceeded, a Natural error message is returned.

> **Notes:**

1. If a processing limit is specified as a constant integer number, for example, `FIND (5)`, the limitation value will be translated into a `FETCH FIRST` *integer* `ROWS ONLY` clause in the generated SQL string.

2. Natural for DB2 supports DB2 multiple row processing on behalf of the `MULTIFETCH` clause of the `FIND` statement.

FIND when using the File Server

As far as the file server is concerned, there are no programming restrictions with selection statements. It is, however, recommended to make yourself familiar with its functionality considering performance and file server space requirements.

HISTOGRAM

The Natural DML statement `HISTOGRAM` returns the number of rows in a table which have the same value in a specific column. The number of rows is returned in the Natural system variable `*NUMBER` as described in the Natural *System Variables* documentation.

Example:

Natural native DML statements:

```
HISTOGRAM EMPLOYEES FOR AGE
OBTAIN AGE
```

Equivalent Natural SQL statement:

```
SELECT COUNT(*), AGE FROM EMPLOYEES
  WHERE AGE > -999
  GROUP BY AGE
  ORDER BY AGE
```

Natural translates the `HISTOGRAM` statement into an SQL `SELECT` statement, which means that the control flow is similar to the flow explained for the `FIND` statement.

> **Note:** With Universal Database Server for z/OS Version 8, Natural for DB2 supports DB2 multiple row processing on behalf of the `MULTIFETCH` clause of the `HISTOGRAM` statement.

READ

The Natural native DML statement `READ` can also be used to access DB2 tables. Natural translates a `READ` statement into an SQL `SELECT` statement.

`READ PHYSICAL` and `READ LOGICAL` can be used; `READ BY ISN`, however, cannot be used, as there is no DB2 equivalent to Adabas ISNs. The `PASSWORD` and `CIPHER` clauses cannot be used either.

Since a `READ LOGICAL` statement is translated into a `SELECT ... ORDER BY` statement, which produces a read-only table, a row read with a `READ LOGICAL` statement cannot be updated or deleted (see Example 1). The start value can only be a constant or a program variable; any other field of the Natural view (that is, any database field) cannot be used.

A `READ PHYSICAL` statement is translated into a `SELECT` statement without an `ORDER BY` clause and can therefore be updated or deleted (see Example 2).

Example 1:

Natural native DML statements:

```
READ PERSONNEL BY NAME
OBTAIN NAME FIRSTNAME DATEOFBIRTH
```

Equivalent Natural SQL statement:

```
SELECT NAME, FIRSTNAME, DATEOFBIRTH FROM PERSONNEL
  WHERE NAME >= ' '
    ORDER BY NAME
```

Example 2:

The Natural native DML statements:

```
READ PERSONNEL PHYSICAL
OBTAIN NAME
```

Equivalent Natural SQL statement:

```
SELECT NAME FROM PERSONNEL
```

If the READ statement contains a WHERE clause, this clause is evaluated by the Natural processor *after* the rows have been selected according to the descriptor value(s) specified in the search criterion.

Processing Limit

If a processing limit is specified as a constant integer number, for example, READ (5), in the SQL string generated, the value that defines the limitation will be translated into the clause

```
FETCH FIRST integer ROWS ONLY
```

Cursors for DB2 Clauses

Natural for DB2 uses insensitive scrollable cursors to process the following READ statement:

```
READ .. [IN] [LOGICAL] VARIABLE/DYNAMIC operand5 [SEQUENCE]
```

Natural for DB2 uses insensitive scrollable cursors to process the READ statement below. If relating to a Positioned UPDATE or Positioned DELETE statement, Natural for DB2 uses insensitive static cursors.

```
READ .. [IN] [PHYSICAL] DESCENDING/VARIABLE/DYNAMIC operand5 [SEQUENCE]
```

operand5

Value `A` will be translated into a `FETCH FIRST/NEXT` DB2 access, and value `D` into a `FETCH LAST/PRIOR` DB2 access.

> **Note:** Natural for DB2 supports DB2 multiple row processing on behalf of the `MULTIFETCH` clause of the `READ` statement.

READ when Using the File Server

As far as the file server is concerned there are no programming restrictions with selection statements. It is, however, recommended to make yourself familiar with its functionality considering performance and file server space requirements.

STORE

The Natural native DML statement `STORE` is used to add a row to a DB2 table. The `STORE` statement corresponds to the SQL statement `INSERT`.

Example:

The Natural native DML statement:

```
STORE RECORD IN EMPLOYEES
  WITH PERSONNEL_ID = '2112'
       NAME        = 'LIFESON'
       FIRST_NAME  = 'ALEX'
```

Equivalent Natural SQL statement:

```
INSERT INTO EMPLOYEES (PERSONNEL_ID, NAME, FIRST_NAME)
  VALUES ('2112', 'LIFESON', 'ALEX')
```

The `PASSWORD`, `CIPHER` and `USING/GIVING NUMBER` clauses of the `STORE` statement cannot be used.

UPDATE

The Natural native DML statement UPDATE updates a row in a DB2 table which has been read with a preceding FIND, READ, or SELECT statement. It corresponds to the SQL statement UPDATE WHERE CURRENT OF *cursor-name* (Positioned UPDATE), which means that only the row which was read last can be updated.

UPDATE when Using the File Server

If a row rolled out to the file server is to be updated, Natural automatically rereads the original row from the database to compare it with its image stored in the file server. If the original row has not been modified in the meantime, the UPDATE operation is performed. With the next terminal I/O, the transaction is terminated and the row is definitely updated on the database.

If the UPDATE operates on a scrollable cursor, the row on the file server and the row in the base table are updated. If the row no longer qualifies for the search criteria of the related SELECT statement after the update, the row is marked as UPDATE hole on the file server.

However, if any modification is detected, the row will not be updated and Natural issues the NAT3703 error message for non-scrollable cursors.

If the UPDATE operates on a scrollable cursor, Natural for DB2 simulates SQLCODE -224 THE RESULT TABLE DOES NOT AGREE WITH THE BASE TABLE USING for DB2 compliance.

If the UPDATE operates on a scrollable cursor and the row has become a hole, Natural for DB2 simulates SQLCODE -222 AN UPDATE OR DELETE OPERATION WAS ATTEMPTED AGAINST A HOLE.

Since an UPDATE statement requires rereading a single row by Natural, a unique index must be available for this table. All columns which comprise the unique index must be part of the corresponding Natural view.

UPDATE with FIND/READ

As explained with the Natural native DML statement FIND, Natural translates a FIND statement into an SQL SELECT statement. When a Natural program contains a DML UPDATE statement, this statement is translated into an SQL UPDATE statement and a FOR UPDATE OF clause is added to the SELECT statement.

Example:

```
FIND EMPLOYEES WITH SALARY < 5000
  ASSIGN SALARY = 6000
  UPDATE
```

Natural would translate the above Natural statements into SQL and assign a cursor name (for example, CURSOR1) as follows:

```
DECLARE CURSOR1 CURSOR FOR
SELECT SALARY FROM EMPLOYEES WHERE SALARY < 5000
  FOR UPDATE OF SALARY
UPDATE EMPLOYEES SET SALARY = 6000
  WHERE CURRENT OF CURSOR1
```

Both the `SELECT` and the `UPDATE` statement refer to the same cursor.

Due to DB2 logic, a column (field) can only be updated if it is contained in the `FOR UPDATE OF` clause; otherwise updating this column (field) is rejected. Natural includes automatically all columns (fields) into the `FOR UPDATE OF` clause which have been modified anywhere in the Natural program or which are input fields as part of a Natural map.

However, an DB2 column is not updated if the column (field) is marked as "not updateable" in the Natural DDM. Such columns (fields) are removed from the `FOR UPDATE OF` list without any warning or error message. The columns (fields) contained in the `FOR UPDATE OF` list can be checked with the `LISTSQL` command.

The Adabas short name in the Natural DDM determines whether a column (field) can be updated.

The following table shows the ranges that apply:

Short-Name Range	Type of Field
AA - N9	non-key field that can be updated
Aa - Nz	non-key field that can be updated
OA - O9	primary key field
PA - P9	ascending key field that can be updated
QA - Q9	descending key field that can be updated
RA - X9	non-key field that cannot be updated
Ra - Xz	non-key field that cannot be updated
YA - Y9	ascending key field that cannot be updated
ZA - Z9	descending key field that cannot be updated
1A - 9Z	non-key field that cannot be updated
1a - 9z	non-key field that cannot be updated

Be aware that a primary key field is never part of a `FOR UPDATE OF` list. A primary key field can only be updated by using a non-cursor `UPDATE` operation (see also Natural SQL `UPDATE` statement in the section *Using Natural SQL Statements*).

A row read with a `FIND` statement that contains a `SORTED BY` clause cannot be updated (due to DB2 limitations as explained with the `FIND` statement). A row read with a `READ LOGICAL` statement cannot be updated either (as explained with the `READ` statement).

If a column is to be updated which is redefined as an array, it is strongly recommended to update the whole column and not individual occurrences; otherwise, results are not predictable. To do so, in reporting mode you can use the `OBTAIN` statement, which must be applied to all field occurrences in the column to be updated. In structured mode, however, all these occurrences must be defined in the corresponding Natural view.

The data locked by an `UPDATE` statement are released when an `END TRANSACTION` (`COMMIT WORK`) or `BACKOUT TRANSACTION` (`ROLLBACK WORK`) statement is executed by the program.

> **Note:** If a length indicator field or `NULL` indicator field is updated in a Natural program without updating the field (column) it refers to, the update of the column is not generated for DB2 and thus no updating takes place.

UPDATE with SELECT

In general, the Natural native DML statement `UPDATE` can be used in both structured and reporting mode. However, after a `SELECT` statement, only the syntax defined for Natural structured mode is allowed:

```
UPDATE [RECORD] [IN] [STATEMENT] [(r)]
```

This is due to the fact that in combination with the `SELECT` statement, the Natural native DML `UPDATE` statement is only allowed in the special case of:

```
...
SELECT ...
   INTO VIEW view-name
   ...
```

Thus, only a whole Natural view can be updated; individual columns (fields) cannot.

Example:

```
DEFINE DATA LOCAL
01 PERS VIEW OF SQL-PERSONNEL
  02 NAME
  02 AGE
END-DEFINE

SELECT *
  INTO VIEW PERS
  FROM SQL-PERSONNEL
  WHERE NAME LIKE 'S%'
```

```
    IF NAME = 'SMITH'
      ADD 1 TO AGE
    UPDATE
    END-IF

END-SELECT
    ...
```

In combination with the Natural native DML `UPDATE` statement, any other form of the `SELECT` statement is rejected and an error message is returned.

In all other respects, the Natural native DML `UPDATE` statement can be used with the `SELECT` statement in the same way as with the Natural `FIND` statement.

Using Natural SQL Statements

This section covers points you have to consider when using Natural SQL statements with DB2. These DB2-specific points partly consist in syntax enhancements which belong to the Extended Set of Natural SQL syntax. The Extended Set is provided in addition to the Common Set to support database-specific features; see *Common Set and Extended Set* in the *Statements* documentation.

For information on logging SQL statements contained in a Natural program, refer to *DBLOG Trace Screen for SQL Statements* in the *DBLOG Utility* documentation.

Below is information on the following Natural SQL statements and on common syntactical items:

- Syntactical Items Common to Natural SQL Statements
- CALLDBPROC - SQL
- COMMIT - SQL
- DELETE - SQL
- INSERT - SQL
- MERGE - SQL
- PROCESS SQL
- READ RESULT SET - SQL
- ROLLBACK - SQL
- SELECT - SQL

■ UPDATE - SQL

Syntactical Items Common to Natural SQL Statements

The following common syntactical items are either DB2-specific and do not conform to the standard SQL syntax definitions (that is, to the Common Set of Natural SQL syntax) or impose restrictions when used with DB2 (see also *Using Natural SQL Statements* in the *Statements* documentation).

Below is information on the following common syntactical items:

■ atom
■ comparison
■ factor
■ scalar-function
■ column-function
■ scalar-operator
■ special-register
■ units
■ case-expression

atom

An atom can be either a parameter (that is, a Natural program variable or host variable) or a constant. When running dynamically, however, the use of host variables is restricted by DB2. For further details, refer to the relevant DB2 literature by IBM.

comparison

The comparison operators specific to DB2 belong to the Natural Extended Set. For a description, refer to *Comparison Predicate* in *Search Conditions* in the *Statements* documentation.

factor

The following factors are specific to DB2 and belong to the Natural SQL Extended Set:

```
special-register
scalar-function(scalar-expression, ...)
scalar-expression unit
case-expression
```

scalar-function

A scalar function is a built-in function that can be used in the construction of scalar computational expressions. Scalar functions are specific to DB2 and belong to the Natural SQL Extended Set.

The scalar functions Natural for DB2 supports are listed below in alphabetical order:

A - H	I - R	S - Z
ABS	IDENTITY_VAL_LOCAL	SCORE
ABSVAL	IFNULL	SECOND
ACOS	INSERT	SIGN
ADD_MONTHS	INTEGER	SIN
ASIN	JULIAN_DAY	SINH
ASCII	LAST_DAY	SMALLINT
ASCII_CHR	LCASE	SOAPHTTPC
ASCII_STR	LEFT	SOAPHTTPV
ATAN	LENGTH	SOAPHTTPNC
ATAN2	LN	SOAPHTTPNV
ATANH	LOCATE	SOUNDEX
BIGINT	LOCATE_IN_STRING	SPACE
BINARY	LOG	SQRT
BLOB	LOG10	STRIP
CCSID_ENCODING	LOWER	SUBSTR
CEIL	LPAD	SUBSTRING
CEILING	LTRIM	TAN
CHAR	MAX	TANH
CHARACTER_LENGTH	MICROSECOND	TIME
CLOB	MIDNIGHT_SECONDS	TIMESTAMP
COALESCE	MIN	TIMESTAMPADD
COLLATION_KEY	MINUTE	TIMESTAMP_FORMAT
COMPARE_DECFLOAT	MOD	TIMESTAMP_ISO
CONCAT	MONTH	TIMESTAMP_TZ
CONTAINS	MONTHS_BETWEEN	TO_CHAR
COS	MQPUBLISH	TO_DATE
COSH	MQPUBLISHXML	TOTALORDER
DATE	MQREAD	TRANSLATE
DAY	MQREADCLOB	TRUNC
DAYOFMONTH	MQREADXML	TRUNC_TIMESTAMP
DAYOFWEEK	MQRECEIVE	TRUNCATE
DAYOFWEEK_ISO	MQRECEIVECLOB	UCASE
DAYOFYEAR	MQRECEIVEXML	UNICODE
DAYS	MQSEND	UNICODE_STR
DBCLOB	MQSENDXML	UNISTR
DEC	MQSENDXMLFILE	UPPER
DECFLOAT	MQSENDXMLFILECLOB	VALUE
DECFLOAT_SORTKEY	MQSUBSCRIBE	VARBINARY
DECIMAL	MQUNSUBSCRIBE	VARCHAR
DECRYPT_BIT	MULTIPLY_ALT	VARCHAR_FORMAT

A - H	I - R	S - Z
DECRYPT_CHAR	NEXT_DAY	VARGRAPHIC
DECRYPT_DB	NORMALIZE_DECFLOAT	WEEK
DEGREES	NORMALIZE_STRING	WEEK_ISO
DIFFERENCE	NULLIF	XMLATTRIBUTES
DIGITS	OVERLAY	XMLCONCAT
DOUBLE	POSSTR	XMLCOMMENT
DOUBLE_PRECISION	POWER	XMLDOCUMENT
DSN_XMLVALIDATE	QUANTIZE	XMLELEMENT
EBCDIC_CHR	QUARTER	XMLFOREST
EBCDIC_STR	RADIANS	XMLMODIFY
ENCRYPT_TDES	RAISE_ERROR	XMLNAMESPACES
ENCRYPT	RAND	XMLPARSE
EXP	REAL	XMLPI
EXTRACT	REPEAT	XMLQUERY
FLOAT	REPLACE	XMLSERIALIZE
FLOOR	RID	XMLTEXT
GRAPHIC	RIGHT	XMLXSROBJECTID
GENERATE_UNIQUE	ROUND	YEAR
GETHINT	ROUND_TIMESTAMP	
GETVARIABLE	ROWID	
HEX	RPAD	
HOUR	RTRIM	

Each scalar function is followed by one or more scalar expressions in parentheses. The number of scalar expressions depends upon the scalar function. Multiple scalar expressions must be separated from one another by commas.

Example:

```
SELECT NAME
  INTO NAME
  FROM SQL-PERSONNEL
  WHERE SUBSTR ( NAME, 1, 3 ) = 'Fri'
      ...
```

column-function

A column function returns a single-value result for the argument it receives. The argument is a set of like values, such as the values of a column. Column functions are also called aggregating functions.

The following column functions conform to standard SQL. They are not specific to DB2:

```
AVG
COUNT
MAX
MIN
```

```
SUM
```

The following column functions do not conform to standard SQL. They are specific to DB2 and belong to the Natural SQL Extended Set.

```
COUNT_BIG
CORRELATION
COVARIANCE
COVARIANCE_SAMP
STDDEV
STDDEV_POP
STDDEV_SAMP
VAR
VAR_POP
VAR_SAMP
VARIANCE
VARIANCE_SAMP
XMLAGG
```

scalar-operator

The concatenation operator (`CONCAT` or `||`) does not conform to standard SQL. It is specific to DB2 and belongs to the Natural Extended Set.

special-register

With the exception of `USER`, the following special registers do not conform to standard SQL. They are specific to DB2 and belong to the Natural SQL Extended Set:

```
CURRENT APPLICATION ENCODING SCHEME
CURRENT CLIENT_ACCNTG
CURRENT CLIENT_APPLNAME
CURRENT CLIENT_USERID
CURRENT CLIENT_WRKSTNNAME
CURRENT DATE
CURRENT_DATE
CURRENT DEBUG MODE
CURRENT DECFLOAT ROUNDING MODE
CURRENT DEGREE
CURRENT FUNCTION PATH
CURRENT_LC_CTYPE
CURRENT LC_CTYPE
CURRENT LOCALE LC_CTYPE
CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
CURRENT_MEMBER
CURRENT OPTIMIZATION HINT
```

```
CURRENT PACKAGE PATH
CURRENT PACKAGESET
CURRENT_PATH
CURRENT PRECISION
CURRENT REFRESH AGE
CURRENT ROUTINE VERSION
CURRENT RULES
CURRENT SCHEMA
CURRENT SERVER
CURRENT SQLID
CURRENT TIME
CURRENT_TIME
CURRENT TIMESTAMP
CURRENT TIMEZONE
CURRENT_TIMEZONE USER
SESSION TIME ZONE
SESSION_USER
USER
```

A reference to a special register returns a scalar value.

Using the command `SET CURRENT SQLID`, the creator name of a table can be substituted by the current SQLID. This enables you to access identical tables with the same table name but with different creator names.

units

Units, also called "durations", are specific to DB2 and belong to the Natural SQL Extended Set.

The following units are supported:

```
DAY
DAYS
HOUR
HOURS
MICROSECOND
MICROSECONDS
MINUTE
MINUTES
MONTH
MONTHS
SECOND
SECONDS
YEAR
YEARS
```

case-expression

```
      ⎧ searched-when-clause ⎫   ⎡      ⎧ NULL              ⎫ ⎤
CASE  ⎨ ...                  ⎬   ⎢ ELSE ⎨                   ⎬ ⎥  END
      ⎩ simple-when-clause   ⎭   ⎣      ⎩ scalar expression ⎭ ⎦
```

Case-expressions do not conform to standard SQL and are therefore supported by the Natural SQL Extended Set only.

Example:

```
DEFINE DATA LOCAL
   01 #EMP
   02 #EMPNO (A10)
   02 #FIRSTNME (A15)
   02 #MIDINIT  (A5)
   02 #LASTNAME (A15)
   02 #EDLEVEL    (A13)
   02 #INCOME (P7)
   END-DEFINE
  SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME,
        (CASE WHEN EDLEVEL < 15 THEN 'SECONDARY'
              WHEN EDLEVEL < 19 THEN 'COLLEGE'
              ELSE               'POST GRADUATE'
         END ) AS EDUCATION, SALARY + COMM AS INCOME
        INTO
        #EMPNO, #FIRSTNME, #MIDINIT, #LASTNAME,
        #EDLEVEL, #INCOME
          FROM DSN8510-EMP
          WHERE (CASE WHEN SALARY = 0 THEN NULL
                                      ELSE SALARY / COMM
                                      END ) > 0.25
  DISPLAY #EMP
  END-SELECT
  END
```

CALLDBPROC - SQL

The Natural SQL statement `CALLDBPROC` is used to call DB2 stored procedures. It supports the result set mechanism of DB2 and it enables you to call DB2 stored procedures. For further details and statement syntax, see *CALLDBPROC (SQL)* in the *Statements* documentation.

The following topics are covered below:

- Static and Dynamic Execution
- Result Sets
- List of Parameter Data Types
- CALLMODE=NATURAL

■ Example of CALLDBPROC/READ RESULT SET

Static and Dynamic Execution

If the `CALLDBPROC` statement is executed dynamically, all parameters and constants are mapped to the variables of the following DB2 SQL statement:

```
CALL :hv USING  DESCRIPTOR :sqlda statement
```

`:hv` denotes a host variable containing the name of the procedure to be called and `:sqlda` is a dynamically generated sqlda describing the parameters to be passed to the stored procedure.

If the `CALLDBPROC` statement is executed statically, the constants of the `CALLDBPROC` statement are also generated as constants in the generated assembler SQL source for the DB2 precompiler.

Result Sets

If the SQLCODE created by the `CALL` statement indicates that there are result sets (SQLCODE +466 and +464), Natural for DB2 runtime executes a `DESCRIBE PROCEDURE :hv INTO :sqlda` statement in order to retrieve the result set locator values of the result sets created by the invoked stored procedure. These values are put into the `RESULT SETS` variables specified in the `CALLDBPROC` statement. Each `RESULT SETS` variable specified in a `CALLDBPROC` for which no result set locator value is present is reset to zero. The result set locator values can be used to read the result sets by means of the `READ RESULT SET` statement as long as the database transaction which created the result set has not yet issued a `COMMIT` or `ROLLBACK`.

If the result set was created by a cursor `WITH HOLD`, the result set locator value remains valid after a `COMMIT` operation.

Unlike other Natural SQL statements, `CALLDBPROC` enables you (optionally!) to specify an SQLCODE variable following the `GIVING` keyword which will contain the SQLCODE of the underlying `CALL` statement. If `GIVING` is specified, it is up to the Natural program to react on the SQLCODE (error message NAT3700 is not issued by the runtime).

List of Parameter Data Types

Below are the parameter data types supported by the `CALLDBPROC` statement:

Natural Format/Length	DB2 Data Type
A*n*	CHAR(*n*)
B2	SMALLINT
B4	INT
B*n* (*n* = not equal 2 or 4)	CHAR(*n*)
F4	REAL

Natural Format/Length	DB2 Data Type
F8	`DOUBLE PRECISION`
I2	`SMALLINT`
I4	`INT`
N*nn.m*	`NUMERIC(nn+m,m)`
P*nn.m*	`NUMERIC(nn+m,n)`
G*n*	`GRAPHIC(n)`
A*n*/1:m	`VARCHAR(n*m)`
D	`DATE`
T	`TIME` **Note:** The Natural format T has a wider data range than the equivalent DB2 `TIME` data type. Compared with DB2 `TIME`, in addition, the Natural T variable has a date fraction (year, month, day) and the tenths of a second. As a result, when converting a Natural T variable into a DB2 `TIME` value, Natural for DB2 cuts off the date fraction and the tenths of a second part. When converting DB2 `TIME` into Natural T format, the date fraction is reset to `0000-01-02` and the tenths of a second part is reset to `0` in Natural.

CALLMODE=NATURAL

This parameter is used to invoke Natural stored procedures defined with `PARAMETER STYLE GENERAL/WITH NULL`.

If the `CALLMODE=NATURAL` parameter is specified, an additional parameter describing the parameters passed to the Natural stored procedure is passed from the client, that is, caller, to the server, that is, the Natural for DB2 server stub. The parameter is the Stored Procedure Control Block (STCB; see also *STCB Layout* in *PARAMETER STYLE* in the section *Processing Natural Stored Procedures and UDFs*) and has the format `VARCHAR` from the viewpoint of DB2. Therefore, every Natural stored procedure defined with `PARAMETER STYLE GENERAL/WITH NULL` has to be defined with the `CREATE PROCEDURE` statement by using this `VARCHAR` parameter as the first in its `PARMLIST` row.

From the viewpoint of the caller, that is, the Natural program, and from the viewpoint of the stored procedure, that is, Natural subprogram, the STCB is invisible. It is passed as first parameter by the Natural for DB2 runtime and it is used as on the server side to build the copy of the passed data in the Natural thread and the corresponding `CALLNAT` statement. Additionally, this parameter serves as a container for error information created during execution of the Natural stored procedure by the Natural runtime. It also contains information on the library where you are logged on and the Natural subprogram to be invoked.

Example of CALLDBPROC/READ RESULT SET

Below is a sample program for issuing CALLDBPROC and READ RESULT SET statements:

```
DEFINE DATA LOCAL
 1 ALPHA        (A8)
 1 NUMERIC      (N7.3)
 1 PACKED       (P9.4)
 1 VCHAR        (A20/1:5) INIT     <'DB25SGCP'>
 1 INTEGER2     (I2)
 1 INTEGER4     (I4)
 1 BINARY2      (B2)
 1 BINARY4      (B4)
 1 BINARY12     (B12)
 1 FLOAT4       (F4)
 1 FLOAT8       (F8)
 1 INDEX-ARRAY (I2/1:11)
 1 INDEX-ARRAY1(I2)
 1 INDEX-ARRAY2(I2)
 1 INDEX-ARRAY3(I2)
 1 INDEX-ARRAY4(I2)
 1 INDEX-ARRAY5(I2)
 1 INDEX-ARRAY6(I2)
 1 INDEX-ARRAY7(I2)
 1 INDEX-ARRAY8(I2)
 1 INDEX-ARRAY9(I2)
 1 INDEX-ARRAY10(I2)
 1 INDEX-ARRAY11(I2)
 1 #RESP        (I4)
 1 #RS1         (I4) INIT <99>
 1 #RS2         (I4) INIT <99>
 LOCAL
 1 V1 VIEW OF SYSIBM-SYSTABLES
 2 NAME
 1 V2 VIEW OF SYSIBM-SYSPROCEDURES
 2 PROCEDURE
 2 RESULT_SETS
 1 V (I2) INIT <99>
 END-DEFINE
 CALLDBPROC 'DAEFDB25.SYSPROC.SNGSTPC'  DSN8510-EMP
  ALPHA   INDICATOR :INDEX-ARRAY1
  NUMERIC INDICATOR :INDEX-ARRAY2
  PACKED  INDICATOR :INDEX-ARRAY3
  VCHAR(*) INDICATOR :INDEX-ARRAY4
  INTEGER2 INDICATOR :INDEX-ARRAY5
  INTEGER4 INDICATOR :INDEX-ARRAY6
  BINARY2  INDICATOR :INDEX-ARRAY7
  BINARY4  INDICATOR :INDEX-ARRAY8
  BINARY12 INDICATOR :INDEX-ARRAY9
  FLOAT4   INDICATOR :INDEX-ARRAY10
  FLOAT8   INDICATOR :INDEX-ARRAY11
```

```
   RESULT SETS #RS1 #RS2
  CALLMODE=NATURAL
 READ (10) RESULT SET #RS2 INTO VIEW V2 FROM SYSIBM-SYSTABLES
 WRITE 'PROC F RS  :' PROCEDURE 50T RESULT_SETS
 END-RESULT
 END
```

COMMIT - SQL

The Natural SQL COMMIT statement indicates the end of a logical transaction and releases all DB2 data locked during the transaction. All data modifications are made permanent. For further details and statement syntax, see *COMMIT (SQL)* in the *Statements* documentation.

COMMIT is a synonym for the Natural native DML statement END TRANSACTION as described in the section *Using Natural Native DML Statements*.

No transaction data can be provided with the COMMIT statement.

If this command is executed from a Natural stored procedure or user-defined function (UDF), Natural for DB2 does not execute the underlying commit operation. This allows the Natural stored procedure or UDF to commit updates against non DB2 databases.

Under CICS, the COMMIT statement is translated into an EXEC CICS SYNCPOINT command. If the file server is used, an implicit end-of-transaction is issued after each terminal I/O. This is due to CICS-specific transaction processing in pseudo-conversational mode, see *Natural for DB2 under CICS*.

Under IMS TM, the COMMIT statement is not translated into an IMS CHECKPOINT command, but is ignored. An implicit end-of-transaction is issued after each terminal I/O. This is due to IMS TM-specific transaction processing, see *Natural for DB2 under IMS TM*.

Unless when used in combination with the WITH HOLD clause (see *Syntax 1 - Cursor-Oriented Selection* in *SELECT (SQL)* in the *Statements* documentation), a COMMIT statement must not be placed within a database loop, since all cursors are closed when a logical unit of work ends. Instead, it has to be placed outside such a loop or after the outermost loop of nested loops.

If an external program written in another standard programming language is called from a Natural program, this external program must not contain its own COMMIT command if the Natural program issues database calls, too. The calling Natural program must issue the COMMIT statement on behalf of the external program.

DELETE - SQL

Both the cursor-oriented or Positioned DELETE, and the non-cursor or Searched DELETE statements are supported as part of Natural SQL; the functionality of the Positioned DELETE statement corresponds to that of the Natural DML DELETE statement. For further details and statement syntax, see *DELETE (SQL)* in the *Statements* documentation.

With DB2, a table name in the *FROM Clause* of a Searched DELETE statement can be assigned a *correlation-name*. This does not correspond to the standard SQL syntax definition and therefore belongs to the Natural SQL Extended Set.

The Searched DELETE statement must be used, for example, to delete a row from a self-referencing table, since with self-referencing tables a Positioned DELETE is not allowed by DB2.

INSERT - SQL

The Natural SQL INSERT statement is used to add one or more new rows to a table.

Since the INSERT statement can contain a select expression, all the DB2-specific **common syntactical items** described above apply.

For further details and statement syntax, see *INSERT (SQL)* in the *Statements* documentation.

MERGE - SQL

The MERGE statement is a hybrid SQL statement consisting of an UPDATE component and an INSERT component. It allows you either to insert a row into a DB2 table or to update a row of a DB2 table if the input data matches an already existing row of a table.

The MERGE statement belongs to the SQL Extended Set.

For further details and statement syntax, see *MERGE (SQL)* in the *Statements* documentation.

PROCESS SQL

The Natural PROCESS SQL statement is used to issue SQL statements to the underlying database. The statements are specified in a *statement-string*, which can also include constants and parameters. The set of statements which can be issued is also referred to as Flexible SQL and comprises those statements which can be issued with the SQL statement EXECUTE.

In addition, Flexible SQL includes the following DB2-specific statements:

```
CALL
CONNECT
GET DIAGNOSTICS
SET APPLICATION ENCODING SCHEME
SET CONNECTION
```

```
SET CURRENT DEGREE
SET CURRENT LC_CTYPE
SET CURRENT OPTIMIZATION HINT
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
SET CURRENT PACKAGE PATH
SET CURRENT PACKAGESET
SET CURRENT PATH
SET CURRENT PRECISION
SET CURRENT REFRESH AGE
SET CURRENT RULES
SET CURRENT SCHEMA
SET CURRENT SQLID
SET ENCRYPTION PASSWORD
SET host-variable=special-register
RELEASE
```

> **Notes:**

1. SQL statements issued by `PROCESS SQL` are skipped during static generation. Thus they are always executed dynamically via `NDBIOMO`.

2. To avoid transaction synchronization problems between the Natural environment and DB2, the `COMMIT` and `ROLLBACK` statements must not be used within `PROCESS SQL`.

For further details and statement syntax, see *PROCESS SQL* in the *Statements* documentation.

READ RESULT SET - SQL

The Natural SQL `READ RESULT SET` statement reads a result set created by a Natural stored procedure that was invoked by a `CALLDBPROC` statement. For details on how to specify the scroll direction by using the variable *scroll-hv*, see the `SELECT` statement.

For further details and statement syntax, see *READ RESULT SET (SQL)* in the *Statements* documentation.

ROLLBACK - SQL

The Natural SQL `ROLLBACK` statement undoes all database modifications made since the beginning of the last logical transaction. Logical transactions can start either after the beginning of a session or after the last `COMMIT/END TRANSACTION` or `ROLLBACK/BACKOUT TRANSACTION` statement. All records held during the transaction are released.

For further details and statement syntax, see *ROLLBACK (SQL)* in the *Statements* documentation.

`ROLLBACK` is a synonym for the Natural statement `BACKOUT TRANSACTION` as described in the section *Using Natural Native DML Statements*.

If this command is executed from a Natural stored procedure or user-defined function (UDF), Natural for DB2 executes the underlying rollback operation. This sets the caller into a must-rollback state. If this command is executed by Natural error processing (implicit `ROLLBACK`), Natural for DB2 does not execute the underlying rollback operation, thus allowing the caller to receive the original Natural error.

Under CICS, the `ROLLBACK` statement is translated into an `EXEC CICS ROLLBACK` command. However, if the file server is used, only changes made to the database since the last terminal I/O are undone. This is due to CICS-specific transaction processing in pseudo-conversational mode, see *Natural for DB2 under CICS*.

Under IMS TM, the `ROLLBACK` statement is translated into an IMS Rollback (`ROLB`) command. However, only changes made to the database since the last terminal I/O are undone. This is due to IMS TM-specific transaction processing, see *Natural for DB2 under IMS TM*.

As all cursors are closed when a logical unit of work ends, a `ROLLBACK` statement must not be placed within a database loop; instead, it has to be placed outside such a loop or after the outermost loop of nested loops.

If an external program written in another standard programming language is called from a Natural program, this external program must not contain its own `ROLLBACK` command if the Natural program issues database calls, too. The calling Natural program must issue the `ROLLBACK` statement on behalf of the external program.

SELECT - SQL

The Natural SQL `SELECT` statement supports both the cursor-oriented selection, which is used to retrieve an arbitrary number of rows and the non-cursor selection (Singleton `SELECT`), which retrieves at most one single row.

For full details and statement syntax, see *SELECT (SQL)* in the *Statements* documentation.

SELECT - Cursor-Oriented

Like the Natural native DML `FIND` statement, the cursor-oriented `SELECT` statement is used to select a set of rows (records) from one or more DB2 tables, based on a search criterion. Since a database loop is initiated, the loop must be closed by a `LOOP` statement (in reporting mode) or by an `END-SELECT` statement (in structured mode). With this construction, Natural uses the same loop processing as with the `FIND` statement. In addition, no cursor management is required from the application program; it is automatically handled by Natural.

For further details and syntax, see *Syntax 1 - Cursor-Oriented Selection* in *SELECT (SQL)* in the *Statements* documentation.

SELECT SINGLE - Non-Cursor-Oriented

The Natural SQL statement `SELECT SINGLE` provides the functionality of a non-cursor selection (Singleton `SELECT`); that is, a select expression that retrieves at most one row without using a cursor.

Since DB2 supports the Singleton `SELECT` command in static SQL only, in dynamic mode, the Natural `SELECT SINGLE` statement is executed in the same way as a set-level `SELECT` statement, which results in a cursor operation. However, Natural checks the number of rows returned by DB2. If more than one row is selected, a corresponding error message is returned.

For further details and syntax, see *Syntax 2 - Non-Cursor Selection* in *SELECT (SQL)* in the *Statements* documentation.

UPDATE - SQL

Both the cursor-oriented or Positioned `UPDATE` and the non-cursor or Searched `UPDATE` statements are supported as part of Natural SQL. Both of them reference either a table or a Natural view.

With DB2, the name of a table or Natural view to be referenced by a Searched `UPDATE` can be assigned a `correlation-name`. This does not correspond to the standard SQL syntax definition and therefore belongs to the Natural Extended Set.

The Searched `UPDATE` statement must be used, for example, to update a primary key field, since DB2 does not allow updating of columns of a primary key by using a Positioned `UPDATE` statement.

> **Note:** If you use the `SET *` notation, all fields of the referenced Natural view are added to the `FOR UPDATE OF` and `SET` lists. Therefore, ensure that your view contains only fields which can be updated; otherwise, a negative SQLCODE is returned by DB2.

For further details and syntax, see *UPDATE (SQL)* in the *Statements* documentation.

Using Natural System Variables

When used with DB2, there are restrictions and/or special considerations concerning the following Natural system variables:

- `*ISN`
- `*NUMBER`
- `*ROWCOUNT`

For information on restrictions and/or special considerations, refer to the section *Database-Specific Information* in the corresponding system variable documentation.

Multiple Row Processing

This section describes the multiple row functionality for DB2 databases.

You have to operate against DB2 for z/OS Version 8 or higher to use these features.

Natural for DB2 provides two kinds of multiple row processing features:

■ **Standard multiple row processing**

■ This feature does not influence the program logic. Although the Natural native DML and Natural SQL DML provide clauses for specification of the multi-fetch-factor, the Natural program operates with one database row and from the program point of view only one row is received from or is send to the database.

■ **Advanced multiple row processing**

This feature is only available with Natural SQL DML and has a lot of impact on the program logic, as it allows the retrieval of multiple rows from the database into the program storage by a single Natural SQL `SELECT` statement into a set of arrays. Additionally it is possible to insert multiple rows into the database from a set of arrays by the Natural SQL `INSERT` statement.

Below is information on the following topics:

- Purpose of Multi-Fetch Feature (Standard)
- Considerations for Multi-Fetch Usage (Standard)
- Size of the Multi-Fetch Buffer (Standard)
- Support of TEST DBLOG Q (Standard)
- Multiple Rows to Program (Advanced)
- Multiple Rows from Program (Advanced)

Purpose of Multi-Fetch Feature (Standard)

In standard mode, Natural does not read multiple records with a single database call; it always operates in a one-record-per-fetch mode. This kind of operation is solid and stable, but can take some time if a large number of database records are being processed.

To improve the performance of those programs, you can use the Multi-Fetch Clause in the Natural DML `FIND`, `READ` or `HISTOGRAM` statements. This allows you to specify the number of records read per database access.

```
┌                 ┐ ┌                 ┌                        ┐  ┐
│ FIND            │ │                 │ ON                     │  │
│ READ            │ │ MULTI-FETCH     │ OFF                    │  │
│ HISTOGRAM       │ │                 │ OF multi-fetch-factor  │  │
└                 ┘ └                 └                        ┘  ┘
```

Where the *multi-fetch-factor* is either a constant or a variable with a format integer (I4).

To improve the performance of the Natural SQL SELECT statements, you can use the WITH ROWSET POSITIONING FOR Clause to specify a multi-fetch-factor.

```
┌                                    ┌            ┐         ┐
│                                    │ [:] row_hv │         │
│  WITH ROWSET POSITIONING FOR       │            │  ROWS   │
│                                    │ integer    │         │
└                                    └            ┘         ┘
```

At statement execution time, the runtime checks if a *multi-fetch-factor* greater than 1 is supplied for the database statement.

If the *multi-fetch-factor* is

less than or equal to 1	the database call is continued in the usual one-record-per-access mode.
greater than 1	the database call is prepared dynamically to read multiple records (e.g. 10) with a single database access into an auxiliary buffer (multi-fetch buffer). If successful, the first record is transferred into the underlying data view. Upon the execution of the next loop, the data view is filled directly from the multi-fetch buffer, without database access. After all records are fetched from the multi-fetch buffer, the next loop results in the next record set being read from the database. If the database loop is terminated (either by end-of-records, ESCAPE, STOP, etc.), the content of the multi-fetch buffer is released.

Considerations for Multi-Fetch Usage (Standard)

- The program does not receive "fresh" records from the database for every loop, but operates with images retrieved at the most recent multi-fetch access.

- If a dynamic direction change (IN DYNAMIC...SEQUENCE) is coded for a Natural DML READ or HISTOGRAM statement, the multi-fetch feature is not possible and leads to a corresponding syntax error at compilation.

- The size occupied by a database loop in the multi-fetch buffer is determined according to the rule:

header + sqldaheader + columns*(sqlvar+lise) + mf*(udind + sum(collen) + sum(LF(columns) + sum(nullind))

=

32 + 16 + columns*(44+12) + mf*(1 + sum(collen) + sum(LF(column)) + sum(2))

where

- header denotes the length of the header of a entry in the DB2 multifetch buffer, that is, 32

- sqldaheader denotes the length of the header of a sqlda, that is, 16

- columns denotes the number of receiving fields of a SQL request

- sqlvar denotes the length of a sqlvar, that is, 44

- lise denotes the length of a Natrual for DB2 specific sqlvar extension

- mf denotes the multifetch factor, that is, the number of rows fetched by one database call

- collen denotes the length of the receiving field

- LF(column) denotes the size of the length field of the receiving field, that is, 0 for fixed length fields, 2 for variable length fields, and 4 for large object columns (LOBs)

- nullind denotes the length of a null indicator, that is, 2

Size of the Multi-Fetch Buffer (Standard)

The multifetch buffer is released at terminal i/o in pseudo conversional mode. Therefore there is no size limitation for the DB2 multifetch buffer (DB2SIZE6). The buffer will be automatical enlarged if necessary.

Support of TEST DBLOG Q (Standard)

When multi-fetch is used, real database calls are only submitted to get a new set of records.

The TEST DBLOG Q facility is also called from the Natural for DB2 multi fetch handler for every rowset fetch from DB2 and for every record moved from the multi fetch buffer to the program storage. The events are distinguished by the literal MULTI FETCH ... and <BUFF FETCH ...

Example: TEST DBLOG List Break-Out

```
10:51:57                 ***** NATURAL Test Utilities *****              2006-01-27
User HGK                        - DBLOG Trace -                    Library NDB42
M No   R SQL Statement (truncated)      CU SN SREF M Typ SQLC/W Program  Line LV
_    1   SELECT EMPNO,FIRSTNME,LASTNAM 01 01 0260 D DB2        MF000001 0260 01
_    2     MULTI FETCH  NEX            01 01 0260 D DB2        MF000001 0260 01
_    3     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_    4     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_    5     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_    6     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_    7     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_    8     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_    9     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_   10     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_   11     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_   12     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_   13     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_   14     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_   15     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_   16     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
_   17     <BUFF FETCH  NEX            00 00 0260 D DB2        MF000001 0260 01
Command ===>
```

where column **No** represents the following:

1	is a open cursor DB2 call.
2	is a "real" database call that reads a set of records via multi-fetch (see `MULTI FETCH NEX` in column **SQL Statement**).
3-17	are "no real" database calls, but only entries that document that the program has received these records from the multi-fetch buffer (see `<BUFF FETCH NEX` in column **SQL Statement**).

Multiple Rows to Program (Advanced)

The feature allows programs to retrieve multiple rows from DB2 into arrays.

This feature is only available with the `SELECT` statement.

- Prerequisites
- DB2ARRY=ON
- INTO Clause
- WITH ROWSET POSITIONING Clause
- ROWS_RETURNED Clause
- Restrictions and Constraints

- File Server Usage and Positioned UPDATE and DELETE

Prerequisites

≫ **To use this feature**

1   Set the compiler option `DB2ARRY=ON` (by using an `OPTIONS` statement or the `COMPOPT` command or the `CMPO` profile parameter).

2   Specify a list of receiving arrays in the `INTO` clause (see *into-clause*) of the `SELECT` statement.

3   Specify the number of rows to be retrieved from the database by a single `FETCH` operation via the `WITH ROWSET POSITIONING` Clause.

4   Specify a variable receiving the number of rows retrieved from the database via the `ROWS_RETURNED` Clause.

DB2ARRY=ON

`DB2ARRY=ON` is necessary to allow the specification of arrays in the `INTO` clause (see *into-clause*). `DB2ARRY=ON` also prevents the usage of arrays as sending or receiving fields for DB2 `CHAR`/`VARCHAR` /`GRAPHIC`/`VARGRAPHIC` columns. Instead Natural scalar fields with the appropriate length have to be used.

INTO Clause

Each array specified in the `INTO` clause (see *into-clause*) has to be contiguous (one occurrence following immediately by another, this is expected by DB2) and has to be one-dimensional. The arrays are filled from the first occurrence (low) to last occurrence (high). The first array occurrences compose the first row of the received rowset, the second array occurrences compose the second row of the received rowset. The array occurrences of the nth index compose the nth row returned from DB2. If an *LINDICATOR Clause* or *INDICATOR Clause* is used in the `INTO` clause for arrays, the specified length indicators or null indicators have also to be arrays. The number of occurrences of `LINDICATOR` and `INDICATOR` arrays have to equal or greater than the number of occurrences of the master array.

WITH ROWSET POSITIONING Clause

The `WITH_ROWSET_POSITIONING` Clause is used to specify the number of rows to be retrieved from the database by one processing cycle. The specified number has to be equal or smaller than the minimum of occurrences of all specified arrays. If a variable, not a constant, is specified the actual content of the variable will be used during each processing cycle. The specified number has to be greater `0` and smaller than `32768`.

ROWS_RETURNED Clause

The `ROWS_RETURNED` Clause is used to specify a variable, which will contain the number of rows read from the database during the actual fetch operation. The format of the variable has to be I4.

Restrictions and Constraints

Natural Views: It is not possible to use Natural arrays of views in the `INTO` clause (see *into-clause*), that is, the use of keyword `VIEW` is not possible.

File Server Usage and Positioned UPDATE and DELETE

The purpose of this feature is to reduce the number of database and database interface calls for bulk batch processing. Therefore it is not recommended to use this kind of programming in online CICS or IMS environments, when terminal I/Os occur within open cursor loops; that is, the file server is used. A fortiori it is not possible to perform a Positioned `UPDATE` or Positioned `DELETE` statement after terminal I/O.

Example:

```
DEFINE DATA LOCAL
01 NAME            (A20/1:10)
01 ADDRESS         (A100/1:10)
01 DATEOFBIRTH     (A10/1:10)
01 SALARY          (P4.2/1:10)
01 L$ADDRESS       (I2/1:10)
01 ROWS            (I4)
01 NUMBER          (I4)
01 INDEX           (I4)
END-DEFINE
OPTIONS DB2ARRY=ON
ASSIGN NUMBER := 10
SEL.
SELECT NAME, ADDRESS , DATEOFBIRTH, SALARY
      INTO  :NAME(*),                            /* <-- ARRAY
            :ADDRESS(*) LINDICATOR :L$ADDRESS(*), /* <-- ARRAY
            :DATEOFBIRTH(1:10),                  /* <-- ARRAY
            :SALARY(01:10)                       /* <-- ARRAY
    FROM NAT-DEMO
    WHERE NAME > ' '
    WITH ROWSET POSITIONING FOR :NUMBER ROWS    /* <-- ROWS REQ
    ROWS_RETURNED :ROWS                         /* <-- ROWS RET
  IF ROWS > 0
   FOR INDEX = 1 TO  ROWS STEP 1
    DISPLAY
            INDEX (EM=99) *COUNTER (SEL.) (EM=99) ROWS (EM=99)
            NAME(INDEX)
            ADDRESS(INDEX) (AL=20)
            DATEOFBIRTH(INDEX)
```

```
            SALARY(INDEX)
    END-FOR
  END-IF
END-SELECT
END
```

Multiple Rows from Program (Advanced)

The feature allows programs to insert multiple rows into a DB2 table from arrays.

This feature is only available with the Natural SQL `INSERT` statement.

Prerequisites

≫ To use this feature

1   Set the compiler option `DB2ARRY=ON` (by using an `OPTIONS` statement or the `COMPOPT` command or the `CMPO` profile parameter).

2   Specify a list of sending arrays in the *VALUES Clause* of the Natural SQL `INSERT` statement.

3   Specify the number of rows to be inserted into the database by a single Natural SQL `INSERT` statement via the *FOR n ROWS Clause*.

DB2ARRY=ON

`DB2ARRY=ON` is necessary to allow the specification of arrays in the *VALUES Clause*. `DB2ARRY=ON` also prevents the usage of arrays as sending or receiving fields for DB2 `CHAR`/`VARCHAR` /`GRAPHIC`/`VARGRAPHIC` columns. Instead Natural scalar fields with the appropriate length have to be used.

VALUES Clause

Each array specified in the *VALUES Clause* has to be contiguous (one occurrence following immediately by another, this is expected by DB2) and has to be one-dimensional. The arrays are read from the first occurrence (low) to last occurrence (high). The first array occurrences compose the first row inserted into the database, the second array occurrences compose the second row inserted into the database. The array occurrences of the nth index compose the nth row inserted into the database. If an *LINDICATOR Clause* or *INDICATOR Clause* is used in the *VALUES Clause* for arrays, the specified length indicators or null indicators have also to be arrays. The number of `LINDICATOR` and `INDICATOR` array occurrences has to be equal or greater than the number of occurrences of the master array.

FOR n ROWS Clause

The *FOR n ROWS Clause* is used to specify how many rows are to be inserted into the database table by one `INSERT` statement. The specified number has to be equal or smaller than the minimum of occurrences of all specified arrays in the *VALUES Clause*. The specified number has to be greater than `0` and smaller than `32768`.

Restrictions and Constraints

■ **Natural Views**

It is not possible to use Natural arrays of views in the *VALUES Clause*, that is, the use of keyword `VIEW` is not possible.

■ **Static Execution**

Due to DB2 restrictions it is not possible to execute multiple row inserts in static mode. Therefore, multiple row inserts are not generated static and are always dynamically prepared and executed by Natural for DB2. The Natural for DB2 static generation creates an assembler language SQL program, which is precompiled by the DB2 precompiler, which in turn creates a DBRM necessary for static execution. However, the DB2 assembler precompiler has no support for host variable arrays. They can only be used when specified in a SQLDA structure, which has to be build at execution time. But a static `INSERT` with a multiple-row-insert `VALUES` clause does not allow the specification of an SQLDA, but only host-variable arrays, which are not supported by the DB2 assembler precompiler.

It is not possible to use Natural arrays of views in the `INTO` clause (see *into-clause*), that is, the use of keyword `VIEW` is not possible.

Example:

```
DEFINE DATA LOCAL
01 NAME        (A20/1:10)  INIT <'ZILLER1','ZILLER2','ZILLER3','ZILLER4'
                                ,'ZILLER5','ZILLER6','ZILLER7','ZILLER8'
                                ,'ZILLER9','ZILLERA'>
01 ADDRESS     (A100/1:10) INIT <'ANGEL STREET 1','ANGEL STREET 2'
                                ,'ANGEL STREET 3','ANGEL STREET 4'
                                ,'ANGEL STREET 5','ANGEL STREET 6'
                                ,'ANGEL STREET 7','ANGEL STREET 8'
                                ,'ANGEL STREET 9','ANGEL STREET 10'>
01 DATENATD (D/1:10)  INIT <D'1954-03-27',D'1954-03-27',D'1954-03-27'
                           ,D'1954-03-27',D'1954-03-27',D'1954-03-27'
                           ,D'1954-03-27',D'1954-03-27',D'1954-03-27'
                           ,D'1954-03-27'>
01 SALARY      (P4.2/1:10) INIT <1000,2000,3000,4000,5000
                                ,6000,7000,8000,9000,9999>
01 SALARY_N    (N4.2/1:10) INIT <1000,2000,3000,4000,5000
                                ,6000,7000,8000,9000,9999>
01 L§ADDRESS   (I2/1:10) INIT <14,14,14,14,14,14,14,14,14,15>
```

```
01 N§ADDRESS    (I2/1:10) INIT <00,00,00,00,00,00,00,00,00,00>
01 ROWS        (I4)
01 INDEX       (I4)
01 V1 VIEW OF NAT-DEMO_ID
02 NAME
02 ADDRESS     (EM=X(20))
02 DATEOFBIRTH
02 SALARY
01 ROWCOUNT  (I4)
END-DEFINE
OPTIONS DB2ARRY=ON                  /* <-- ENABLE DB2 ARRAY
ROWCOUNT := 10
INSERT INTO NAT-DEMO_ID
      (NAME,ADDRESS,DATEOFBIRTH,SALARY)
      VALUES
      (:NAME(*),                    /* <-- ARRAY
       :ADDRESS(*)                  /* <-- ARRAY
       INDICATOR :N§ADDRESS(*)      /* <-- ARRAY
       LINDICATOR :L§ADDRESS(*),    /* <-- ARRAY DB2 VCHAR
       :DATENATD(1:10),             /* <-- ARRAY NATURAL DATES
       :SALARY_N(01:10)             /* <-- ARRAY NATURAL NUMERIC
      )
      FOR :ROWCOUNT ROWS
SELECT * INTO VIEW V1 FROM NAT-DEMO_ID WHERE NAME > 'Z'
DISPLAY V1                          /* <-- VERIFY INSERT
END-SELECT
BACKOUT
END
```

Error Handling

In contrast to the normal Natural error handling, where either an `ON ERROR` statement is used to intercept execution time errors or standard error message processing is performed and program execution is terminated, the enhanced error handling of Natural for DB2 provides an application controlled reaction to the encountered SQL error.

Two Natural subprograms, `NDBERR` and `NDBNOERR`, are provided to disable the usual Natural error handling and to check the encountered SQL error for the returned SQLCODE. This functionality replaces the `E` function of the `DB2SERV` interface, which is still provided but no longer documented.

For further information on Natural subprograms provided for DB2, see the section *Interface Subprograms*.

Example:

```
DEFINE DATA LOCAL
 01 #SQLCODE            (I4)
 01 #SQLSTATE           (A5)
 01 #SQLCA              (A136)
 01 #DBMS               (B1)
 END-DEFINE
 *
 *        Ignore error from next statement
 *
 CALLNAT 'NDBNOERR'
 *
 *        This SQL statement produces an SQL error
 *
 INSERT INTO SYSIBH-SYSTABLES (CREATOR, NAME, COLCOUNT)
   VALUES ('SAG', 'MYTABLE', '3')
 *
 *        Investigate error
 *
 CALLNAT 'NDBERR' #SQLCODE #SQLSTATE #SQLCA #DBMS
 *
 IF #DBMS NE 2                              /* not DB2
   MOVE 3700 TO *ERROR-NR
 END-IF
 *
 DECIDE ON FIRST VALUE OF #SQLCODE
   VALUE 0, 100                            /* successful execution
     IGNORE
   VALUE -803                              /* duplicate row
     /* UPDATE existing record
     /*
     IGNORE
   NONE VALUE
     MOVE 3700 TO *ERROR-NR
 END-DECIDE
 *
 END
```

17 Processing Natural Stored Procedures and UDFs

Natural for DB2 supports the writing and executing of Natural stored procedures and Natural user-defined functions (Natural UDFs).

Natural stored procedures are user-written programs that are invoked by the SQL statement `CALL` and executed by DB2 in the SPAS (Stored Procedure Address Space). SPAS is a separate address space reserved for stored procedures.

A function is an operation denoted by a function name followed by zero or more operands that are enclosed in parentheses. A function represents a relationship between a set of input values and a set of result values. If a function has been implemented by a user-written program, DB2 refers to it as a user-defined function (UDF).

The following topics are covered below:

Types of Natural UDF

There are two types of Natural used defined functions (UDF):

- **Scalar UDF**

  The scalar UDF accepts several input arguments and returns one output value. It can be invoked by any SQL statement like a DB2 built-in-function.

- **Table UDF**

  The table UDF accepts several input arguments and returns a set of output values comprising one table row during each invocation.

  You invoke a table UDF with a Natural SQL `SELECT` statement by specifying the table-function name in the *FROM Clause*. A table UDF performs as a DB2 table and is invoked for each `FETCH` operation for the table-function specified in the `SELECT` statement.

PARAMETER STYLE

The `PARAMETER STYLE` identifies the linkage convention used to pass parameters to a DB2 stored procedure or a DB2 user defined functions (UDFs).

This section describes the `PARAMETER STYLE`s and the STCB Natural for DB2 uses for processing Natural for DB2 stored procedures or Natural UDFs.

> **Note:** `PARAMETER STYLE GENERAL` (or `GENERAL WITH NULL`) and **STCB Layout** only apply to Natural stored procedures.

- GENERAL and GENERAL WITH NULL

- STCB Layout
- DB2SQL

GENERAL and GENERAL WITH NULL

> **Note:** Only applies to Natural stored procedures.

A Natural stored procedure defined with `PARAMETER STYLE GENERAL` only receives the user parameters specified.

A Natural stored procedure defined with `PARAMETER STYLE GENERAL WITH NULL` receives the user parameters specified and, additionally, a `NULL` indicator array that contains one `NULL` indicator for each user parameter.

Natural stored procedures defined with `PARAMETER STYLE GENERAL`/`PARAMETER STYLE GENERAL WITH NULL`, require that the definition of the stored procedure within the DB2 catalog includes one additional parameter of the type `VARCHAR` in front of the user parameters of the stored procedure.

This parameter in front of the parameters is the Stored Procedure Control Block (STCB); see also *STCB Layout* below.

Below is information on:

- Stored Procedure Control Block
- Example of PARAMETER STYLE GENERAL
- Example of GENERAL WITH NULL

Stored Procedure Control Block

The Stored Procedure Control Block (STCB) contains information the Natural for DB2 server stub uses to execute Natural stored procedures, such as the library and the subprogram to be invoked. It also contains the format descriptions of the parameters passed to the stored procedure.

The STCB is invisible to the Natural stored procedure called. The STCB is evaluated by the Natural for DB2 server stub and stripped off the parameter list that is passed to the Natural stored procedure.

If the caller of a Natural stored procedure defined with `PARAMETER STYLE GENERAL`/`PARAMETER STYLE GENERAL WITH NULL` is a Natural program, the program must use a Natural SQL `CALLDBPROC` statement with the keyword `CALLMODE=NATURAL`.

If the caller of the Natural stored procedure is *not* a Natural program, the caller has to set up the STCB for the DB2 `CALL` statement and pass the STCB as the first parameter.

If an error occurs during the execution of a Natural stored procedure defined with `PARAMETER STYLE GENERAL`/`PARAMETER STYLE GENERAL WITH NULL`, the error message text is returned to the STCB.

If the caller is a Natural program that uses `CALLDBPROC` and `CALLMODE=NATURAL`, the Natural for DB2 runtime will wrap up the error text in the NAT3286 error message.

Example of PARAMETER STYLE GENERAL

In the Natural stored procedure, define the parameters as shown in the example program below:

```
DEFINE DATA PARAMETER
01 P1 ...
01 P2 ...
...
...
01 Pn ...
LOCAL
...
...
END-DEFINE
```

Example of GENERAL WITH NULL

In the Natural stored procedure, define the parameters as shown in the example program below:

```
DEFINE DATA PARAMETER
01 P1 ...
01 P2 ...
...
...
01 Pn ...
01 NULL-INDICATOR-ARRAY   (I2/1:n)
LOCAL
...
...
END-DEFINE
```

STCB Layout

Note: Only applies to Natural stored procedures.

The following table describes the first parameter passed between the caller and the Natural stored procedure if `CALLMODE=NATURAL` is specified in a Natural SQL `CALLDBPROC` statement.

Name	Format	Processing Mode Server
STCBL	I2	Input (size of following information)
Procedure Information		
STCBLENG	A4	Input (printable STCBL)
STCBID	A4	Input (STCB)
STCBVERS	A4	Input (version of STCB 310)
STCBUSER	A8	Input (user ID)
STCBLIB	A8	Input (library)
STCBPROG	A8	Input (calling program)
STCBPSW	A8	Unused (password)
STCBSTNR	A4	Input (CALLDBPROC statement number)
STCBSTPC	A8	Input (procedure called)
STCBPANR	A4	Input (number of parameters)
Error Information		
STCBERNR	A5	Output (Natural error number)
STCBSTAT	A1	Unused (Natural error status)
STCBLIB	A8	Unused (Natural error library)
STCBPRG	A8	Unused (Natural error program)
STCBLVL	A1	Unused (Natural error level)
STCBOTP	A1	Unused (error object type)
STCBEDYL	A2	Output (error text length)
STCBEDYT	A88	Output (error text)
	A100	Reserved for future use
Parameter Information		
STCBPADE	A variable	Input. See also *PARAMETER DESCRIPTION (STCBPADE)* below.

PARAMETER DESCRIPTION (STCBPADE)

PARAMETER DESCRIPTION contains a description for each parameter passed to the Natural stored procedure consisting of parameter type, format specification and length. Parameter type is the AD attribute of the Natural CALLNAT statement as described in the *Statements* documentation.

Each parameter has the following format description element in the STCBPADE string

at1,p[,d1]....

where

- *a* is an attribute mark which specifies the parameter type:

Mark	Type	Equivalent AD Attribute	Equivalent DB2 Clause
M	modifiable	AD=M	INOUT
0	non-modifiable	AD=0	IN
A	input only	AD=A	OUT

- t is one of the following Natural format tokens:

t	Description	l	p	$d1$	Example
A	Alphanumeric	1-253	0	1-32767 or -	A30,0 or A30,0,10
N	Numeric unpacked	1-29	0-7	-	N10,3
P	Packed numeric	1-29	0-7	-	P13,4
I	Integer	2 or 4	0	-	I2,0
F	Floating point		0	-	I4,0
B	Binary		0	-	B23,0
D	Date	6	0	-	D6
T	Time	12	0	-	T12
L	Logical (unsupported)				

- l is an integer denoting the length/scale of the field. For numeric and packed numeric fields, l denotes the total number of digits of the field that is, the sum of the digits left and right of the decimal point. The Natural format N7.3 is, for example, represented by N10.3. See also the **table** above.

- p is an integer denoting the precision of the field. It is usually 0, except for numeric and packed fields where it denotes the number of digits right of the decimal point. See also the table above.

- $d1$ is also an integer denoting the occurrences of the alphanumeric array (alphanumeric only). See also the **table** above.

This descriptive/control parameter is invisible to the calling Natural program and to the called Natural stored procedure, but it has to be defined in the parameter definition of the stored procedure row with the CREATE PROCEDURE statement and the DB2 PARAMETER STYLE GENERAL/PARAMETER STYLE GENERAL WITH NULL.

The following table shows the number of parameters which have to be defined with the CREATE PROCEDURE statement for a Natural stored procedure defined with PARAMETER STYLE GENERAL depending on the number of user parameters and whether the client (that is, the caller of a stored procedure for DB2) and the server (that is, the stored procedure for DB2) is written in Natural or in another standard programming host language. n denotes the number of user parameters.

Client\Server	Natural	not Natural
Natural	$n + 1$	n (CALLMODE=NONE)
non-Natural	$n + 1$	n

DB2SQL

> **Note:** `PARAMETER DB2SQL` applies to Natural stored procedures and Natural UDFs.

A Natural stored procedure or Natural user defined function (UDF) with `PARAMETER STYLE DB2SQL` first receives the user parameters specified and then the parameters listed below, under *Additional Parameters Passed*. For a Natural UDF, the input parameters are passed before the output parameters.

Additional Parameters Passed:

■ A `NULL` indicator for each user parameter of the `CALL` statement,

■ the SQLSTATE to be returned to DB2,

■ the qualified name of the Natural stored procedure or UDF,

■ the specific name of the Natural stored procedure or UDF,

■ the SQL DIAGNOSE field with a diagnostic string to be returned to DB2.

The SQLSTATE, the qualified name, the specific name and the `DIAGNOSE` field are defined in the Natural parameter data area (PDA) `DB2SQL_P` which is supplied in the Natural system library `SYSDB2`.

If the optional feature `SCRATCHPAD` *nnn* is specified additionally in the `CREATE FUNCTION` statement for the Natural UDF, the `SCRATCHPAD` storage parameter is passed to the Natural UDF.

Use the following definitions:

```
01 SCRATCHPAD A(4+nnn)
01 REDEFINE SCRATCHPAD
02 SCRATCHPAD_LENGTH(I4)
02 ...
```

Redefine the `SCRATCHPAD` in the Natural UDF according to your requirements.

The first four bytes of the `SCRATCHPAD` area contain an integer length field. Before initially invoking the Natural UDF with an SQL statement, DB2 resets the `SCRATCHPAD` area to `x'00'` and sets the size *nnn* specified for the `SCRATCHPAD` into the first four bytes as an integer value.

Thereafter, DB2 does not reinitialize the `SCRATCHPAD` between the invocations of the Natural UDF for the invoking SQL statement. Instead, after returning from the Natural UDF, the contents of the `SCRATCHPAD` is preserved and restored at the next invocation of the Natural UDF.

Below is information on:

Parameter CALL TYPE

Note: This parameter is optional and only applies to Natural UDFs.

The `CALL TYPE` parameter is passed if the `FINAL CALL` option is specified for a Natural scalar UDF, or if the Natural UDF is a table UDF. The `CALL TYPE` parameter is an integer indicating the type of call DB2 performs on the Natural UDF. See the *DB2 SQL GUIDE* for details on the parameter values provided in the `CALL_TYPE` parameter.

Parameter DBINFO

This parameter is optional.

If the option `DBINFO` is used, the `DBINFO` structure is passed to the Natural stored procedure or UDF. The `DBINFO` structure is described in the Natural PDA `DBINFO_P` supplied in the Natural system library `SYSDB2`.

Determining Library, Subprogram and Parameter Formats

The Natural for DB2 server stub determines the subprogram and the library from the qualified and specific name of the Natural stored procedure or UDF. The SCHEMA name is used as library name, and the procedure or function name is used as subprogram name.

The `ROUTINEN` subprogram is supplied in the Natural system library `SYSDB2`. This subprogram is used to access the DB2 catalog to determine the formats of the user parameters defined for the Natural stored procedure or UDF. After the formats have been determined, they are stored in the Natural buffer pool. During subsequent invocations of the Natural stored procedure, the formats are then retrieved from the Natural buffer pool. This requires that at least `READS SQL DATA` is specified for Natural stored procedures or UDFs with `PARAMETER STYLE DB2SQL`.

The `ROUTINEN` subprogram is generated statically. The DBRM of `ROUTINEN` is bound as package in the `COLLECTION SAGNDBROUTINENPACK`. Before starting to access the DB2 catalog, the subprogram will save the `CURRENT PACKAGESET` and set `SAGNDBROUTINENPACK` to `CURRENT PACKAGESET`. After processing, the `ROUTINEN` subprogram will restore the `CURRENT PACKAGESET` saved.

Invoking a Natural Stored Procedure

If the caller of the Natural stored procedure with `PARAMETER STYLE DB2SQL` is a Natural program, the caller must use the Natural SQL `CALLDBPROC` statement with the specification `CALLMODE=NATURAL`, which is the default.

Error Handling

If a Natural runtime error occurs during the execution of a Natural stored procedure or UDF with `PARAMETER STYLE DB2SQL`, SQLSTATE is set to `38N99` and the diagnostic string contains the text of the Natural error message.

If an error occurs in the Natural for DB2 server stub during the execution of the Natural stored procedure or UDF with `PARAMETER STYLE DB2SQL`, the SQLSTATE is set to `38S99` and the diagnostic string contains the text of the error message.

If the application wants to raise an error condition during the execution of a Natural stored procedure or UDF, the SQLSTATE parameter must be set to a value other than `'00000'`. See the *DB2 SQL Guide* for specifications of user errors in the SQLSTATE parameter.

Additionally, a text describing the errors can be placed in the `DIAGNOSE` parameter.

If a Natural table UDF wants to signal to DB2 that it has found no row to return, `'02000'` must be returned in the SQLSTATE parameter.

Lifetime of Natural Session

For a Natural UDF that contains the attributes `DISALLOW PARALLEL` and `FINAL CALL`, the Natural for DB2 server stub retains the Natural session allocated earlier. This Natural session will then be reused by all subsequent UDF invocations until Natural encounters the final call.

Example of DB2SQL - Natural Stored Procedure

In a Natural stored procedure, define the parameters as shown in the example program below:

```
DEFINE DATA PARAMETER
01 P1 ...
01 P2 ...
...
...
01 PN ...
01 N1 (I2)
01 N2 (I2)
...
...
01 N
n (I2)
PARAMETER USING DB2SQL_P
```

```
[ PARAMETER USING DBINFO_P ]   /*  only if DBINFO is defined
LOCAL
...
...
END-DEFINE
```

Example of DB2SQL - Natural UDF

In a Natural UDF, define the parameters as shown in the example program below:

```
DEFINE DATA PARAMETER
01 PI1 ...  /* first input parameter
01 PI2 ...
...
...
01 PIn ...  /*  last input parameter
01 RS1...  /* first result parameter
...
...
01 RSn ...  /* last result parameter
01 N_PI1 (I2)  /* first NULL indicator
01 N_PI2 (I2)
...
...
01 N_Pin (I2)
01 N_RS1 (I2)
...
...
01 N_RSn (I2) /* last NULL indicator
PARAMETER USING DB2SQL_P /* function, specific, sqlstate, diagnose
PARAMETER
01 SCRATCHPAD A(4+nnn)  /* only if SCRATCHPAD nnn is specified
  01 REDEFINES SCRATCHPAD
02 SCRATCHPAD_LENGTH (I4)
02 ...
01 CALL_TYPE (I4) /* --- only if FINAL CALL is specified or table UDF


PARAMETER USING DBINFO_P   /* ---- only if DBINFO is specified
LOCAL
...
...
END-DEFINE
```

Writing a Natural Stored Procedure

This section provides a general guideline of how to write a Natural Stored Procedure and what to consider when writing it.

> ### To write a Natural stored procedure

1   Determine the format and attributes of the parameters that are passed between the caller and the stored procedure. Consider creating a Natural parameter data area (PDA). Stored procedures do not support data groups and redefinition within their parameters.

2   Determine the `PARAMETER STYLE` of the stored procedure: `GENERAL`, `GENERAL WITH NULL` or `DB2SQL`.

- If you use `GENERAL WITH NULL`, append the parameters to the Natural stored procedure by defining a `NULL` indicator array that contains a `NULL` indicator (I2) for each other parameter.

- If you use `DB2SQL`, append the parameters of the Natural stored procedure by defining `NULL` indicators (one for each parameter), include the PDA `DB2SQL_P` and the PDA `DBINFO_P` (only with `DBINFO` specified), if desired. See also the relevant DB2 literature by IBM.

3   Decide which and how many result sets the stored procedure will return to the caller.

4   Code your stored procedure as a Natural subprogram.

- **Returning result sets**
  To return result sets, code a Natural SQL `SELECT` statement with the `WITH RETURN` option.

  To return the whole result set, code an `ESCAPE BOTTOM` statement immediately after the `SELECT` statement.

  To return part of the result set code, an `IF *COUNTER = 1 ESCAPE TOP END-IF` immediately following the `SELECT` statement. This ensures that you do not process the first empty row that is returned by the `SELECT WITH RETURN` statement. To stop row processing, execute an `ESCAPE BOTTOM` statement.

  If you do not leave the processing loop initiated by the `SELECT WITH RETURN` **via** `ESCAPE BOTTOM`, the result set created is closed and nothing is returned to the caller.

- **Attention when accessing other databases**
  You can access other databases (for instance Adabas) within a Natural stored procedure. However, keep in mind that your access to other databases is synchronized neither with the updates done by the caller of the stored procedure, nor with the updates done against DB2 within the stored procedure.

■ **Natural for DB2 handling of COMMIT and ROLLBACK statements**
DB2 does not allow a stored procedure to issue Natural SQL `COMMIT` or `ROLLBACK` statements (the execution of those statements puts the caller into a must-rollback state). Therefore, the Natural for DB2 runtime handles those statements as follows when they are issued from a stored procedure:

`COMMIT` against DB2 will be skipped. This allows the stored procedure to commit Adabas updates without getting a must-rollback state from DB2.

`ROLLBACK` against DB2 will be skipped if it is created by Natural itself.

`ROLLBACK` against DB2 will be executed if it is user-programmed. Thus, after a Natural error, the caller receives the Natural error information and not the unqualified must-rollback state. Additionally, this function ensures that, if the user program backs out the transaction, every database transaction of the stored procedure is backed out.

5 **For DB2 UDB:** Issue a `CREATE PROCEDURE` statement that defines your stored procedure, for example:

```
CREATE PROCEDURE <PROCEDURE>
  (INOUT  STCB      VARCHAR(274+13*N),
   INOUT  <PARM1>   <FORMAT>,
   INOUT  <PARM2>   <FORMAT>,
   INOUT  <PARM3>   <FORMAT>
   .
  )
 DYNAMIC RESULT SET <RESULT_SETS>
 EXTERNAL NAME <LOADMOD>
 LANGUAGE ASSEMBLE
 PROGRAM TYPE <PGM_TYPE>
 PARAMETER STYLE GENERAL <WITH NULLS depending on LINKAGE>;
```

The data specified in angle brackets (< >) correspond to the data listed in the **table** above, `PARM1` - `PARM3` and `FORMAT` depend on the call parameter list of the stored procedure. See also *Example Stored Procedure NDBPURGN*, Member `CR6PURGN`.

6 Code your Natural program invoking the stored procedure via the Natural SQL `CALLDBPROC` statement.

Code the parameters in the `CALLDBPROC` statement in the same sequence as they are specified in the stored procedure. Define the parameters in the calling program in a format that is compatible with the format defined in the stored procedure.

If you use result sets, specify a `RESULT SETS` clause in the `CALLDBPROC` statement followed by a number of result set locator variables of `FORMAT (I4)`. The number of result set locator variables should be the same as the number or result sets created by the stored procedure. If you specify fewer than are created, some result sets are lost. If you specify more than are

created, the remaining result set locator variables are lost. The sequence of locator variables corresponds to the sequence in which the result sets are created by the stored procedure.

Keep in mind that the fields into which the result set rows are read have to correspond to the fields used in the `SELECT WITH RETURN` statement that created the result set.

Writing a Natural UDF

This section provides a general guideline of how to write a Natural user defined function (UDF) and what to consider when writing it.

See also the section *Writing a Natural Stored Procedure*.

> **To write a Natural UDF**

1   Determine the format and attributes of the parameters, which are passed between the caller and the stored procedure.

2   Create a Natural parameter data area (PDA).

3   Append the parameter definitions of the Natural UDF by defining `NULL` indicators (one for each parameter) and include the PDA `DB2SQL_P`.

4   If required, code a `SCRATCHPAD` area in the parameter list.

5   If required, code a call-type parameter. If you have specified `DBINFO`, include the PDA `DBINFO_P`. See also the relevant DB2 literature by IBM.

6   Code your UDF as a Natural subprogram and consider the following:

   ■ **Attention when accessing other databases**
   You can access other databases (for example, Adabas) within a Natural UDF. However, keep in mind that your access to other databases is synchronized neither with the updates done by the caller of the stored procedure, nor with the updates done against DB2 within the stored procedure.

   ■ **Natural for DB2 handling of COMMIT and ROLLBACK statements**
   DB2 does not allow a stored procedure to issue `COMMIT` or `ROLLBACK` statements; the execution of these statements results in a must-rollback state. If a Natural stored procedure issues a `COMMIT` or `ROLLBACK`, the Natural for DB2 runtime processes these statements as follows:

   `COMMIT` against DB2 is skipped. This allows the stored procedure to commit Adabas updates without entering a must-rollback state by DB2.

   `ROLLBACK` against DB2 is skipped if it is implicitly issued by the Natural runtime.

   `ROLLBACK` against DB2 is executed if it is user-programmed. Thus, after a Natural error, the caller receives a corresponding Natural error message text, but does not enter an unqualified

must-rollback state. Additionally, this reaction ensures that every database transaction the stored procedure performs is backed out if the user program backs out the transaction.

7   Issue a `CREATE FUNCTION` statement that defines your UDF, for example:

```
CREATE FUNCTION <FUNCTION>
  ([PARM1]    <FORMAT>,
   [PARM2]    <FORMAT>,
   [PARM3]    <FORMAT>

  )
  RETURNS <FORMAT>

  EXTERNAL NAME <LOADMOD>
  LANGUAGE ASSEMBLE
  PROGRAM TYPE <PGM TYPE>
  PARAMETER STYLE DB2SQL
.
.
.;
```

In the example above, the variable data are enclosed in angle brackets (< >) and refer to the keywords preceding the brackets. Specify a valid value, for example:

`LOADMOD` denotes the Natural for DB2 server stub module, for example, `NDB`*vr*`SRV`, where *vr* stands for the Natural version number. `PARM1` - `PARM3` and `FORMAT` relate to the call parameter list of the UDF. See also the *Example Natural User Defined Function*.

8   Code a Natural program containing SQL statements that invoke the UDF.

Specify the parameters of the Natural UDF invocation in the same sequence as specified in the Natural UDF definition. The format of the parameters in the calling program must be compatible with the format defined in the Natural UDF.

Example Stored Procedure

This section describes the example stored procedure `NDBPURGN`, a Natural subprogram which purges Natural objects from the buffer pool used by the Natural stored procedures server.

The following topics are covered below:

- Objects of NDBPURGN

Objects of NDBPURGN

The example stored procedure `NDBPURGN` comprises the following text objects (members) which are stored in the Natural system library `SYSDB2`:

Object	Explanation
CR6PURGN	Input member (text object) for `SYSDB2` ISQL. Contains SQL statements used to declare `NDBPURGN` in DB2.
NDBPURGP	The client (Natural) program which ■ Requests the name of the program to be purged and the library where it resides, ■ Invokes the stored procedure `NDBPURGN` and ■ Reports the outcome of the request. Ensure that the DB2 special register `PATH` contains the schema name of the stored procedure `NDBPURGN` created by `CR6PURGN`. This can be achieved with the SQL statement `SET PATH` *schema-name*. If the schema name is missing, SQLCODE-440 can be returned.
NDBPURGN	The stored procedure which purges objects from the buffer pool. `NDBPURGN` invokes the application programming interface `USR0340N` supplied in the Natural system library `SYSEXT`. Therefore, `USR0340N` must be available in the library defined as the steplib for the execution environment.

Defining the Stored Procedure NDBPURGN

≫ **To define the example stored procedure NDBPURGN**

1   Define the stored procedure in the DB2 catalog by using the SQL statements provided as text objects `CR5PURGN` (for DB2 Version 5) and `CR6PURGN` (for DB2 Version 6).

2   Specify the name of the Natural stored procedure stub (here: `NDB`*vr*`SRV`, where *vr* stands for the Natural version number) as `LOADMOD` (V5) or `EXTERNAL NAME` (V6). The Natural stored procedure stub is generated during the installation by assembling the `NDBSTUB` macro.

3   As the first parameter, pass the internal Natural parameter `STCB` to the stored procedure. The `STCB` parameter is a `VARCHAR` field which contains information required to invoke the stored procedure in Natural:

■ The program name of the stored procedure and the library where it resides,

■ The description of the parameters passed to the stored procedure and

■ The error message created by Natural if the stored procedure fails during the execution.

The STCB parameter is generated automatically by the CALLMODE=NATURAL clause of the Natural SQL CALLDBPROC statement and is removed from the parameters passed to the Natural stored procedure by the server stub. Thus, STCB is invisible to the caller and the stored procedure. However, if a non-Natural client intends to call a Natural stored procedure, the client has to pass the STCB parameter explicitly. See also *Stored Procedure Control Block* below.

Stored Procedure Control Block (STCB)

Below is the Stored Procedure Control Block (STBC) generated by the CALLMODE=NATURAL clause as generated by the stored procedure NDBPURGN *before* and *after* execution. Changed values are emphasized in boldface:

STCB before Execution:

```
004C82   0132F0F3  F0F6E2E3  C3C2F3F1  F040C8C7  *..0306STCB310 HG*  11097D42
004C92   D2404040  4040C8C7  D2404040  4040D5C4  *K     SAG    ND*  11097D52
004CA2   C2D7E4D9  C7D74040  40404040  4040F0F5  *BPURGP        05*  11097D62
004CB2   F7F0D5C4  C2D7E4D9  C7D5F0F0  F0F6F0F9  *70NDBPURGN000609*  11097D72
004CC2   F9F9F940  40404040  40404040  40404040  *999            *  11097D82
004CD2   40404040  40404040  40404040  40404040  *              *   11097D92
004CE2   40404040  40404040  40404040  40404040  *              *   11097DA2
004CF2   40404040  40404040  40404040  40404040  *              *   11097DB2
004D02   40404040  40404040  40404040  40404040  *              *   11097DC2
004D12   40404040  40404040  40404040  40404040  *              *   11097DD2
004D22   40404040  40404040  40404040  40404040  *              *   11097DE2
004D32   40404040  40404040  40404040  40404040  *              *   11097DF2
004D42   40404040  40404040  40404040  40404040  *              *   11097E02
004D52   40404040  40404040  40404040  40404040  *              *   11097E12
004D62   40404040  40404040  40404040  40404040  *              *   11097E22
004D72   40404040  40404040  40404040  40404040  *              *   11097E32
004D82   40404040  40404040  40404040  40404040  *              *   11097E42
004D92   40404040  D4C1F86B  F0D4C1F4  F06BF0D4  *    MA8,0MA40,0M*  11097E52
004DA2   C2F26BF0  D4C2F26B  F0D4C9F2  6BF0D4C9  *I2,0MI2,0MI2,0MI*  11097E62
004DB2   F26BF04B                                *2,0.          *   11097E72
```

STCB after Execution:

```
004C82   0132F0F3  F0F6E2E3  C3C2F3F1  F040C8C7  *..0306STCB310 HG*  11097D42
004C92   D2404040  4040C8C7  D2404040  4040D5C4  *K     SAG    ND*  11097D52
004CA2   C2D7E4D9  C7D74040  40404040  4040F0F5  *BPURGP        05*  11097D62
004CB2   F7F0D5C4  C2D7E4D9  C7D5F0F0  F0F6F0F0  *70NDBPURGN000600*  11097D72
004CC2   F0F0F040  40404040  40404040  40404040  *000            *  11097D82
004CD2   40404040  40404040  40404040  40404040  *              *   11097D92
004CE2   40404040  40404040  40404040  40404040  *              *   11097DA2
004CF2   40404040  40404040  40404040  40404040  *              *   11097DB2
004D02   40404040  40404040  40404040  40404040  *              *   11097DC2
004D12   40404040  40404040  40404040  40404040  *              *   11097DD2
004D22   40404040  40404040  40404040  40404040  *              *   11097DE2
004D32   40404040  40404040  40404040  40404040  *              *   11097DF2
004D42   40404040  40404040  40404040  40404040  *              *   11097E02
```

```
004D52   40404040   40404040   40404040   40404040   *                  *   11097E12
004D62   40404040   40404040   40404040   40404040   *                  *   11097E22
004D72   40404040   40404040   40404040   40404040   *                  *   11097E32
004D82   40404040   40404040   40404040   40404040   *                  *   11097E42
004D92   40404040   D4C1F86B   F0D4C1F4   F06BF0D4   *    MA8,0MA40,0M*   11097E52
004DA2   C2F26BF0   D4C2F26B   F0D4C9F2   6BF0D4C9   *I2,0MI2,0MI2,0MI*   11097E62
004DB2   F26BF04B                                    *2,0.              *   11097E72
```

Example Natural User Defined Function

This section describes the example user-defined function (UDF) `NAT.DEM2UDFN`, a Natural subprogram used to calculate the product of two numbers.

The example UDF `NAT.DEM2UDF` comprises the following objects that are supplied in the Natural system library `SYSDB2`:

Object	Explanation
DEM2CUDF	Contains SQL statements used to create `DEM2UDFN` (see below).
DEM2UDFP	The client (Natural) program that ■ Fetches rows from the UDF `NAT.DEMO` table, ■ invokes the `NAT.DEM2UDFN` (see below) in the `WHERE` clause, and ■ Displays the rows fetched.
DEM2UDFN	The UDF that builds the product of two numbers. `DEM2UDFN` has to be copied to the Natural library `NAT` on the Natural sytem file `FUSER` in the executing environment.

18   Interface Subprograms

Several Natural and non-Natural subprograms are available to provide you with internal information from Natural for DB2 or specific functions for which no equivalent Natural statements exist.

This section covers the following topics:

- *Natural Subprograms*
- *DB2SERV Interface*

Natural Subprograms

The following Natural subprograms are provided:

Subprogram	Function
NDBCONV	Sets or resets conversational mode 2.
NDBDBRM	Checks whether a Natural program contains SQL access and whether it has been modified for static execution.
NDBDBR2	Checks whether a Natural program contains SQL access and whether it has been modified for static execution.
NDBDBR3	Checks whether a Natural program contains SQL access, whether it has been modified for static execution, and whether it can be generated as static.
NDBERR	Provides diagnostic information on the most recently executed SQL call.
NDBISQL	Executes SQL statements in dynamic mode.
NDBISQLD	Executes SQL statements in dynamic mode, using dynamic variables.
NDBNOERR	Suppresses normal Natural error handling.
NDBNROW	Obtains the number of rows affected by a Natural SQL statement.
NDBSTMP	Provides a DB2 TIMESTAMP column as an alphanumeric field and vice versa.

All these subprograms are provided in the Natural system library SYSDB2 and the Natural library SYSTEM on the system file FNAT.

In addition, the Natural library SYSTEM in the FNAT system file contains the subprogram DBTLIB2N and the subroutine DBDL219S. They are used by NDBDBRM and NDBDBR2. The corresponding parameters must be defined in a DEFINE DATA statement.

The Natural subprograms NDBDBRM, NDBDBR2 and NDBDBR3 allow the optional specification of the database ID, file number, password and cipher code of the library file containing the program to be examined.

If these parameters are not specified, either the actual FNAT file or the FUSER file is used to locate the program to be examined depending on whether the library name begins with "SYS" or not.

Programs invoking `NDBDBRM`, `NDBDBR2` or `NDBDBR3` without these parameters will also work like before this change as the added parameters are declared as optional.

For detailed information on these subprograms, follow the links shown in the table above and read the description of the call format and of the parameters in the text object provided with the subprogram (*subprogram-name*T).

Invoking Subprograms from within a Natural Program

- Natural subprograms are invoked with the Natural `CALLNAT` statement.
- Non-Natural subprograms are invoked with the Natural `CALL` statement.

NDBCONV Subprogram

The Natural subprogram `NDBCONV` is used to either set or reset the conversational mode 2 in CICS environments. Conversational mode 2 means that update transactions are spawned across terminal I/Os until either a `COMMIT` or `ROLLBACK` has been issued (Caution DB2 and CICS resources are kept across terminal I/Os!). This means conversational mode 2 has the same effect as the Natural profile parameter `PSEUDO=OFF`, except that the conversational mode is entered after an DB2 update statement (`UPDATE`, `DELETE`, `INSERT`) and left again after a `COMMIT` or `ROLLBACK`, while `PSEUDO=OFF` causes conversational mode for the total Natural session.

A sample program called `CALLCONV` is provided in library `SYSDB2`; it demonstrates how to invoke `NDBCONV`. A description of the call format and of the parameters is provided in the text object `NDBCONVT`.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBCONV' #CONVERS #RESPONSE
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
#CONVERS	I1	Contains the desired conversational mode (input)
#RESPONSE	I4	Contains the response of NDBCONV (output)

The #CONVERS parameter can contain the following values:

Code	Explanation
0	The conversational mode 2 has to be reset.
1	The conversational mode 2 has to be set.

The #RESPONSE parameter can contain the following response codes:

Code	Explanation
0	The conversational mode 2 has been successfully set or reset.
-1	The specified value of #CONVERS is invalid, the conversational mode has not been changed.
-2	NDBCONV is called in a environment, which is not a CICS environment, where the conversational mode 2 is not supported.

NDBDBRM Subprogram

The Natural subprogram NDBDBRM is used to check whether a Natural program contains SQL access and whether it has been modified for static execution. It is also used to obtain the corresponding DBRM (database request module) name from the header of a Natural program generated as static (see also **Preparing Programs for Static Execution**).

A sample program called CALLDBRM is provided on the installation medium; it demonstrates how to invoke NDBDBRM. A description of the call format and of the parameters is provided in the text object NDBDBRMT.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBDBRM' #LIB #MEM #DBRM #RESP #DBID #FILENR #PASSWORD #CIPHER
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
#LIB	A8	Contains the name of the library of the program to be checked.
#MEM	A8	Contains the name of the program (member) to be checked.
#DBRM	A8	Returns the DBRM name.
#RESP	I2	Returns a response code. The possible codes are listed below.
#DBID	N5	Optional. Database ID of library file.
#FILENR	N5	Optional. File number of library file.
#PASSWORD	A8	Optional. Password of library file.
#CIPHER	N8	Optional. Cipher code of library file.

The #RESP parameter can contain the following values:

Code	Explanation
0	The member #MEM in library #LIB has SQL access; it is static if #DBRM contains a value.
-1	The member #MEM in library #LIB has no SQL access.
-2	The member #MEM in library #LIB does not exist.
-3	No library name has been specified.
-4	No member name has been specified.
-5	The library name must start with a letter.
>-5	Further negative response codes correspond to error numbers of Natural error messages.
>0	Positive response codes correspond to error numbers of Natural Security messages.

NDBDBR2 Subprogram

The Natural subprogram NDBDBR2 is used to check whether a Natural program contains SQL access and whether it has been modified for static execution. It is also used to obtain the corresponding DBRM (database request module) name from the header of a Natural program generated as static (see also **Preparing Programs for Static Execution**) and the time stamp generated by the precompiler.

A sample program called CALLDBR2 is provided on the installation medium; it demonstrates how to invoke NDBDBR2. A description of the call format and of the parameters is provided in the text object NDBDBR2T.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBDBR2' #LIB #MEM #DBRM #TIMESTAMP #PCUSER #PCRELLEV #ISOLLEVL #DATEFORM ↵
#TIMEFORM #RESP #DBID #FILENR #PASSWORD #CIPHER
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
#LIB	A8	Contains the name of the library of the program to be checked.
#MEM	A8	Contains the name of the program (member) to be checked.
#DBRM	A8	Returns the DBRM name.
#TIMESTAMP	B8	Consistency token generated by precompiler.
#PCUSER	A8	Unsupported parameter; retained for compatibility reasons. only.
#PCRELLEV	A1	Unsupported parameter; retained for compatibility reasons only.
#ISOLLEVL	A1	Unsupported parameter; retained for compatibility reasons only.
#DATEFORM	A1	Unsupported parameter; retained for compatibility reasons only.
#TIMEFORM	A1	Unsupported parameter; retained for compatibility reasons only.

Parameter	Format/Length	Explanation
#RESP	I2	Returns a response code. The possible codes are listed below.
#DBID	N5	Optional. Database ID of library file.
#FILENR	N5	Optional. File number of library file.
#PASSWORD	A8	Optional. Password of library file.
#CIPHER	N8	Optional. Cipher code of library file.

The #RESP parameter can contain the following values:

Code	Explanation
0	The member #MEM in library #LIB has SQL access; it is static if #DBRM contains a value.
-1	The member #MEM in library #LIB has no SQL access.
-2	The member #MEM in library #LIB does not exist.
-3	No library name has been specified.
-4	No member name has been specified.
-5	The library name must start with a letter.
>-5	Further negative response codes correspond to error numbers of Natural error messages.
>0	Positive response codes correspond to error numbers of Natural Security messages.

NDBDBR3 Subprogram

The Natural subprogram NDBDBR3is used to check whether a Natural program contains SQL access (#RESP 0), whether the Natural program contains solely SQL statements, which are dynamically executable (#RESP 0, #DBRM '*DYNAMIC') and whether it has been modified for static execution (#RESP 0, #DBRM *dbrmname*). It is also used to obtain the corresponding DBRM (database request module) name from the header of a Natural program generated as static (see also **Preparing Programs for Static Execution**) and the time stamp generated by the precompiler.

A sample program called CALLDBR3 is provided on the installation medium; it demonstrates how to invoke NDBDBR3. A description of the call format and of the parameters is provided in the text object NDBDBR3T.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBDBR3' #LIB #MEM #DBRM #TIMESTAMP #PCUSER #PCRELLEV #ISOLLEVL #DATEFORM ↵
#TIMEFORM #RESP #DBID #FILENR #PASSWORD #CIPHER
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
#LIB	A8	Contains the name of the library of the program to be checked.
#MEM	A8	Contains the name of the program (member) to be checked.
#DBRM	A8	Returns the DBRM name. ■ Space, if program has SQL access, ■ *DYNAMIC, if program contains only dynamically executable SQL, ■ DBRM name, if program has been generated static.
#TIMESTAMP	B8	Consistency token generated by precompiler.
#PCUSER	A8	Unsupported parameter; retained for compatibility reasons only.
#PCRELLEV	A1	Unsupported parameter; retained for compatibility reasons only.
#ISOLLEVL	A1	Unsupported parameter; retained for compatibility reasons only.
#DATEFORM	A1	Unsupported parameter; retained for compatibility reasons only.
#TIMEFORM	A1	Unsupported parameter; retained for compatibility reasons only.
#RESP	I2	Returns a response code. The possible codes are listed below.
#DBID	N5	Optional. Database ID of library file.
#FILENR	N5	Optional. File number of library file.
#PASSWORD	A8	Optional. Password of library file.
#CIPHER	N8	Optional. Cipher code of library file.

The #RESP parameter can contain the following values:

Code	Explanation
0	The member #MEM in library #LIB has SQL access; it is static if #DBRM contains a value other than space and *DYNAMIC.
-1	The member #MEM in library #LIB has no SQL access.
-2	The member #MEM in library #LIB does not exist.
-3	No library name has been specified.
-4	No member name has been specified.
-5	The library name must start with a letter.
>-5	Further negative response codes correspond to error numbers of Natural error messages.
>0	Positive response codes correspond to error numbers of Natural Security messages.

NDBERR Subprogram

The Natural subprogram `NDBERR` replaces Function `E` of the `DB2SERV` interface, which is still provided but no longer documented. It provides diagnostic information on the most recent SQL call. It also returns the database type which returned the error. `NDBERR` is typically called if a database call returns a non-zero SQLCODE (which means a NAT3700 error).

A sample program called `CALLERR` is provided on the installation medium; it demonstrates how to invoke `NDBERR`. A description of the call format and of the parameters is provided in the text object `NDBERRT`.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBERR' #SQLCODE #SQLSTATE #SQLCA #DBTYPE
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
`#SQLCODE`	I4	Returns the SQL return code.
`#SQLSTATE`	A5	Returns a return code for the output of the most recently executed SQL statement.
`#SQLCA`	A136	Returns the SQL communication area of the most recent DB2 access.
`#DBTYPE`	B1	Returns the identifier (in hexadecimal format) for the currently used database (where `X'02'` identifies DB2).

NDBISQL Subprogram

The Natural subprogram `NDBISQL` is used to execute SQL statements in dynamic mode. The `SELECT` statement and all SQL statements which can be prepared dynamically (according to the DB2 literature by IBM) can be passed to `NDBISQL`.

A sample program called `CALLISQL` is provided on the installation medium; it demonstrates how to invoke `NDBISQL`. A description of the call format and of the parameters is provided in the text object `NDBISQLT`.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBISQL'#FUNCTION #TEXT-LEN #TEXT (*) #SQLCA #RESPONSE #WORK-LEN #WORK (*)
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation	
#FUNCTION	A8	For valid functions, see below.	
#TEXT-LEN	I2	Length of the SQL statement or of the buffer for the return area.	
#TEXT	A1(1:V)	Contains the SQL statement (EXECUTE) or receives a data row (FETCH).	
#SQLCA	A136	Contains the SQLCA.	
#RESPONSE	I4	Returns a response code.	
#WORK-LEN	I2	Length of the workarea specified by #WORK (optional).	
#WORK	A1(1:V)	Workarea used to hold SQLDA/SQLVAR and auxiliary fields across calls (optional).	
#DBTYPE	I2	Database type (optional).	
		0	Default
		2	DB2
		4	CNX

Valid functions for the #FUNCTION parameter are:

Function	Parameter	Explanation
CLOSE		Closes the cursor for the SELECT statement.
EXECUTE	#TEXT-LEN #TEXT (*)	Executes the SQL statement. Contains the length of the statement. Contains the SQL statement. The first two characters must be blank.
FETCH	#TEXT-LEN #TEXT (*)	Returns a record from the SELECT statement. Size of #TEXT (in bytes). Buffer for the record.
TITLE	#TEXT-LEN #TEXT (*)	Returns the header for the SELECT statement. Size of #TEXT (in bytes); receives the length of the header (= length of the record). Buffer for the header line.

The #RESPONSE parameter can contain the following response codes:

Code	Function	Explanation
5	EXECUTE	The statement is a SELECT statement.
6	TITLE, FETCH	Data are truncated; only set on first TITLE or FETCH call.
100	FETCH	No record / end of data.
-2		Unsupported data type (for example, GRAPHIC).
-3	TITLE, FETCH	No cursor open; probably invalid call sequence or statement other than SELECT.
-4		Too many columns in result table.
-5		SQLCODE from call.
-6		Version mismatch.
-7		Invalid function.
-8		Error from SQL call.
-9		Workarea invalid (possibly relocation).
-10		Interface not available.
-11	EXECUTE	First two bytes of statement not blank.

Call Sequence

The first call must be an EXECUTE call. NDBISQL has a fixed SQLDA AREA holding space for 50 columns. If this area is too small for a particular SELECT it is possible to supply an optional work area on the calls to NDBISQL by specifying #WORK-LEN (I2) and #WORK(A1/1:V).

This workarea is used to hold the SQLDA and temporary work fields like null indicators and auxiliary fields for numeric columns. Calculate 16 bytes for SQLDA header and 44 bytes for each result column and 2 bytes null indicator for each column and place for each numeric column, when supplying #WORK-LEN and #WORK(*) during NDBISQL calls. If these optional parameters are specified on an EXECUTE call they have also to be specified on any following call.

If the statement is a SELECT statement (that is, response code 5 is returned), any sequence of TITLE and FETCH calls can be used to retrieve the data. A response code of 100 indicates the end of the data.

The cursor must be closed with a CLOSE call.

Function code EXECUTE implicitly closes a cursor which has been opened by a previous EXECUTE call for a SELECT statement.

In TP environments, no terminal I/O can be performed between an EXECUTE call and any TITLE, FETCH or CLOSE call that refers to the same statement.

NDBISQLD Subprogram

The Natural subprogram NDBISQLD is used to execute SQL statements in dynamic mode. The SELECT statement and all SQL statements which can be prepared dynamically (according to the DB2 literature by IBM) can be passed to NDBISQLD.

A sample program called CALISQLD is provided on the installation medium. It demonstrates how to invoke NDBISQLD. A description of the call format and of the parameters is provided in the text object ISQLDT.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBISQLD'#FUNCTION #TEXT #SQLCA #RESPONSE #WORK #DBTYPE
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation	
#FUNCTION	A8	For valid functions, see below.	
#TEXT	A DYNAMIC	Contains the SQL statement (EXECUTE) or receives the data row (FETCH).	
#SQLCA	A136	Contains the SQLCA.	
#RESPONSE	I4	Returns a response code.	
#WORK	A DYNAMIC	Workarea used to hold SQLDA/SQLVAR and auxiliary fields across calls (optional). If specified, #WORK has to be sized large enough to hold all auxiliary fields (SQLDA) for the SQL request.	
#DBTYPE	I2	Database type (optional).	
		0	Default
		2	DB2
		4	CNX

Valid functions for the #FUNCTION parameter are:

Function	Parameter	Explanation
CLOSE	-	Closes the cursor for the SELECT statement.
EXECUTE	#TEXT	Executes the SQL statement. Contains the SQL statement. The first four characters must be blank.
FETCH	#TEXT	Returns a row from the SELECT statement. #TEXT has to be sized large enough to hold the row of the result set created by the SELECT statement.

Function	Parameter	Explanation
		After `FETCH`, the `*LENGTH(#TEXT)` is reduced to the exact size of the row.
`TITLE`	`#TEXT`	Returns the header literals for the `SELECT` statement. `#TEXT` has to be sized large enough to hold the row of the result set created by the `SELECT` statement.

The `#RESPONSE` parameter can contain the following response codes:

Code	Function	Explanation
5	`EXECUTE`	The statement is a `SELECT` statement.
6	`TITLE, FETCH`	Data are truncated; only set on first `TITLE` or `FETCH` call.
100	`FETCH`	No record/end of data.
-2	-	Unsupported data type (for example, `GRAPHIC`).
-3	`TITLE, FETCH`	No cursor open. Probably invalid call sequence or statement other than `SELECT`.
-4	-	Too many columns in result table.
-5	-	SQLCODE from call.
-6	-	Version mismatch.
-7	-	Invalid function.
-8	-	Error from SQL call.
-9	-	Workarea invalid (possibly relocation).
-10	-	Interface not available.
-11	`EXECUTE`	First two bytes of statement not blank.

Call Sequence

The first call must be an `EXECUTE` call. `NDBISQLD` has a fixed SQLDA AREA, holding space for 50 columns. If this area is too small for a particular `SELECT`, it is possible to supply an optional work area on the calls to `NDBISQLD` by `#WORK(A)DYNAMIC`.

This workarea is used to hold the SQLDA and temporary work fields like null indicators and auxiliary fields for numeric columns. Calculate 16 bytes for SQLDA header and 44 bytes for each result column and 2 bytes null indicator for each column and place for each numeric column, when supplying `#WORK(A)DYNAMIC` during `NDBISQLD` calls. If these optional parameters are specified on an `EXECUTE` call, they have also to be specified on any following call.

If the statement is a `SELECT` statement (that is, response code 5 is returned), any sequence of `TITLE` and `FETCH` calls can be used to retrieve the data. A response code of `100` indicates the end of the data.

The cursor must be closed with a `CLOSE` call.

Function code `EXECUTE` implicitly closes a cursor which has been opened by a previous `EXECUTE` call for a `SELECT` statement.

In TP environments, no terminal I/O can be performed between an `EXECUTE` call and any `TITLE`, `FETCH` or `CLOSE` call that refers to the same statement.

NDBNOERR Subprogram

The Natural subprogram `NDBNOERR` is used to suppress Natural NAT3700 errors caused by the next SQL call. This allows a program controlled continuation if an SQL statement produces a non-zero SQLCODE. After the SQL call has been performed, `NDBERR` is used to investigate the SQLCODE.

A sample program called `CALLNOER` is provided on the installation medium; it demonstrates how to invoke `NDBNOERR`. A description of the call format and of the parameters is provided in the text object `NDBNOERT`.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBNOERR'
```

There are no parameters provided with this subprogram.

> **Note:** Only NAT3700 errors (that is, non-zero SQL response codes) are suppressed, and also only errors caused by the next following SQL call.

Restrictions with Database Loops

- If `NDBNOERR` is called before a statement that initiates a database loop and an initialization error occurs, no processing loop will be initiated, unless a `IF NO RECORDS FOUND` clause has been specified.
- If `NDBNOERR` is called within a database loop, it does not apply to the processing loop itself, but only to the SQL statement subsequently executed inside this loop.

NDBNROW Subprogram

The Natural subprogram `NDBNROW` is used to obtain the number of rows affected by the Natural SQL statements Searched `UPDATE`, Searched `DELETE`, and `INSERT`. The number of rows affected is read from the SQL communication area (SQLCA). A positive value represents the number of affected rows, whereas a value of minus one (`-1`) indicates that all rows of a table in a segmented tablespace have been deleted; see also the Natural system variable `*NUMBER` as described in the Natural *System Variables* documentation.

A sample program called `CALLNROW` is provided on the installation medium; it demonstrates how to invoke `NDBNROW`. A description of the call format and of the parameters is provided in the text object `NDBNROWT`.

The calling Natural program must use the following syntax:

```
CALLNAT 'NDBNROW' #NUMBER
```

The parameter `#NUMBER` (I4) contains the number of affected rows.

NDBSTMP Subprogram

For DB2, Natural provides a `TIMESTAMP` column as an alphanumeric field (A26) of the format *YYYY-MM-DD-HH.MM.SS.MMMMMM*.

Since Natural does not yet support computation with such fields, the Natural subprogram `NDBSTMP` is provided to enable this kind of functionality. It converts Natural time variables to DB2 time stamps and vice versa and performs DB2 time stamp arithmetics.

A sample program called `CALLSTMP` is provided on the installation medium; it demonstrates how to invoke `NDBSTMP`. A description of the call format and of the parameters is provided in the text object `NDBSTMPT`.

The functions available are:

Code	Explanation
ADD	Adds time units (labeled durations) to a given DB2 time stamp and returns a Natural time variable and a new DB2 time stamp.
CNT2	Converts a Natural time variable (format T) into a DB2 time stamp (column type `TIMESTAMP`) and labeled durations.
C2TN	Converts a DB2 time stamp (column type `TIMESTAMP`) into a Natural time variable (format T) and labeled durations.
DIFF	Builds the difference between two given DB2 time stamps and returns labeled durations.
GEN	Generates a DB2 time stamp from the current date and time values of the Natural system variable `*TIMX` and returns a new DB2 time stamp.
SUB	Subtracts labeled durations from a given DB2 time stamp and returns a Natural time variable and a new DB2 time stamp.
TEST	Tests a given DB2 time stamp for valid format and returns `TRUE` or `FALSE`.

> **Note:** Labeled durations are units of year, month, day, hour, minute, second and microsecond.

DB2SERV Interface

`DB2SERV` is an Assembler program entry point which can be called from within a Natural program.

`DB2SERV` performs either of the following functions:

- Function D
- Function P

■ **Function D**, which performs the SQL statement `EXECUTE IMMEDIATE`.

■ **Function P**, invokes an Assembler module named `NDBPLAN`.

The parameter or variable values returned by each of these functions are checked for their format, length, and number.

Function D

Function `D` performs the SQL statement `EXECUTE IMMEDIATE`. This allows SQL statements to be issued from within a Natural program.

The SQL statement string that follows the `EXECUTE IMMEDIATE` statement must be assigned to the Natural program variable `STMT`. It must contain valid SQL statements allowed with the `EXECUTE IMMEDIATE` statement as described in the relevant IBM literature. Examples can be found below and in the demonstration programs `DEM2*` in the Natural system library `SYSDB2`.

> **Note:** The conditions that apply to issuing the Natural `END TRANSACTION` or `BACKOUT TRANSACTION` statements also apply when issuing the SQL `COMMIT` or `ROLLBACK` statements.

Command Syntax

```
CALL 'DB2SERV' 'D' STMT STMTL SQLCA RETCODE
```

The variables used in this command are described in the following table:

Variable	Format/Length	Explanation
STMT	A*nnn*	Contains a command string which consists of SQL syntax as described above.
STMTL	I2	Contains the length of the string defined in STMT.
SQLCA	A136	Returns the current contents of the SQL communication area.
RETCODE	I2	Returns an interface return code. The following codes are possible: 0 No warning or error occurred. 4 SQL statement produced an SQL warning.

Variable	Format/Length	Explanation
		8 SQL statement produced an SQL error.
		12 Internal error occurred; the corresponding Natural error message number can be displayed with `SQLERR`.

The current contents of the SQLCA and an interface return code (RETCODE) are returned. The SQLCA is a collection of variables that are used by DB2 to provide an application program with information on the execution of its SQL statements.

The following two examples show you how to use `DB2SERV` with Function `D`.

Example of Function D - DEM2CREA:

```
**********************************************************************
*   DEM2CREA - CREATE TABLE NAT.DEMO                                 *
**********************************************************************
*
DEFINE DATA
LOCAL USING DEMSQLCA
LOCAL
*                                    Parameters for DB2SERV
1 STMT          (A250)
1 STMTL         (I2)     CONST <250>
1 RETCODE       (I2)
*
END-DEFINE
*
COMPRESS  'CREATE TABLE NAT.DEMO'
  '(NAME         CHAR(20)     NOT NULL,'
  ' ADDRESS      VARCHAR(100) NOT NULL,'
  ' DATEOFBIRTH DATE          NOT NULL,'
  ' SALARY       DECIMAL(6,2),'
  ' REMARKS      VARCHAR(500))'
  INTO STMT
CALL 'DB2SERV' 'D' STMT STMTL SQLCA RETCODE
*
END TRANSACTION
*
IF RETCODE = 0
  WRITE 'Table NAT.DEMO created'
ELSE
  FETCH 'SQLERR'
END-IF
END
**********************************************************************
```

> **Note:** The functionality of the DB2SERV Function D is also provided with the PROCESS SQL statement.

Example of Function D - DEM2SET:

```
**********************************************************************
*  DEM2SET - Set Current SQLID                                       *
**********************************************************************
*
DEFINE DATA
LOCAL USING DEMSQLCA
LOCAL
*                                    Parameter for DB2SERV
1 STMT        (A250)
1 STMTL       (I2)     CONST <250>
1 RETCODE     (I2)
1 OLDSQLID    (A8)
1 NEWSQLID    (A8)
*
END-DEFINE
*
SELECT DISTINCT CURRENT SQLID
  INTO OLDSQLID
  FROM SYSIBM.SYSTABLES
ESCAPE BOTTOM
END-SELECT
*
MOVE  'SET CURRENT SQLID="PROD"';
  TO   STMT
CALL 'DB2SERV' 'D' STMT STMTL SQLCA RETCODE
*
IF RETCODE > 0
  FETCH 'SQLERR'
ELSE
  SELECT DISTINCT CURRENT SQLID
    INTO NEWSQLID
    FROM SYSIBM.SYSTABLES
  ESCAPE BOTTOM
  END-SELECT
*
  WRITE ' Old SQLID was :' OLDSQLID
  WRITE ' New SQLID is  :' NEWSQLID
END-IF
*
END
**********************************************************************
```

When using SET CURRENT SQLID, the creator name of a table can be substituted by the current SQLID. This enables you to access identical tables with the same table name but with different

creator names. Thus, table names must not be qualified by a creator name if this is to be substituted by the `SQLID`.

In all supported TP-monitor environments, the `SQLID` can then be kept across terminal I/Os until either the end of the session or its resetting via `DB2SERV`.

Function P

Function `P` invokes an Assembler module named `NDBPLAN`, which is used to establish and/or terminate the DB2 connection under TSO and in batch mode. This allows a Natural application to perform **plan switching under TSO and in batch mode**.

The program `DEM2PLAN` is an example of the use of `DB2SERV` with Function `P`.

The name of the current DB2 subsystem (`#SSM`) and the name of the new application plan (`#PLAN`) must be specified. In addition, an interface return code (`#RETCODE`) and the DB2 reason code (`#REASON`) are returned.

Command Syntax

```
CALL 'DB2SERV' 'P' #SSM #PLAN #RETCODE #REASON
```

Variable	Format/Length	Explanation
`#SSM`	A4	Contains the name of the current DB2 subsystem.
`#PLAN`	A8	Contains the new plan name.
`#RETCODE`		Returns an interface return code. The following codes are possible: 0 No warning or error occurred. 12 The specified new application plan is not scheduled. 99 The current environment is not a Call Attachment Facility (CAF) environment. *nnn* Return code from the CAF interface (see also the relevant DB2 literature by IBM).
`#REASON`	I4	Returns the reason code of the CAF interface (see also the relevant DB2 literature by IBM).

Example of Function P - DEM2PLAN:

```
************************************************************************
*  DEM2PLAN - Switch application plan under TSO/Batch with CAF interface *
************************************************************************
*
DEFINE DATA
LOCAL
*                                        Parameter for DB2SERV
01 #SSM        (A4))    CONST <'DB2'>
01 #PLAN       (A8
01 #RETCODE    (I2)
01 #REASON     (I4)
*
END-DEFINE
*
INPUT 'PLEASE ENTER NEW PLAN NAME' #PLAN (AD='_'I)
*
END TRANSACTION
*
CALL 'DB2SERV' 'P' #SSM #PLAN #RETCODE #REASON
*
DECIDE FOR FIRST VALUE OF #RETCODE
*
  VALUE  0
    IGNORE
  VALUE  99
    INPUT 12/23 'This is not a CAF environment !!'
  VALUE  8,12
    INPUT 12/18 'New plan not scheduled, reason code'
                #REASON (AD=OI EM=H(4))
  NONE
    INPUT 12/15 'CAF interface error'
                #RETCODE (AD=OI EM=Z(3))
                'with reason code'
                #REASON (AD=OI EM=H(4))
*
END-DECIDE
*
END
************************************************************************
```

⚠ **Important:** Plan switching under TSO and in batch mode is possible with the CAF interface only; see also the section *Plan Switching under TSO and in Batch Mode*.

19 Natural File Server for DB2

In all supported TP-monitor environments (CICS, IMS TM, and TSO), Natural for DB2 provides an intermediate work file, referred to as the File Server, to prevent database selection results from being lost with each terminal I/O. Exception: Com-plete.

Concept of the File Server

To avoid reissuing the selection statement used and repositioning the cursors, Natural writes the results of a database selection to an intermediate file. The saved selected rows, which may be required later, are then managed by Natural as if the facilities for conversational processing were available. This is achieved by automatically scrolling the intermediate file for subsequent screens, maintaining position in the work file rather than in DB2.

All rows of all open cursors are rolled out to the file server before the first terminal I/O operation. Subsequently, all data is retrieved from this file if Natural refers to one of the cursors which were previously rolled out (see the description of roll out in *Logical Structure of File Server* below).

If a row is to be updated or deleted, the row is first checked to see if it has been updated in the meantime by some other process. This is done by reselecting and fetching the row from the DB2 database, and then comparing it with the original version as retrieved from the file server. If the row is still unchanged, the update or delete operation can be executed. If not, a corresponding error message is returned. The reselection required when updating or deleting a row is possible in both dynamic mode and static mode.

Only the fields which are stored in the file server are checked for consistency against the record retrieved from DB2.

As the row must be uniquely identified, the Natural view must contain a field for which a unique row has been created. This field must be defined as a unique key in DB2. In a Natural data definition module (DDM), it will then be indicated as a unique key via the corresponding Natural-specific short name.

Preparations for Using the File Server

The size of a row which can be written to the file server is limited to 32 KB or 32767 bytes. If a row is larger, a corresponding error message is returned.

The File Server can use either a VSAM RRDS file or the Software AG Editor buffer pool as storage medium to save selected rows of DB2 tables.

This section covers the following topics:

- File Server - VSAM
- File Server - Editor Buffer Pool

■ File Server – Shared Memory Object

File Server - VSAM

The file server is installed via a batch job, which defines and formats the intermediate file. Samples of this batch job are supplied on the installation medium as described in the relevant section.

Defining the Size of the File Server

The file server is created by defining an RRDS VSAM file using AMS (Access Method Services). Its physical size and its name must be specified.

Formatting the File Server

The file server is formatted by a batch job, which requires five input parameters specified by the user, and which formats the file server according to these parameters. The parameters specify:

1. The number of blocks to be formatted (logical size of the VSAM file); this value is taken from the first parameter of the `RECORD` subcommand of the AMS `DEFINE CLUSTER` command.
2. The number of users that can log on to Natural concurrently.
3. The number of formatted blocks to be defined as primary allocation per user.
4. The number of formatted blocks to be used as secondary allocation per user.
5. The maximum number of file server blocks to be allocated by each user. If this number is exceeded, a corresponding Natural error message is returned.

Immediately before the first access to the file server, a file server directory entry is allocated to the Natural session and the amount of blocks specified as primary allocation is allocated to the Natural session.

The primary allocation is used as intermediate storage for the result of a database selection and should be large enough to accommodate all rows of an ordinary database selection. Should more space in the file server be required for a large database selection, the file server modules allocate a secondary allocation equal to the amount that was specified for secondary allocation when the file server was formatted.

Thus, a secondary area is allocated only when your current primary allocation is not large enough to contain all of the data which must be written to the intermediate file. The number of secondary allocations allowed depends upon the maximum number of blocks you are allowed to allocate. This parameter is also specified when formatting the file server.

The number of blocks defined as the secondary allocation is allocated repeatedly, until either all selected data has been written to the file or the maximum number of blocks you are allowed to allocate is exceeded. If so, a corresponding Natural error message is returned. When the blocks received as a secondary allocation are no longer needed (that is, once the Natural loop associated with this allocation is closed), they are returned to the free blocks pool of the file server.

Your primary allocation of blocks, however, is always allocated to you, until the end of your Natural session.

Changes Required for a Multi-Volume File Server

To minimize channel contention or bottlenecks that can be caused by placing a large and heavily used file server on a single DASD volume, you can create a file server that spans several DASD volumes.

To create and format such a file server, two changes are needed in the job that is used to define the VSAM cluster:

1. Change `VOLUME ()` to `VOLUMES (vol1,vol2,...)`.

2. Divide the total number of records required for the file (as specified with the first format job parameter) by the number of volumes specified above. The result of the calculation is used for the `RECORDS` parameter of the `DEFINE CLUSTER` command.

This means that in the file server format job, the value of the first parameter is the result of multiplying two parameters taken from the `DEFINE CLUSTER` command: `RECORDS` and `VOLUMES`.

File Server - Editor Buffer Pool

The Software AG Editor buffer pool is used as storage medium when `EBPFSRV=ON` is set in the `NTDB2` macro. In this case, the primary, secondary and maximum allocation amounts for the file server are specified by `EBPPRAL`, `EBPSEC`, `EBPMAX` parameters of the `NTDB2` macro. Before Natural for DB2 tries to write data from a Natural user session to the file server for the first time, a Software AG Editor buffer pool logical file is allocated with the Natural terminal identifier as user name and the number 2240 as session number.

The operation of the file server is in this case depending on the definition of the Software AG Editor buffer pool; see *Editor Buffer Pool* in the Natural *Operations* documentation.

The number of logical files for the buffer pool limits the number of users concurrently accessing the file server. The number of work file blocks limits the amount of data to be saved at a specific moment. (You also have to consider that there are other users than Natural for DB2 of the Software AG Editor.)

However, using the Software AG Editor buffer pool as storage medium for the file server enables Natural for DB2 to run in a Sysplex environment.

If you like to use the file server in a sysplex environment, it is recommended to use the Software AG Editor buffer pool as storage medium.

File Server – Shared Memory Object

A z/OS shared memory object above the bar is used as the storage medium for the file server. In this case, the shared memory object has the same structure as the VSAM RRDS file mentioned in the previous section, but all data is maintained in virtual storage above the bar. A file server that uses a shared memory object is referred to as a Shared Memory Objects File Server (FSSM).

≫ **To use a Shared Memory Objects File Server**

1   Define the shared memory object in the `ASMPARM` parameter file (see the *Operations* documentation) of a Natural Authorized Services Manager that contains the `NATFSSM` module.

   For the messages that can be returned by the Authorized Services Manager, see *Messages from the Shared Memory Objects File Server under NDB* in the *Messages and Codes* documentation.

2   Set the `SMFSRV=ON` as the keyword subparameter of the `DB2` profile parameter or `NTDB2` macro.

3   Specify the name of the shared memory object to be used with the `DDFSERV` keyword subparameter of `DB2` (`NTDB2`).

4   Start the Authorized Services Manager configured in Step 1.

When Natural for zIIP is enabled in your Natural environment, switching between SRB and TCB modes for accessing the file server is not required since no disk I/O has to be performed.

Defining Size and Format of an FSSM

The size and format of the FSSM is defined in the `ASMPARM` data set of the Authorized Services Manager, similar to the way the VSAM file server is defined. Each definition has the following format:

```
FSSMxxxx=(name,number-of-blocks,number-of-users,primary-blocks,secondary-blocks,maximum-blocks,block-size)
```

Explanations:

`FSSMxxxx`	The initialization keyword required to define a file server. `FSSM` is a mandatory prefix that can be followed by 1 to 4 characters `xxxx`.
`name`	The logical name for the file server. The name can contain up to 8 characters and must match the name specified with the `DDFSERV` keyword subparameter of `DB2` (`NTDB2`) used in the Natural session.
`number-of-blocks`	The number of blocks used by the file server. Valid range: 3 - 2147483647. The number has to be a multiple of 8.
`number-of-users`	The number of users that can use the file server concurrently. Valid range: 1 - 32767

primary-blocks	The primary block allocation for each user. Valid range: 1 - 32767
secondary-blocks	The secondary block allocation for each user. Valid range: 1 - 32767
maximum-blocks	The maximum block allocation for each user. Valid range: 1 - 32767
block-size	The size of each block in the file server. Valid range: 1 - 32767

Examples:

```
FSSMPRM1=(CMFSERV,1000,500,50,10,100,31744)
```

```
FSSMPRM2=(CMFSERV2,1000,203,50,10,200,4080)
```

Formatting of an FSSM

The FSSM is formatted implicitly when the first user starts a Natural session with `SMFSRV=ON` with the name of the shared memory object specified in `DDFSERV`.

If another user from another address space also uses the same shared memory object, Natural for DB2 implicitly establishes access to the shared memory object for this address space. Access to the shared memory object for an address space only fails when the address space terminates.

Immediately before the first access to the file server, a file server directory entry is allocated to the Natural session and the number of blocks specified as primary allocation is allocated to the Natural session.

The primary allocation is used as the intermediate storage for the result of a database selection and should be large enough to contain all rows of an ordinary database selection. Should the file server require more space, the file server modules perform a secondary allocation using the number of blocks specified for the secondary allocation when the file server was formatted.

Thus, a secondary area is allocated only when your current primary allocation is not large enough to contain all of the data which must be written to the intermediate file. The number of blocks allowed for the secondary allocations depends on the maximum number of blocks you are allowed to allocate.

The number of blocks defined for the secondary allocation is allocated repeatedly, until either all selected data has been written to the file, or until the maximum number of blocks you are allowed to allocate is exceeded. If so, a corresponding Natural error message is returned. When the blocks received as secondary allocation are no longer needed (that is, once the Natural loop associated with this allocation ends), they are returned to the free blocks pool of the file

server. However, your primary allocation of blocks is always allocated to you until the end of your Natural session.

Logical Structure of the File Server

Immediately before a Natural user session accesses the file server, a file server directory entry (VSAM) or a logical file (Software AG Editor buffer pool) is allocated to the Natural user session and the number of blocks specified as primary allocation is reserved until the end of the session.

Generally, the file server is only used when a terminal I/O occurs within an active `READ`, `FIND`, or `SELECT` loop, where database selection results would be lost. Before each terminal I/O operation, Natural checks for any open cursors. For each non-scrollable cursor found, all remaining rows are retrieved from DB2 and written to an intermediate file. For each scrollable cursor, all rows are retrieved from DB2 and written to an intermediate file. In the Natural for DB2 documentation, this process is referred to as cursor roll out.

For each cursor roll out (scrollable and non-scrollable), a logical file is opened to hold all the rows fetched from this cursor. The space for the intermediate file is managed within the space allocated to your session. The logical file is then positioned on the row that was `CURRENT OF CURSOR` when the terminal I/O occurred.

Subsequent requests for data are then satisfied by reading the rows directly from the intermediate file. The database is no longer involved, and DB2 is only used for update, delete or store operations.

Positioned `UPDATE` and/or Positioned `DELETE` statements against rolled-out scrollable cursors are performed against the DB2 base table and against the logical file on the file server.

Once the corresponding processing loop in the application has been closed, the file is no longer needed and the blocks it occupies are returned to your pool of free blocks. From here, the blocks are returned to the free blocks pool of the file server, so that you are left with your primary allocation only.

In the following example, the space allocated to the first selection is not released until all rows selected during the third selection have been retrieved. The same applies to the space allocated to the third selection.

The space allocated to the second selection, however, is released immediately after the last row of the corresponding selection result has been retrieved.

Therefore, the space allocated to the second selection can be used for the selection results of the third selection.

Example:

```
:
:
FIND ... (1st selection)
   :
   :
   FIND ... (2nd selection)
      :
      INPUT ...
      :
   END-FIND
   :
   :
   FIND ... (3rd selection)
      :
      INPUT ...
      :
   END-FIND
   :
   :
END-FIND
:
:
```

Primary Allocation Area

1st Selection

1st Selection

2nd Selection — 2nd/3rd Selection
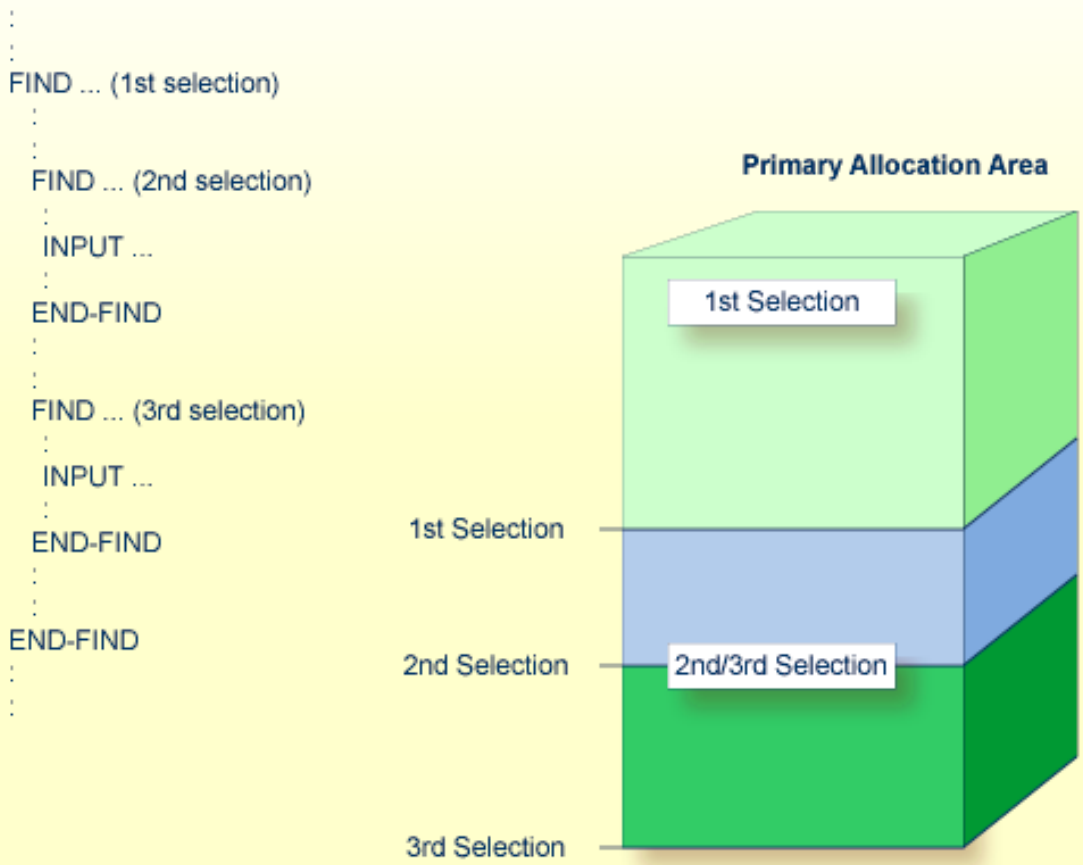
3rd Selection

If the primary allocation area is not large enough, for example, if the third selection is nested within the second selection, the secondary allocation area is used.

Example:

```
    :
    :
FIND ... (1st selection)
    :
    FIND ... (2nd selection)
        :
        FIND ... (3rd selection)
            :
            INPUT ...
            :
        END-FIND
        :
    END-FIND
    :
    :
END-FIND
    :
    :
```

Primary Allocation Area

1st Selection

2nd Selection

3rd Selection

1st Selection

2nd Selection

Secondary Allocation Area

3rd Selection
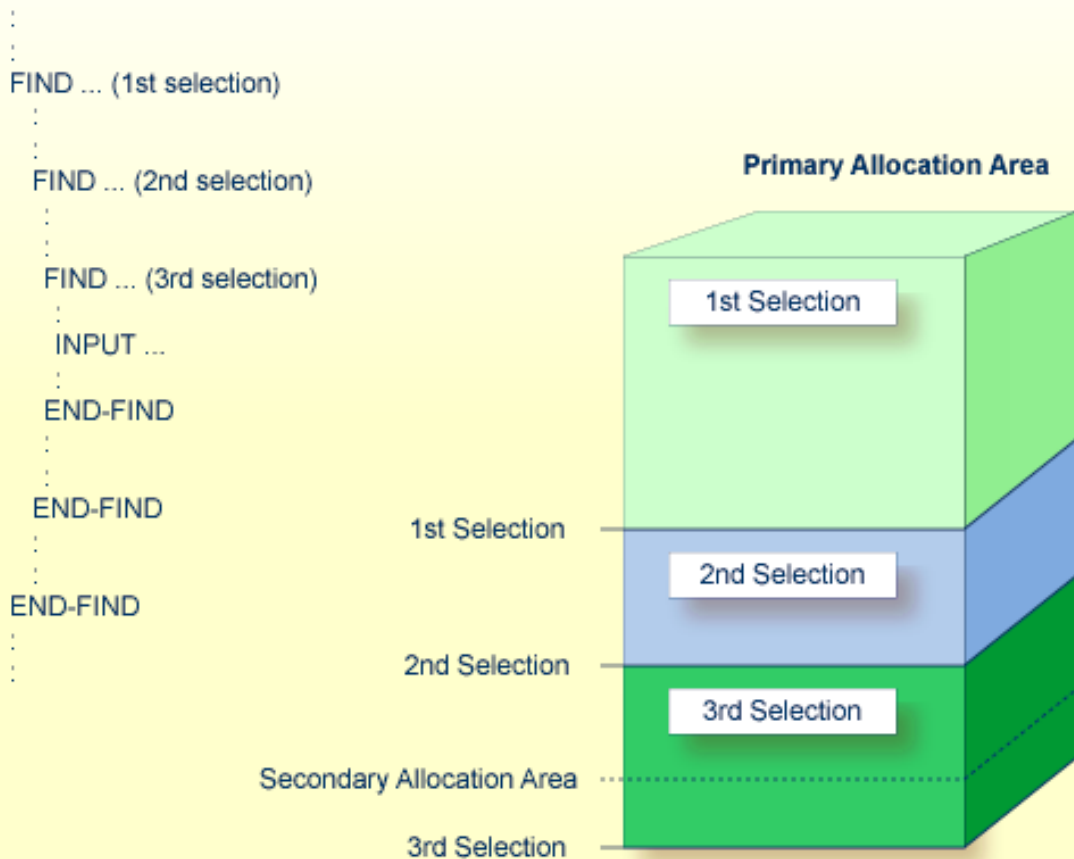
When a session is terminated, all of a user's blocks are returned to the free blocks pool. If a session ends abnormally, Natural checks, where possible, whether a file server directory entry for the corresponding user exists. If so, all resources held by this user are released.

If Natural is unable to free the resources of an abnormally-ended user session, these resources are not released until the same user ID logs on from the same logical terminal again.

If the same user ID and/or logical terminal are not used again for Natural, the existing directory entry and the allocated space remain until the file server is formatted again. A new run of the formatting job deletes all existing data and recreates the directory.

20   Natural for DB2 Version 8.4 - Documentation Updates

> **Note:** The documentation updates provided here only cover the changes specific to Natural for DB2 Version 8.4 and above.

Most changes have been implemented in the SQL statements of Natural for DB2 Version 8.4.1 in support of IBM DB2 Version 12.

For the changes in installation, see *Installing Natural for DB2 Version 8.4.3* in the Natural *Installation* documentation.

Using Natural Statements and System Variables under Natural for DB2 Version 8.4

> **Note:** This is an extract of the chapter *Using Natural SQL Statements and System Variables* and only describes the changes specific to Natural for DB2 Version 8.4.

- UPDATE with FIND/READ
- scalar-function
- column-function
- special-register

UPDATE with FIND/READ

When a Natural program contains a DML `UPDATE` statement, this statement is translated into an SQL `UPDATE` statement and a `FOR UPDATE OF` clause is added to the `SELECT` statement.

Be aware that a primary key field is not part of a `FOR UPDATE OF` list except that the compiler option `DB2PKYU` is set to `ON`. If `DB2PKYU` is set to `OFF` (default), a primary key field can only be updated by using a non-cursor `UPDATE` operation (see also Natural SQL `UPDATE` statement in the section *Using Natural SQL Statements*).

scalar-function

A scalar function is a built-in function that can be used in the construction of scalar computational expressions. Scalar functions are specific to DB2 and belong to the Natural SQL Extended Set.

The scalar functions Natural for DB2 supports are listed below in alphabetical order:

A - H	I - R	S - Z
ABS	IDENTITY_VAL_LOCAL	SCORE
ABSVAL	IFNULL	SECOND
ACOS	INSERT	SIGN
ADD_MONTHS	INTEGER	SIN
ASIN	JULIAN_DAY	SINH
ASCII	LAST_DAY	SMALLINT
ASCII_CHR	LCASE	SOAPHTTPC
ASCII_STR	LEFT	SOAPHTTPV
ATAN	LENGTH	SOAPHTTPNC
ATAN2	LN	SOAPHTTPNV
ATANH	LOCATE	SOUNDEX
BIGINT	LOCATE_IN_STRING	SPACE
BINARY	LOG	SQRT
BITAND	LOG10	STRIP
BITANDNOT	LOWER	SUBSTR
BITNOT	LPAD	SUBSTRING
BITOR	LTRIM	TAN
BITXOR	MAX	TANH
BLOB	MICROSECOND	TIME
CCSID_ENCODING	MIDNIGHT_SECONDS	TIMESTAMP
CEIL	MIN	TIMESTAMPADD
CEILING	MINUTE	TIMESTAMPDIFF
CHAR	MOD	TIMESTAMP_FORMAT
CHARACTER_LENGTH	MONTH	TIMESTAMP_ISO
CLOB	MONTHS_BETWEEN	TIMESTAMP_TZ
COALESCE	MQPUBLISH	TO_CHAR
COLLATION_KEY	MQPUBLISHXML	TO_DATE
COMPARE_DECFLOAT	MQREAD	TO_NUMBER
CONCAT	MQREADCLOB	TOTALORDER
CONTAINS	MQREADXML	TRANSLATE
COS	MQRECEIVE	TRIM
COSH	MQRECEIVECLOB	TRUNC
DATE	MQRECEIVEXML	TRUNC_TIMESTAMP
DAY	MQSEND	TRUNCATE
DAYOFMONTH	MQSENDXML	UCASE
DAYOFWEEK	MQSENDXMLFILE	UNICODE
DAYOFWEEK_ISO	MQSENDXMLFILECLOB	UNICODE_STR
DAYOFYEAR	MQSUBSCRIBE	UNISTR
DAYS	MQUNSUBSCRIBE	UNPACK
DBCLOB	MULTIPLY_ALT	UPPER
DEC	NEXT_DAY	VALUE
DECFLOAT	NORMALIZE_DECFLOAT	VARBINARY
DEDCFLOAT_FORMAT	NORMALIZE_STRING	VARCHAR
DECFLOAT_SORTKEY	NULLIF	VARCHAR9
DECIMAL	NVL	VARCHAR_BIT_FORMAT
DECODE	OVERLAY	VARCHAR_FORMAT
DECRYPT_BIT	PACK	VARGRAPHIC
DECRYPT_CHAR	POSSTR	WEEK

A - H	I - R	S - Z
DECRYPT_DB	POWER	WEEK_ISO
DEGREES	QUANTIZE	XMLATTRIBUTES
DIFFERENCE	QUARTER	XMLCONCAT
DIGITS	RADIANS	XMLCOMMENT
DOUBLE	RAISE_ERROR	XMLDOCUMENT
DOUBLE_PRECISION	RAND	XMLELEMENT
DSN_XMLVALIDATE	REAL	XMLFOREST
EBCDIC_CHR	REPEAT	XMLMODIFY
EBCDIC_STR	REPLACE	XMLNAMESPACES
ENCRYPT_TDES	RID	XMLPARSE
ENCRYPT	RIGHT	XMLPI
EXP	ROUND	XMLQUERY
EXTRACT	ROUND_TIMESTAMP	XMLSERIALIZE
FLOAT	ROWID	XMLTEXT
FLOOR	RPAD	XMLXSROBJECTID
GRAPHIC	RTRIM	YEAR
GENERATE_UNIQUE		
GENERATE_UNIQUE_BINARY		
GETHINT		
GETVARIABLE		
HASH_CRC32		
HASH_MD5		
HASH_SHA1		
HASH_SHA256		
HEX		
HOUR		

Each scalar function is followed by one or more scalar expressions in parentheses. The number of scalar expressions depends upon the scalar function. Multiple scalar expressions must be separated from one another by commas.

Example:

```
SELECT NAME
  INTO NAME
  FROM SQL-PERSONNEL
  WHERE SUBSTR ( NAME, 1, 3 ) = 'Fri'
      ...
```

column-function

The following column functions do not conform to standard SQL. They are specific to DB2 and belong to the Natural SQL Extended Set.

```
COUNT_BIG
CORRELATION
COVARIANCE
COVARIANCE_SAMP
MEDIAN
PERCENTILE_CONT
PERCENTILE_DISC
STDDEV
STDDEV_POP
STDDEV_SAMP
VAR
VAR_POP
VAR_SAMP
VARIANCE
VARIANCE_SAMP
XMLAGG
```

special-register

With the exception of `USER`, the following special registers do not conform to standard SQL. They are specific to DB2 and belong to the Natural SQL Extended Set:

```
CURRENT APPLICATION COMPATIBILITY
CURRENT APPLICATION ENCODING SCHEME
CURRENT CLIENT_ACCNTG
CURRENT CLIENT_APPLNAME
CURRENT CLIENT_CORR_TOKEN
CURRENT CLIENT_USERID
CURRENT CLIENT_WRKSTNNAME
CURRENT DATE
CURRENT_DATE
CURRENT DEBUG MODE
CURRENT DECFLOAT ROUNDING MODE
CURRENT DEGREE
CURRENT FUNCTION PATH
CURRENT GET_ACCEL_ARCHIVE
CURRENT_LC_CTYPE
CURRENT LC_CTYPE
CURRENT LOCALE LC_CTYPE
CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
```

```
CURRENT_MEMBER
CURRENT OPTIMIZATION HINT
CURRENT PACKAGE PATH
CURRENT PACKAGESET
CURRENT_PATH
CURRENT PRECISION
CURRENT QUERY ACCELERATION
CURRENT REFRESH AGE
CURRENT ROUTINE VERSION
CURRENT RULES
CURRENT SCHEMA
CURRENT SERVER
CURRENT SQLID
CURRENT TEMPORAL BUSINESS_TIME
CURRENT TEMPORAL_SYSTEM_TIME
CURRENT TIME
CURRENT_TIME
CURRENT TIMESTAMP
CURRENT TIMEZONE
CURRENT_TIMEZONE USER
SESSION TIME ZONE
SESSION_USER
USER
```

A reference to a special register returns a scalar value.

Using the command `SET CURRENT SQLID`, the creator name of a table can be substituted by the current SQLID. This enables you to access identical tables with the same table name but with different creator names.

Select Expressions under Natural for DB2 Version 8.4

Note: This is an extract of the chapter *Select Expressions* (*Statements* documentation) and only describes the changes specific to Natural for DB2 Version 8.4.

- Selection

- GROUP BY Clause

Selection

```
┌       ┐ ┌ ┌                                              ┐       ┐
│ ALL   │ │ │ scalar-expression [[AS] correlation-name]    │       │
│DISTINCT│ │ │ unpack-row                                   │ , ... │
└       ┘ │ │                                              │       │
          │ └                                              ┘       │
          │              *                                         │
          └                                                        ┘
```

A *selection* specifies the columns of the result set tables to be selected.

Syntax Element Description:

Syntax Element	Description
ALL|DISTINCT	**Elimination of Duplicate Rows:** Duplicate rows are not automatically eliminated from the result of a *select-expression*. To request this, specify the keyword DISTINCT. The alternative to DISTINCT is ALL. ALL is assumed if neither is specified.

scalar-expression	**Scalar Expression:** Instead of, or as well as, simple column names, a selection can also include general scalar expressions containing scalar operators and scalar functions which provide computed values (see also the section *Scalar Expressions*). Example:
	
	`SELECT NAME, 65 - AGE`
	`  FROM SQL-PERSONNEL`
	`  ...`

AS	The optional keyword AS introduces a *correlation-name* for a column.
correlation-name	**Correlation Name:** A *correlation-name* can be assigned to a *scalar-expression* as an alias name for a result column. The *correlation-name* need not be unique. If no *correlation-name* is specified for a result column, the corresponding *column-name* will be used (if the result column is derived from a column name; if not, the result table will have no name). The name of a result column may be used, for example, as column name in the ORDER BY clause of a SELECT statement.
unpack-row	See *unpack-row* below.
*	**Asterisk Notation:** All columns of the result table are selected.

Syntax Element	Description

	Example:
	
	``
	SELECT *
	  FROM SQL-PERSONNEL, SQL-AUTOMOBILES
	   ...
	``

unpack-row

```
UNPACK (scalar-expression) .* AS ({field-name data-type} ,…)
```

An *unpack-row* specifies a row of unpacked binary values that are returned when the SQL UNPACK function is invoked. The number of *field-names* and *data-types* must match the number of fields returned by the UNPACK function.

GROUP BY Clause

```
[            { grouping-expression }        ]
[  GROUP BY  { grouping-set        } , …    ]
[            { super-group         }        ]
```

The GROUP BY clause specifies a grouping of the result table. The result of GROUP BY is a set of groups of rows. Within each group of more than one row, all values defining the group are equal.

grouping-expression
  A grouping expression is a **scalar expression** that defines the grouping of a result set.

grouping-set

```
                    {  { grouping-expression }              }
                    {  { super-group         }              }
GROUPING SETS  (    {                                } , …  )
                    {  ( { grouping-expression } , … )       }
                    {  ( { super-group         }     )       }
```

A *grouping-set* is used to specify multiple grouping clauses in a single statement. A *grouping-set* combines two or more groups of rows into a single result set. It is the same as the union of multiple select expressions with a GROUP BY clause where each expression corresponds to one *grouping-set*. A *grouping-set* is a single element or a list of elements delimited by parentheses. An element is either a *grouping-expression* or a *super-group*. A *grouping-set* has the advantage that the groups are computed with a single pass over the base table.

super-group

```
{ ROLLUP (grouping-expression-list) }
{ CUBE   (grouping-expression-list }
{          ( )                      }
```

A *super-group* is a more complex *grouping-set*.

A *grouping-expression-list* defines the number of elements used in a `ROLLUP` or `CUBE` operation. Elements with multiple *grouping-expressions* are delimited by parentheses:

```
{ grouping-expression          }
{ (grouping-expression, ...)   } , ...
```

Grand total ( ):

`ROLLUP` and `CUBE` return a row which is the overall (grand total) aggregation. This can be specified with empty parentheses ( ) within the `GROUPING SETS` clause.

ROLLUP

A `ROLLUP` grouping is like a series of *grouping-sets*. In addition to the regular grouped rows, a `ROLLUP` grouping produces a result set that contains subtotal rows. Subtotal rows are "super-aggregate" rows which contain additional aggregates. The aggregate values are retrieved with the same column functions that are used to obtain the regular grouped rows.

In general, you specify a `ROLLUP` with n elements as

```
GROUP BY ROLLUP (c1, c2, ..., cn-1, cn)
```

which is the equivalent of:

```
GROUP BY GROUPING SETS ((c1, c2, ..., cn-1, cn),

                        (c1, c2, ..., cn-1),

                        ...

                        (c1, c2),

                        (c1),

                        ( ))
```

CUBE

A `CUBE` grouping is like a series of *grouping-sets*. In addition to the `ROLLUP` aggregation rows, `CUBE` produces a result set that contains cross-tabulation rows. Cross-tabulation rows

are additional "super-aggregate" rows. The *grouping-expression-list* of a CUBE computes all permutations along with the **grand total**. As a result, the *n* elements of a CUBE translate to 2**n grouping-sets. For example:

```
GROUP BY CUBE (a, b, c)
```

is the equivalent of:

```
GROUP BY GROUPING SETS ((a, b, c),

                        (a, b),

                        (a, c),

                        (b, c),

                        (a),

                        (b),

                        (c),

                        ())
```

Dynamic and Static SQL Support under Natural for DB2 Version 8.4

Note: This is an extract of the chapter *Dynamic and Static SQL Support* and only describes the changes specific to Natural for DB2 Version 8.4.

Plan Switching by CICS/DB2 Exit Routine

If #SWITCH-BY-TRANSACTION-ID is set to FALSE, the desired plan name is written to a temporary storage queue for a CICS/DB2 exit routine specified as PLANExit attribute of a DB2ENTRY or of the DB2CONN definition, the NATPLAN program must be invoked before the first DB2 access. Natural for DB2 provides NDBUEXT as CICS DB2 plan selection exit program. For additional information on CICS/DB2 exit routines, refer to the relevant IBM literature.

The name of the temporary storage queue is PLAN*sssstttt*, where *ssss* is the remote or local CICS system identifier and *tttt* the CICS terminal identifier.

When running in a CICSplex environment, the CICS temporary storage queue PLAN*sssstttt* containing the plan name must be defined with TYPE=SHARED or TYPE=REMOTE in a CICS TST.

For each new DB2 unit of recovery, the appropriate plan selection exit routine is automatically invoked. This exit routine reads the temporary storage record and uses the contained plan name for plan selection.

When no temporary storage record exists for the Natural session, a default plan name, contained in the plan exit, can be used. If no plan name is specified by the exit, the name of the plan used is the same as the name of the static program (DBRM) issuing the SQL call. If no such plan name exists, an SQL error results.

SELECT under Natural for DB2 Version 8.4

Note: This is an extract of *Syntax 1 - Extended Set* and *Syntax Element Description* in the section *SELECT (SQL)* (*Statements* documentation) and only describes the changes specific to Natural for DB2 Version 8.4.

Syntax 1 - Extended Set

```
[WITH_CTE common-table-expression, ...]
SELECT selection into-clause table-expression

                                            (SELECT selection
      {   UNION     }                        table-expression)
      {   EXCEPT    }    [  DISTINCT  ]  [                        ]
[     {   INTERSECT }    [  ALL       ]     SELECT selection      ]     ...
                                            table-expression

[ORDER BY criteria]

                               {  ROW   }
[   OPTIMIZE FOR integer       {  ROWS  }  ]

[WITH isolation-level]
[SKIP LOCKED DATA]
[QUERYNO integer]
[OFFSET row-count]
[FETCH FIRST row-limit]
[WITH HOLD]
[WITH RETURN]
[WITH scroll-mode]
[WITH ROWSET POSITIONING FOR max-rowsets]
[IF NO RECORDS FOUND instruction]
statement ...
{   END-SELECT   }
{   LOOP         }
```

OFFSET *row-count*

```
OFFSET [offset-row-count] { ROW  }
                          { ROWS }
```

The OFFSET clause specifies the number of rows to skip in the result table before retrieving any rows from there. A limited number of rows at the end of a result set can improve the performance of queries with potentially large result sets.

offset-row-count is a numeric variable or constant which determines the number of rows to be skipped. The number must be zero (0) or a positive integer.

FETCH FIRST *row-limit*

```
FETCH FIRST [ 1         ] { ROW  }
            [ row-count ] { ROWS }
```

The FETCH FIRST clause limits the number of rows to be fetched. A limited number of rows can improve the performance of queries with potentially large result sets.

row-count specifies a numeric variable or constant which determines the number of rows to be fetched. The number must be a positive integer.

MERGE under Natural for DB2 Version 8.4

Note: This is an extract of the chapter *MERGE (SQL)* (*Statements* documentation) and only describes the changes specific to Natural for DB2 Version 8.4.

```
MERGE INTO table-name [[AS] correlation-name]
[include-columns] USING source-table
ON search-condition
{WHEN matching-condition THEN modification-operation} ...
[ELSE IGNORE]
[NOT ATOMIC CONTINUE ON SQLEXCEPTION]
[QUERYNO integer]
```

Syntax Element	Description
MERGE INTO	**MERGE INTO Clause:** MERGE INTO initiates an SQL MERGE statement, which is a combination of an SQL INSERT and an SQL Searched UPDATE statement.
table-name	**Table Name:** Identifies the target of the INSERT or UPDATE operation of the MERGE statement.
[AS] *correlation-name*	**[AS] *correlation-name* Clause:** Specifies an alternate name for the target table. The alternate name can be used as qualifier when referencing columns of the intermediate result table.
include-columns	**Include Columns Clause:** Specifies a set of columns that are included, along with the columns of the target table, in the result table of the MERGE statement if it is nested in the FROM clause in a SELECT statement. The included columns are appended to end of the column list identified by the target table.
USING *source-table*	**USING *source-table* Clause:** Specifies the values for the row data to merge into the target table.
ON *search-condition*	**ON *search-condition* Clause:** Specifies join conditions between the *source-table* and the target table. Each column name in the search condition must name a column of the target table or *source-table*.
WHEN *matching-condition*	**WHEN *matching-condition* Clause:** Specifies the condition for which to perform the modification operation defined in the following THEN clause. See *matching-condition*.
THEN *modification-operation*	**THEN *modification-operation* Clause:** Specifies the operation to perform on the matches of the condition defined in the preceding WHEN clause. See *modification-operation*.
ELSE IGNORE	Specifies that no action is taken on source columns that do not match the condition specified in the WHEN clause.
NOT ATOMIC CONTINUE ON SQLEXCEPTION	**NOT ATOMIC CONTINUE ON SQLEXCEPTION Clause:** Specifies whether merge processing continues in case an error occurred during processing one row of a set of source rows.
QUERYNO *integer*	**QUERYNO *integer* Clause:** Specifies the number for this SQL statement that is used in EXPLAIN output and DB2 trace records.

source-table

```
table-reference

          ⎧    values-single-row   ⎫
(VALUES  ⎨                          ⎬        )
          ⎩    values-multiple-row  ⎭

 [AS] correlation-name (column-name,...)
```

Syntax Element	Description
table-reference	Specifies the source table to merge into the target table.
VALUES	VALUES introduces the specification of values for the row data to merge into the target table.
values-single-row	Specifies a single row of source data.
values-multiple-row	Specifies multiple rows of source data.
[AS] *correlation-name*	Specifies a correlation name for the source table.
column-name	Specifies a column name to associate the input data to the **UPDATE SET assignment clause** for an UPDATE operation or the VALUES clause for an INSERT operation.

matching-condition

```
[NOT] MATCHED [AND search-condition]
```

Syntax Element	Description
[NOT] MATCHED	Specifies the *modification-operation* to perform when an ON *search-condition* evaluates to true or not true (NOT can be specified optionally).
[AND *search-condition*]	Specifies an (optional) additional condition to evaluate to true before the *modification-operation* performs.

modification-operation

```
update-operation
DELETE
signal-operation
insert-operation
```

Syntax Element	Description
update-operation	Specifies that the matching target row is updated with the values assigned in the UPDATE SET assignment clause. An UPDATE operation is only allowed if the *matching-condition* evaluates to true.
DELETE	Specifies that the matching target row is deleted. A DELETE operation is only allowed if the *matching-condition* evaluates to true.
signal-operation	Specifies the SQL error to raise. A SIGNAL operation is only allowed if the *matching-condition* evaluates to true.
insert-operation	Specifies the rows to insert into the target table. An INSERT operation is only allowed if the *matching-condition* evaluates to not true.

signal-operation

```
SIGNAL SQLSTATE [VALUE] sqlstate [SET MESSAGE_TEXT = scalar-expression]
```

Syntax Element	Description
SIGNAL	Specifies the SIGNAL operation to perform when the *matching-condition* evaluates to true. DB2 sets an SQLCODE -438 if an error is raised by the SIGNAL statement.
SQLSTATE [VALUE] *sqlstate*	Specifies the SQLSTATE to be set by DB2. *sqlstate* is a 5-character alphanumeric constant or an alphanumeric variable. *sqlstate* values are assigned to SQLSTATE by DB2. See the appropriate DB2 documentation for recommended values.
[SET MESSAGE_TEXT = *scalar-expression*]	This optional clause specifies an error or warning message which is placed into the SQLEERRMC field of the SQLCA or which can be retrieved with the GET DIAGNOSTICS statement.

Examples - Example 3:

Merge sales data from the MSALES table into the MPRODUCT table. Demonstrate the MERGE operation with DELETE, UPDATE, INSERT and SIGNAL statements.

```
DEFINE DATA
LOCAL USING DEMSQLCA
LOCAL
1 V1 VIEW OF MPRODUCT
2 ID
2 NAME
2 INVENTORY
1  #M_TEXT  (A10) INIT <'Oversold: '>
   END-DEFINE
...
MERGE INTO MPRODUCT AS T
     USING (SELECT MSALES.ID        ,SUM(MSALES.SOLD) AS SOLD,
                     MAX(MCATALOG.NAME) AS NAME
             FROM MSALES, MCATALOG
             WHERE MSALES.ID = MCATALOG.ID
             GROUP BY MSALES.ID) AS S
             (ID,SOLD,NAME)
     ON S.ID = T.ID
     WHEN MATCHED AND T.INVENTORY = S.SOLD
                 THEN DELETE
     WHEN MATCHED AND T.INVENTORY < S.SOLD
                 THEN SIGNAL SQLSTATE '78000'
                       SET MESSAGE_TEXT =:#M_TEXT     || S.NAME
     WHEN MATCHED
                 THEN UPDATE SET T.INVENTORY = T.INVENTORY - S.SOLD
     WHEN NOT MATCHED
                 THEN INSERT VALUES(S.ID, S.NAME, -S.SOLD)
END TRANSACTION
END
```

Searched DELETE under Natural for DB2 Version 8.4

☐ **Note:** This is an extract of *Syntax 1 - Searched DELETE* in the chapter *DELETE (SQL)* (*State-ments* documentation) and only describes the changes specific to Natural for DB2 Version 8.4.

Syntax 1 - Extended Set

```
DELETE FROM table-name [period-clause] [correlation-name]
[include-columns [SET assignment-clause]]
[WHERE search-condition]
[FETCH FIRST row-limit

  [              {     RR     }           ]
  [      WITH    {     RS     }           ]        [SKIP LOCKED DATA] [QUERYNO integer]
  [              {     CS     }           ]
```

The `FETCH FIRST` clause limits the effects of the `DELETE` statement to a subset of qualifying rows. It corresponds to the `FETCH FIRST` clause of the `SELECT` statement described in *FETCH FIRST row-limit*.

Search Conditions under Natural for DB2 Version 8.4

Note: This is an extract of the chapter *Search Conditions* (*Statements* documentation) and only describes the changes specific to Natural for DB2 Version 8.4.

Comparison Predicate

```
{  scalar-expression     comparison scalar-expression    }
{  row-value-expression comparison row-value-expression  }
```

A comparison predicate compares two values or a set of values with another set of values.

In the syntax diagram above, *comparison* can be one of the following operators:

=	equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
<>	not equal to

II   Natural for Db2 for zIIP

This documentation describes the functionality and the use of Natural for Db2 for zIIP (NDZ). The documentation is organized under the following headings:

Overview	Information about the purpose and architectire of NDZ.
Unsupported and Restricted Functions	Information about date, time, timestamp, statements, and static execution.
Operation	Information about the NDZ operation and server commands.
Parameters for NDZ	Information about the NDZ parameter options.
Preparing Programs for Static Execution	Information about the necessary prerequisites to execute static programs with NDZ.

Related documentation

For installation instructions, refer to *Installing Natural for Db2 for zIIP* in the *Installation for z/OS* documentation.

21 Overview

Natural for Db2 for zIIP (NDZ) complements Natural for Db2 (NDB) Database Management Interface by adding the ability to run Db2 workloads on IBM System z Integrated Information Processors (zIIP).

Natural for Db2 for zIIP (NDZ) makes use of the zIIP capabilities by accessing Db2 through a remote DRDA protocol, instead of using conventional Db2 local attachments. Local calls to Db2 are not zIIP eligible, therefore accessing Db2 using the DRDA protocol enables Natural programs to run continuously on zIIP processors. Some workloads, e.g. I/O calls, are not zIIP eligible. zIIP ineligible workloads require the respective Natural programs to run temporarily on the general processors.

Java workloads running on z/OS are zIIP eligible, therefore Natural for Db2 for zIIP (NDZ) uses Java and Java Database Connectivity (JDBC) to access Db2 using the DRDA protocol. The following diagrams show an approximate comparison of the workload distribution when accessing Db2 using only NDB and when using NDB and NDZ at the same time.

Diagram X - Natural for Db2 (NDB) only:
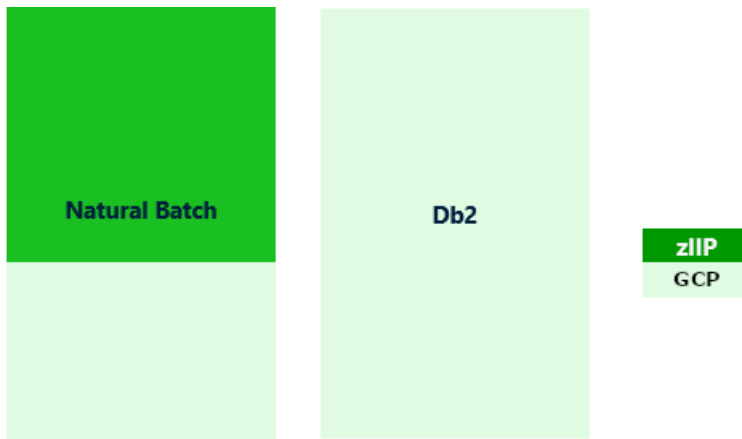
Diagram Y - zIIP utilization - Natural for Db2 (NDB) with Natural for Db2 for zIIP (NDZ)
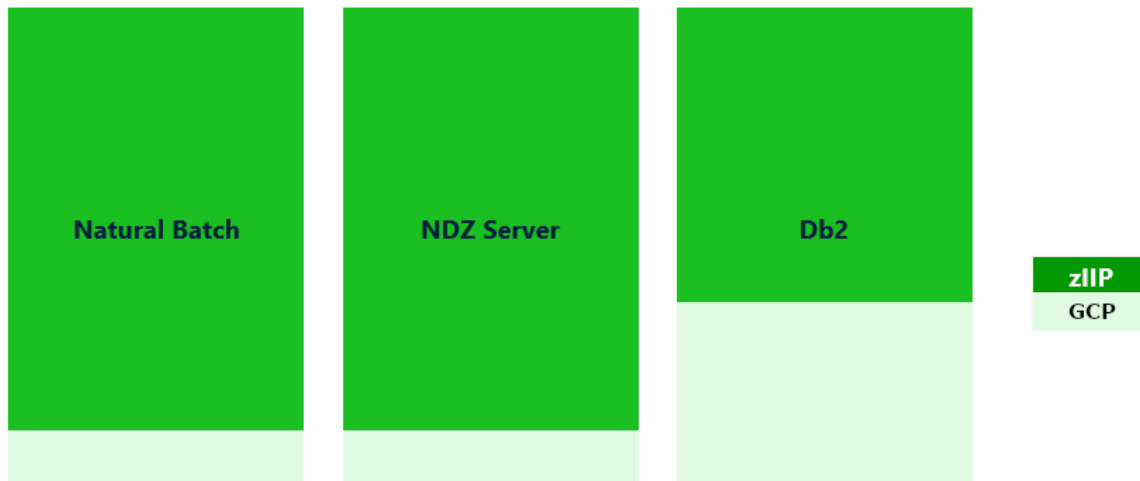
To benefit from Natural for Db2 for zIIP (NDZ) capabilities, you must ensure that enough System z Integrated Information Processors (zIIP) are available in the target environment. Running NDZ workloads without the required zIIP capacity can lead to increased general processor utilization, due to zIIP eligible workload running on general processors. Please refer to the section *Honor Priority* in the manual *z/OS MVS Planning: Workload Management*.

Architecture

Natural for Db2 for zIIP (NDZ) runs on its own address space. An instance of the NDZ Server is capable to handle requests from multiple Natural batch jobs.
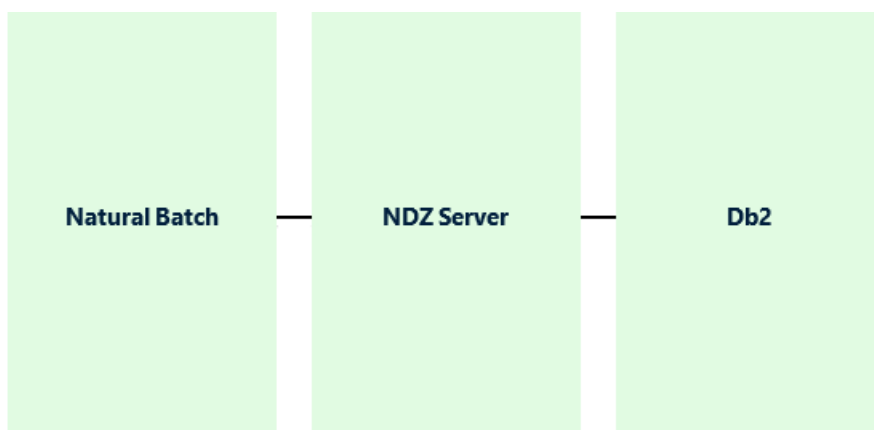
22 **Unsupported and Restricted Functions**

Date, Time and Timestamp

- Date
- Time
- Timestamp
- Timestamp with Timezone
- Floating Point

Date

When casting a `DATE` to `CHAR`, the date format resulting from the function will always have the ISO format. It will not honor the format selected in the `db2.properties`. This occurs due to a Db2 JDBC driver limitation. You can use the additional parameter in the `CHAR` function to change the DATE/TIME format.

```
CHAR(CURRENT DATE)
/* RESULT FORMAT EXAMPLE: 2023-05-01
```

Time

When casting a `TIME` to `CHAR`, the time format resulting from the function will always have the ISO format. It will not honor the format selected in the `db2.properties file`. This occurs due to a Db2 JDBC driver limitation. You can use the additional parameter in the `CHAR` function to change the DATE/TIME format

```
CHAR(CURRENT TIME)
/* RESULT FORMAT EXAMPLE: 01.01.01
```

Timestamp

When selecting timestamp data, if the fractional part of the source column has more than 9 digits, the result may be truncated to 9 digits. This occurs due to a Db2 JDBC driver limitation. Casting the result to `CHAR` solves the problem.

```
CURRENT TEMPORAL BUSINESS_TIME
/* RESULT EXAMPLE: 2011-02-28-01.01.01.123456789
CHAR(CURRENT TEMPORAL BUSINESS_TIME)
/* RESULT EXAMPLE: 2011-02-28-01.01.01.123456789012
CURRENT TIMESTAMP(12)
/* RESULT EXAMPLE: 2023-04-25-11.06.11.292553363
CHAR(CURRENT TIMESTAMP(12))
/* RESULT EXAMPLE: 2023-04-25-11.06.11.292553363769
```

Timestamp with Timezone

When selecting data from a column of the type `TIMESTAMP WITH TIMEZONE`, the time zone is not provided in the result. This occurs due to a Db2 JDBC driver limitation. Casting the result to `CHAR` solves the problem.

```
SELECT TIMESTAMP_W_TIMEZONE_COLUMN FROM ...
/* RESULT EXAMPLE: 2012-02-29-05.03.59.100000
SELECT CHAR(TIMESTAMP_W_TIMEZONE_COLUMN) FROM ...
/* RESULT EXAMPLE: 2012-02-29-07.03.59.100000+02:00
```

Floating Point

When selecting data from a `FLOATING POINT` column type, and the data has more than 16 digits, the round off rule is different from Natural for Db2 (NDB). Thus, the last digit value may be different from the NDB result.

```
/* Result with NDB
STDDEV SALARY = .9742432961021595E 04
/* Result with NDZ
STDDEV SALARY = .9742432961021594E 04
/* Result with NDB
STDDEV SALARY =  9742.432961021595
/* Result with NDZ
STDDEV SALARY =  9742.432961021594
```

Statements

- Statements with Restrictions
- Unsupported Statements

Statements with Restrictions

- DELETE
- INSERT
- UPDATE

- MERGE

DELETE

Positioned `DELETE` over a rowset using the clause `FOR ROW` or `WHERE CURRENT OF` is not supported by Natural for Db2 for zIIP. A positioned delete using the clause `WHERE CURRENT OF` is supported over a single row.

INSERT

Natural for Db2 for zIIP does not support multiple-row-insert.

UPDATE

Positioned `UPDATE` over a rowset using the clause `FOR ROW` or `WHERE CURRENT OF` is not supported by Natural for Db2 for zIIP. A positioned update using the clause `WHERE CURRENT OF` is supported over a single row.

MERGE

Natural for Db2 for zIIP does not support `MERGE` statements with multiple rows (*values-multiple-row*).

Unsupported Statements

The following Natural/Db2 SQL statements are not supported by Natural for Db2 for zIIP:

- `CALLDBPROC`
- `READ RESULT SET`
- `CONNECT TO`
- `GET DIAGNOSTICS`

Static Execution

Static preparation and execution of programs are experimental features and have restricted functionality.

23   Operation

Starting Natural for Db2 for zIIP

Issue the following MVS console start command to start the NDZ Server:

```
START NDZ STARTED TASK NAME
```

Stopping Natural for Db2 for zIIP

Issue the following MVS console stop command to stop the NDZ Server:

```
STOP NDZ STARTED TASK NAME
```

NDZ will perform the shutdown process after the last client disconnects. The following message displays in the MVS console if there are clients connected to the NDZ Server when the STOP command is issued:

```
NDZMAIN Stop requested. Waiting for clients to finish
```

To force NDZ termination despite the connected clients, add the F option to the stop command. Example:

```
STOP NDZ STARTED TASK NAME F
```

NDZ displays the following message in the console and performs the shutdown process:

```
NDZMAIN Forced stop requested
```

NDZ Server Commands

Use the MVS console modify command to issue commands to the NDZ Server:

```
MODIFY NDZ STARTED TASK NAME,COMMAND PARAMETER1, PARAMETER2, ...
```

NDZ Servers accept the following commands:

D CLI - Display clients

This command lists the clients that are currently connected to the NDZ Server in the STDOUT file. The output message has the following format:

```
------------------------------------------------
yyyy-MM-dd hh:mm:ss LIST OF CLIENTS CONNECTED
------------------------------------------------
 CLIENT#  JOBNAME  JOBID    USERID   PROGRAM
  cli#    name    job id user   prog.
------------------------------------------------
yyyy-MM-dd hh:mm:ss END OF LIST
------------------------------------------------
```

Where:

- CLIENT# = The client slot which this client occupies. Client numbers are in the interval between 1 and the value of CLIENTS parameter in the NDZ Parameter module.
- JOBNAME = The name of the client's job.
- JOBID = The ID of the client's job.
- USERID = The User ID that is associated with the client job.
- PROGRAM = The name of the Natural program that the client is executing at the moment.

P CLI=`CLIENT NUMBER` **– Stop client**

Release a client slot that is held by a non-executing client job.

> ⚠ **Important:** This command **SHOULD NOT** be used to cancel the execution of NDZ client jobs that are currently executing. Please use the appropriate MVS or JES2 command instead if you need to cancel a job. Using this command on a job that is currently in execution can lead to unexpected results including, but not limited to, loss of uncommitted data.

If a NDZ client job abnormally terminates or is cancelled, it usually disconnects from NDZ and releases the client slot it was occupying automatically. If the client job doesn't disconnect automatically, you can issue this command to release the client slot that is currently held by the job that is no longer running. To release a client slot:

1. Identify the Job ID of the client job.
2. Issue the `D CLI` modify command to list all clients that are connected to the NDZ Server.
3. Check the CLIENT# and JOBID columns to identify the client number of the job.
4. Issue the `P CLI=`*client number* modify command, where *client number* is the value from the CLIENT# column.

24 Parameters for NDZ

You must define the parameters for NDZ in the NDZ server started task and the ndz.properties USS file.

Parameters to Specify in the NDZ Started Task

- `PORT` – The number of the socket port. Accepted values are 0-65535.
- `PATH` – The home directory of NDZ.

Example:

```
//NDZ11 EXEC PGM=NDZNUC11,REGION=0M,
//PARM=('PATH=/u/nat/ndz/dev/ndz11','PORT=4932') ↵
```

Parameters to Specify in the ndz.properties File

- `java.home` – The Java installation directory.
- `db2.home` – The Db2 installation directory.
- `ndz.maxClients` – The maximum amount of client connections. Accepted values are 1-999. The default value is 10.
- `ndz.initDb2Connection` – The number of initial connections to Db2. The default value is 2.
- `ndz.bufferLentgh` – The length of the client buffer in MB. Accepted values are 1-5000. The default value is 2.
- `ndz.staticPath` – Path(s) for the static profiles.

25 Preparing Programs for Static Execution

If you want to prepare a program for static execution with NDZ, the program must be prepared for static execution with NDB beforehand. Please refer to Dynamic and Static SQL Support for more information.

NDZ uses the temporary assembler program created in the *Generation Procedure (CMD CREATE)* (Step 1 in the following diagram) and relies on the information added to the Natural object during the the *Modification Procedure (CMD MODIFY)* (Step 3). In addition, the temporary assembler program must be converted into a temporary SQLJ program (Step Z1). Then, the Db2 SQLJ translator generates the SQLJ Profile using the temporary SQLJ program as input (Step Z2). During the final step, the Db2 bind is executed using the SQLJ Profile.

Currently it is not possible to prepare programs for static execution only for NDB or only for NDZ. Both preparations must be executed. Nevertheless, you do not need to complete steps 2, 3 and 4 for programs that are already prepared for static execution with NDB. In this case, for step 1, you must ensure that you use the same list of programs that was used when the program was prepared for static execution with NDB. In addition, you must also make sure that the same DBRM name is used in step Z3. Having a different list of programs and/or a different DBRM name will lead to errors when executing the programs, because the information about the static execution located in the Natural objects will not be accurate. The following chart represents an overview of the necessary steps to prepare a program for static execution with NDZ.
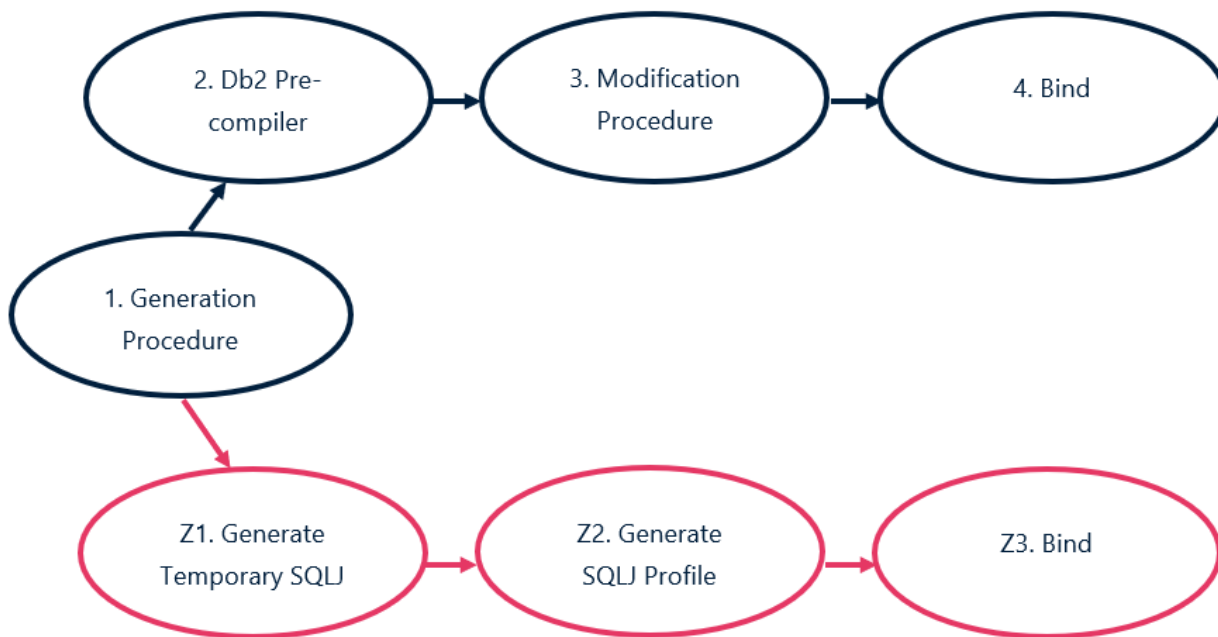
Here is detailed information about the steps:

Step 1 - Generation Procedure

This step is described in the *Generation Procedure: CMD Create Command* **section.**

Step 2 - Db2 Pre-compiler

This step is described in the *Precompilation of the Generated Assembler Program* **section.**

Step 3 - Modification Procedure

This step is described in the *Precompilation of the Generated Assembler Program* **section.**

Step 4 - Bind

This step is described in the *Bind the Precompiled DBRM* **section.**

Step Z1 - Generate Temporary SQLJ

The temporary assembler program generated in "Step 1 - Generation Procedure" is converted to a SQLJ program.

The Java program `com.softwareag.ndz.sqljgen.SQLJGenerator` is provided to perform this conversion.

Input Parameters

`DBRM` name - the name of the generated SQLJ program. This name has to be the same name that was used in Step 1.

DD Names

`INPUT` - The temporary assembler program that was generated in Step 1.

`OUTPUT` - The temporary SQLJ program.

Step Z2 - Generate SQLJ Profile

The Db2 SQLJ Translator (sqlj) generates a SQLJ profile using the SQLJ program generated in the previous step as input. The provided shell script `genprof.sh` executes this task.

Step Z3 - Bind

The Db2 SQLJ Profile binder (db2sqljbind) binds packages for the SQLJ Profile generated in the previous step. The provided shell script `bind.sh` executes this task.

Shell Scripts for static preparation

`setenv.sh` - Set the environment variables for Java, Db2 and NDZ.The variables are required by other scripts.

`genprof.sh` - Generate profile for static execution. Syntax:

```
genprof ndz_path program
```

Where *ndz_path* is the NDZ installation path and *program* is the SQLJ program that is used to generate the profile.

`bind.sh` - Bind a static profile to Db2. Syntax:

```
bind NDZ_PATH url user password program bind_opts
```

Where ...

- *NDZ_PATH* is the installation path of NDZ.
- *url* is the URL for Db2 access.
- *user* is the username for Db2 access.
- *password* is the password for the selected username for Db2 access.
- *program* is the Natural program that is accessing Db2.
- *bind_opts* is a list of bind options. Every option has to be seperated by a space and carries the same function as Db2 precompile and bind options. For more information, see the BIND PACKAGE command section in the IBM Db2 documentation.

III Natural for VSAM

This documentation describes the various aspects of Natural when used in a VSAM environment.

General Information	Special considerations on the environments supported by Natural for VSAM, known incompatibilities and constraints when using Natural for VSAM, terms used in this documentation, and on error messages related to Natural for VSAM.
Introduction to Natural for VSAM	Components of Natural for VSAM, structure of the Natural interface to VSAM.
Customizing Natural for VSAM	Description of the Natural for VSAM parameters, macros and I/O modules.
Operation	Information on operational aspects like how to invoke Natural for VSAM, `OPEN`/`CLOSE` processing, Natural file access, buffers for memory management, and application programming interfaces.
Natural Statements and Transaction Logic with VSAM	Special considerations on the use of Natural statements and system variables with VSAM. In addition, the Natural transaction logic with VSAM is discussed.

Related Documentation

For installation instructions, see Installing Natural for VSAM in the *Installation for z/OS* and *Installation for z/VSE* documentation.

For various aspects of accessing data in a database with Natural, see also *Database Access* in the Natural *Programming Guide*.

For a list of the abend codes of Natural for VSAM, refer to *Natural for VSAM Abend Codes* (in the Natural *Messages and Codes* documentation).

26 General Information

Purpose

With the Natural interface to VSAM, a Natural user can access data stored in VSAM files. As a prerequisite, the current version of Natural for Mainframes must be installed.

In general, there is no difference between using Natural with VSAM and using it with Adabas or any other supported database management system. The Natural interface to VSAM allows Natural programs to access VSAM data, using the same Natural DML statements that are available for Adabas. Therefore, programs written for VSAM can also be used to access, for example, Adabas databases.

All operations requiring interaction with VSAM are performed by the Natural interface to VSAM.

Environment-Specific Considerations

Natural for VSAM is fully ESA- and z/OS Parallel Sysplex-compliant. It runs in batch mode or under the online environments CICS, Com-plete and TSO. Under CICS, it also runs in conversational or pseudo-conversational mode.

Natural for VSAM supports the following types of VSAM file:

- KSDS,
- ESDS,
- RRDS,
- VRDS.

Under z/OS, Natural for VSAM supports the data set access modes record-level sharing (RLS) and DFSMS Transactional VSAM Services (DFSMStvs).

The Natural system files `FNAT`, `FUSER`, `FDIC`, `FSPOOL` and `FSEC` can also be located on VSAM system files. For VSAM system files, Natural for VSAM uses the multi-fetch option to speed up the process of loading objects into the buffer pool.

Natural for VSAM supports local shared resources (LSR) under TSO and in z/OS and z/VSE batch modes. For CICS and Com-plete, the appropriate file definition tools must be used. The LSR option for VSAM files improves the performance of random access.

Natural for VSAM supports Create/Loading Mode for empty files under TSO as well as in batch mode.

Natural for VSAM supports the following types of Data Table under CICS z/OS:

- User-Maintained Data Tables (UMT),

- CICS-Maintained Data Tables (CMT),

- Coupling Facility Data Tables (CFDT).

It also supports data set name sharing (DSN) under TSO, and batch-mode processing in z/OS and z/VSE, in particular to access data sets using a defined path.

Natural for VSAM supports extended-format data sets for all types of VSAM data set organization. There are, however, restrictions for ESDS, RRDS and VRDS which result from the use of the Natural system variable `*ISN` (see *Database-Specific Information*) and its internal size limit of 4 bytes.

Natural for VSAM with Natural Security

Since Natural Security supports the `FSEC` system file as VSAM system file, the following restrictions must be considered:

- Generation of ETIDs is disabled.

- Logging of maintenance actions is disabled.

- Password history is disabled.

- Definition of utility profiles is disabled.

Integration with Predict

Predict, Software AG's open, operational data dictionary for fourth-generation-language development with Natural, is a central repository of application metadata and provides documentation and cross-reference features. Predict lets you automatically generate code from definitions, enhancing development and maintenance productivity.

Since Predict supports VSAM, direct access to VSAM files is possible via Predict and information from VSAM can be transferred to the Predict dictionary to be integrated with data definitions for other environments.

VSAM physical and logical views can be incorporated and compared, new VSAM views can be generated, and Natural views can be generated and compared. All VSAM-specific data types and the referential integrity of VSAM are supported. See the *Predict* documentation for details.

Terms Used in this Documentation

Term	Explanation
CFDT	Coupling Facility Data Tables
CMT	CICS-Maintained Data Tables
DDM	Natural data definition module
DFSM	Data Facility Storage Management Subsystem
DFSMStvs	DFSMS Transactional VSAM Services
Front-end	When used in this documentation, the term "front-end" refers to the driver in conjunction with the Natural parameter module.
LSR	Local Shared Resources
NVS	This is the product code of Natural for VSAM. In this documentation the product code is often used as prefix in the names of data sets, modules, etc.
UMT	User-Maintained Data Tables

Messages Related to VSAM

The message number ranges of Natural system messages related to VSAM are 3500-3599.

For a list of the abend codes that may be issued by the Natural interface to VSAM, see *Natural for VSAM Interface Abend Codes* in the Natural *Messages and Codes* documentation.

27 Introduction to Natural for VSAM

This section describes the components and the structure of the Natural interface to VSAM.

Components of Natural for VSAM

The Natural interface to VSAM consists of the following components:

- The `NVSNUC` module, which is mandatory, environment-independent, and delivered as a load module only.

- The Natural parameters specific to VSAM, defined in the Natural parameter module.

- The I/O module, which is mandatory, differs depending on the actual environment, and is delivered in source form only.

- The modules necessary when running with VSAM system files; they are optional and delivered as load modules only.

- The user exits.

- Callable system services.

Natural for VSAM is fully (E)LPA or SVA-compliant for multiple environments (for example, CICS, Com-plete and batch). The Natural parameter module and the appropriate I/O module must be linked to the front-end module.

Structure of the Natural Interface to VSAM

1. VSAM system-file handling for FNAT, FUSER and FDIC.
2. VSAM system-file handling for FSPOOL.
3. VSAM system-file handling for FSEC.
4. VSAM system-file handling for Natural ISPF.
5. IBM's record-level sharing (RLS) query routine to support RLS=CHECK, z/OS only (not CICS).

28 Customizing Natural for VSAM

The Natural parameters in a VSAM environment are defined in one location:

- the Natural standard parameters, contained in the Natural parameter module; see *Building a Natural Parameter Module* in the *Operations* documentation,

- the Natural parameters specific to VSAM, also contained in the Natural parameter module; see parameter macro `NTVSAM` in the *Parameter Reference* documentation.

The Natural parameter module can be edited to conform to your site standards, and then assembled and linked using the appropriate jobs (see *Installing Natural for VSAM Installation for z/OS* and *Installation for z/VSE* documentation).

Customizing the Natural Parameter Module

To be able to run Natural in a VSAM environment, you must include the profile parameter `VSIZE`, the `NTDB` and the `NTVSAM` macro in your Natural parameter module (see the section *Installing Natural for VSAM* in the *Installation for z/OS* and *Installation for z/VSE* documentation).

For an Adabas system file:

```
VSIZE=72,
NTDB VSAM,vsam-dbid
NTVSAM
```

For a VSAM system file:

```
VSIZE=160,

FNAT=(vsam-dbid,fnr,dd-name),
FUSER=(vsam-dbid,fnr,dd-name),
FDIC=(vsam-dbid,fnr,dd-name),
FSPOOL=(vsam-dbid,fnr,dd-name),
FSEC=(vsam-dbid,fnr,dd-name)

NTDB VSAM,vsam-dbid
NTVSAM ... SFILE=ON,...
```

`dd-name` is the logical name (DD or DLBL) of the system file; see also *Installing Natural for VSAM* in the *Installation for z/OS* and *Installation for z/VSE* documentation.

> **Note:** If you use VSAM system files with Natural ISPF, see also the *Natural ISPF* documentation.

Below is information on:

- VSIZE Parameter
- NTDB Macro

- NTVSAM Macro

VSIZE Parameter

`VSIZE` is a Natural profile parameter which can also be specified dynamically. It is used to specify the size of the Natural buffer area for VSAM and defines the maximum memory usage for the internal tables of the Natural interface to VSAM; the actual sizes of these tables depend on the values set in the Natural parameter module (see *Assembling the VSAM-specific Natural Parameter Module*).

Possible values are 0, 1 - 512 KB.

If you use the default values specified in the Natural parameter module, the value of the `VSIZE` parameter must be at least 72 KB.

If `VSIZE` is set to `0`, Natural for VSAM is not available and a corresponding error message is returned when trying to access VSAM files. Disabling Natural for VSAM leads to slight performance improvements because of skipping the initialization, relocation and roll efforts of the Natural interface to VSAM.

NTDB Macro

The `NTDB` macro is used to specify the database numbers that relate to VSAM files; that is, the logical assignments available for Natural.

The value range of `NTDB` parameters is described in the Natural *Parameter Reference* documentation.

> **Note:** Ensure that the DBIDs selected in the `NTDB` macro for VSAM do not conflict with DBIDs selected for other database management systems.

NTVSAM Macro

The `NTVSAM` macro is used to specify the VSAM specific parameters.

The value range of the `NTVSAM` keyword subparameters is described in the Natural *Parameter Reference* documentation

Assembling the VSAM-specific Natural Parameter Module

If the default values supplied in the Natural parameter module do not meet your requirements, you can change the parameter values to suit your environment. The individual VSAM-specific parameters contained in the Natural parameter module are described in the following section.

The VSAM-specific Natural parameter module is created by assembling the macro:

- `NTVSAM`

and optionally one or more of the following macros:

- `NTVEXIT`
- `NTVLSR`
- `NTVTVSD`

If more than one macro is specified, the `NTVSAM` macro must be specified first; further macros after the `NTVSAM` macro can be specified in any order.

Natural I/O Modules for VSAM

The Natural I/O module for VSAM depends on the actual environment in use.

All available I/O modules are delivered in source form so you can make site-specific modifications and use environment-specific macros and/or precompilers. The I/O module must be linked to the Natural parameter module.

The I/O modules available are:

- NVSCICS Module
- NVSMISC Module

NVSCICS Module

The `NVSCICS` module is required for CICS under z/OS or z/VSE. The module contains the following parameter:

&FCTRELI - Indicator of Reliable Remote FCT Entries

The `&FCTRELI` parameter indicates whether the key length and record size of a remote file are correctly defined in the FCT entry of the Application Owning Region (AOR).

Possible values	Default value
0 or 1	0

When this parameter is set to `1`, `NVSCICS` assumes a correct FCT entry.

When this parameter is set to `0`, `NVSCICS` issues dummy commands to force opening of the file in the File Owning Region (FOR) region and then repeats inquiring for the real values.

If the FCT entry does not contain a key length definition, `NVSCICS` uses the key length of the corresponding VSAM DDM.

NVSMISC Module

The `NVSMISC` module is required in all environments except for CICS. The module mainly consists of the name of the relocatable module for z/VSE and the `NVMMISC` macro, which is used to generate the `NVSMISC` I/O interface according to your operating system and/or TP-monitor environment.

`NVSMISC` is specified as follows:

```
name NVMMISC NONRLS=value TIMEOUT=value DSECTS=value DEFER=value COMMIT=value ERROR=value
HFACTOR=value READINT=value SMARTS=value TVS=value
```

The *name* of the relocatable module must be 8 characters long; the default name is `NVSMISCD` (z/VSE only).

The individual parameters are described in the following section; specify these parameters according to your requirements.

NONRLS - Switch from RLS to Non-RLS Mode

This parameter is ignored under z/VSE.

When Natural for VSAM issues an `RLS-OPEN` for an RLS file and this file has already been opened in non-RLS mode in this z/OS session, this parameter specifies whether Natural for VSAM issues an open retry in a non-RLS mode, or whether an open error occurs.

Possible values	Default value
YES/NO	YES

TIMEOUT - Timeout in Seconds for an RLS Request

This parameter is ignored under z/VSE.

This parameter specifies the time in seconds Natural for VSAM is waiting to obtain a lock on a Natural for VSAM record when a lock on the record is already held by another user. For further details refer to the IBM manual *z/OS DFSMS* Version 1.6 or higher, *Macro Instructions for Data Sets*.

Possible values	Default value
0 - 10	0

DEFER - Defer Writes in LSR Pools

This parameter only applies in batch mode and under TSO.

This parameter specifies whether write operations to disk are to be deferred in the LSR pool. If so and if the LSR pool becomes full, Natural for VSAM writes to disk those 5% of the pool area which have not been used for the longest time.

Possible values	Default value
YES/NO	NO

DSECTS - List VSAM System DSECTs

The `DSECTS` parameter specifies whether the VSAM system DSECTs are to be listed or not.

Possible values	Default value
YES/NO	NO

COMMIT - Support of Buffer Flush for LSR Pools

This parameter only applies in batch mode and under TSO.

The `COMMIT` parameter specifies whether all non-committed updates in any LSR pool are to be written to disk with each `END TRANSACTION` statement of a user program.

Possible values	Default value
YES/NO	NO

> **Note:** The specification of `COMMIT=YES` increases the I/O rate considerably.

ERROR - Issue Initialization Error

This parameter issues a Natural initialization error if any DD or DLBL card is omitted in the runtime JCL (see also the macro `NTVLSR`).

Possible values	Default value
YES/NO	YES

If set to `NO`, processing is continued and Natural for VSAM will be initialized.

HFACTOR - Factor for Hiperspace Buffers

The `HFACTOR` parameter specifies a factor for the creation of ESO Hiperspace buffers. When initializing such a Hiperspace, the corresponding `BLDVRP` request may lead to a Natural error message, in which case the value of `HFACTOR` must be reduced.

Possible values	Default value
0 - a value where a corresponding Natural error message is returned	100

READINT - Read Integrity for Upgrade Set

The `READINT` parameter specifies whether read integrity for an upgrade set should be granted or not.

Possible values	Default value
YES/NO	NO

SMARTS - Support of SMARTS and Com-plete

The `SMARTS` parameter is required if installing Natural for VSAM under SMARTS and/or in a Complete environment.

Possible values	Default value
YES/NO	NO

TVS - Support of DFSMS Transactional VSAM Services (DFSMStvs)

This parameter is ignored under z/VSE. The `TVS` parameter specifies the support of DFSMStvs in a z/OS environment.

Possible values	Default value
YES/NO	NO

29 Operation

This section provides information on various operational aspects of Natural for VSAM:

Invoking Natural for VSAM

If the Natural interface to VSAM is available, it is initialized when you start a Natural session. It can be switched off by setting the `VSIZE` parameter to `0` (see also the relevant description in the section *Natural for VSAM Parameters*).

OPEN/CLOSE Processing

In this section, VSAM files means both VSAM user files and VSAM Natural system files.

Database `OPEN`/`CLOSE` processing is controlled by the Natural parameter `OPRB`, which is described in *Profile Parameters* in the Natural *Parameter Reference* documentation.

Instead of using the `OPRB` parameter, you can also use the `NTOPRB` macro of the Natural parameter module, which is described in Parameter Modules in the Natural Parameter Reference documentation.

An `OPEN`/`CLOSE` error must be followed by the NAT3539 error message. In a TP environment, the NAT3516 error message can also occur during an active Natural session if the file is closed.

> **Note:** For dynamic `OPEN` handling within a session, you can use the application programming interface **USR2008N**.

The section below covers the following topic:

- OPRB Parameter for VSAM Databases

OPRB Parameter for VSAM Databases

The `OPRB` parameter is not applicable under CICS or Com-plete, because in these environments, the TP monitor controls the `OPEN`/`CLOSE` processing of VSAM files.

By default, that is, without the `OPRB` parameter being specified, VSAM files are opened for input/output so that they can be read and/or updated.

If you want all used VSAM files to be opened for input only, you specify the `OPRB` parameter using the following syntax:

```
OPRB = (.ALL)
```

With this syntax, you specify an `OPEN` request for *all* VSAM files to be addressed. All files are opened for input only; individual files, however, are only opened when they are actually addressed by a given program.

> **Note:** If you want all VSAM system files to be opened for input, you have to set the Natural profile parameter `ROSY=ON`; see also the relevant section in the Natural *Parameter Reference* documentation.

If you want to open VSAM files for input (`I`) or output (`O`) per DBID, use the following syntax:

$$
OPRB = (DBID = nnn, \left\{ MODE = \begin{Bmatrix} I \\ O \\ string\,;\ \dots \end{Bmatrix} [,\, string\,;\ \dots] \right\} [,\ \dots]\ )
$$

With `MODE`, you specify a global default handling for `DBID` *nnn*.

If you do not want to specify a default handling per DBID or if, for some VSAM files, you want an input/output handling other than the default one, you specify the *string* parameter in the appropriate way.

The DBID must be defined with the `NTDB` macro as a VSAM DBID, and *string* varies depending on the operating system (see below).

Important: If several strings are to be defined, a semicolon (;) must be specified as delimiter character. If not, the semicolon must be omitted.

Under z/OS

Under z/OS, you specify the *string* as follows:

$$
\begin{Bmatrix} FNR = nnn \\ DD = dd\text{-}name,\ TYP = \begin{Bmatrix} K \\ E \\ R \\ P \end{Bmatrix} \end{Bmatrix} \begin{Bmatrix} ,\ O \\ ,\ I \end{Bmatrix} \begin{Bmatrix} ,\ B \\ ,\ A \end{Bmatrix} [,R]
$$

The specified VSAM files must be defined as DDMs. However, instead of specifying the file number of the Natural DDM that corresponds to the VSAM file to be addressed, the *dd-name* and type (`KSDS`, `ESDS`, `RRDS`, or `PATH`) of this file can be specified directly, which saves you from having to look into the DDM first.

Individual files can be opened for output (option **O**), input (option **I**), opened **before** they are actually accessed (option **B**), or when they are accessed for the first time (option **A**), opened as reusable file (option **R**).

For performance reasons, it is sometimes desirable to modify the VSAM STRNO (string number) parameter to provide more index and data buffers. By default, Natural uses string number 3 for input processing and string number 5 for output processing. Since STRNO is specified in the JCL, both values can be modified with the AMP parameter in the corresponding DD card.

Under z/VSE

Under z/VSE, no string number can be specified in the JCS. Therefore, the syntax has been enhanced to be able to specify a string number with the OPRB parameter, where *nn* can be in the range from 1 to 10. Thus , *string* represents:

```
┌                         ┐
│        FNR = nnn    ┌ K ┐ │
│                     │ E │ │   ┌   O ┐ ┌   B ┐        [,STRNO
│ DD = dd-name , TYP = │ R │ │ { , I } { , A } [,R]     = nn ]
│                     └ P ┘ │
└                         ┘
```
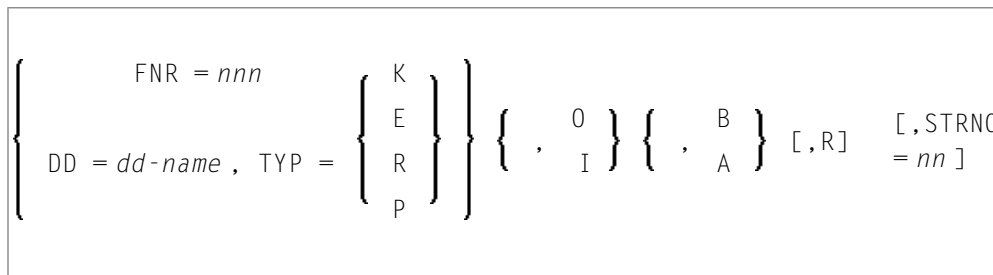
Sample OPRB Specification

The following OPRB example opens the specified files for input, while files not specified are opened for output by default:

```
OPRB=(DBID=254,MODE=I)
```

or

```
OPRB=(DBID=254,FNR=21,I,A;FNR=22,I,A)
```

The VSAM DBID and FNR as specified in the DDM are required. Option I specifies the corresponding FNR to be opened for input; option A specifies the corresponding FNR to be opened only if the file is accessed by a Natural program.

The corresponding NTOPRB macro example would be:

```
NTOPRB 254,'MODE=I'
```

or

```
NTOPRB 254,'FNR=21,I,A';'FNR=22,I,A'
```

Natural File Access

The Natural interface to VSAM supports VSAM entry-sequenced data sets (ESDS), key-sequenced data sets (KSDS), relative record data sets (RRDS), variable relative record data sets (VRDS), and paths for alternate indexes.

To enable Natural to access VSAM files, a Natural DDM is required for each VSAM file that is to be made accessible to Natural programs.

The section below covers the following topics:

- Natural Data Definition Modules (DDMs)
- SYSDDM Main Menu
- Catalog DDM
- Edit DDM
- Restrictions with DDM Generation as Compared to Adabas

Natural Data Definition Modules (DDMs)

A data definition module (DDM) must be set up for each file. DDMs are created and maintained with Predict (see the Predict documentation for details) or with the Natural utility SYSDDM; they are stored in the Natural dictionary system file (FDIC).

With VSAM, in addition to logical Natural DDMs, also VSAM user DDMs can be created from one physical DDM.

If you do not have Predict installed, use the SYSDDM utility to create DDMs from VSAM files. The SYSDDM utility is described in the Natural *Editors* documentation; the parts of it relevant to VSAM are described in the following sections.

All DDMs used within a session are located in the Natural buffer pool. This increases performance and enables synchronization of DDM usage across multiple sessions.

SYSDDM Main Menu

The following functions on the main menu of the SYSDDM utility are relevant to Natural for VSAM:

Function	Explanation
Catalog DDM	The DDM currently in the work area is cataloged, making it available for use within Natural applications. The DDM must have previously been placed in the work area by a READ command (see also *Editor and System Commands* in the *SYSDDM Utility* documentation), or have been entered by using the **Edit DDM** function described below. Below are further details about *Catalog DDM*.

Function	Explanation
Edit DDM	Reads a DDM from the system file `FDIC` and into the `SYSDDM` work area, where it can be edited.
List DDMs	Displays a single DDM source (DDM editor *not* invoked) or a list of DDMs. The display format and options are identical to those of the `LIST DDM` command (see also *Editor and System Commands* in the *SYSDDM Utility* documentation).
Copy DDM to Another FDIC File	One or all DDMs can be copied to a different Natural system file (`FDIC`) and/or to a different database. This is, for example, necessary during conversion of a Natural application from test to production status. In addition to the DDM name, DBID and FNR, with Natural for VSAM the file type `V` must be specified, as well as the DD/FCT name of the Natural system file `FDIC`, if the `FDIC` file is a VSAM file.
List DDMs with Additional Information	Displays a list of the DDMs stored in the specified `FDIC` system file. From the list, you can select individual DDMs for further processing. This function differs from the **List DDMs** function in that it displays additional items of information on the individual DDMs. The information displayed includes file name, DBID, file number, DDM length, security type (with Natural Security only), file type (that is, `LOG.DDM`, `PHY.FILE`, `LOG.FILE` or `USERDDM` for VSAM DDMs) and remarks as, for example, the VSAM file organization (KSDS, VRDS, RRDS, ESDS); see the section SYSDDM Utility in the Natural *Editors* documentation for details.
Delete DDM	Deletes a previously cataloged DDM from the Natural system file `FDIC`. The DDM remains in the work area. **Important:** If a DDM is deleted with `SYSDDM`, the corresponding Natural Security file profile is automatically deleted.

The following parameters relevant to Natural for VSAM can be specified for the various functions:

Parameter	Explanation
`DDM Name`	The name of the DDM to be processed.
`FNR`	The file number of the DDM to be processed.
`DBID`	The database which contains the DDM to be processed.
`Replace`	If `Y` is entered, DDMs which are being copied or cataloged and which are already existent are replaced. If `N` is entered, such DDMs are not replaced.
`FDIC Type`	The type of the system file `FDIC`.
`DDM Type`	The type of the DDM. For VSAM, the type must be `V`.
`DBID Type`	The type of the DDM. For VSAM, the type is `V`.

Catalog DDM

A DDM can be cataloged by either using function code `C` in the `SYSDDM` main menu or entering the `CATALOG` command in the **Command** line of the DDM maintenance editor.

File name and file number are required for this function. With Natural for VSAM, a DBID assigned to VSAM must be specified. If no DBID is entered, it is assumed to be 0 and is generated dynamically at execution time based on the DBID of the Natural user system file (`FUSER`) in use (see also the description of the `UDB` parameter in the section *Profile Parameters* in the Natural *Parameter Reference* documentation).

If a DBID assigned to VSAM is specified (and `V` for VSAM in the field **Type of this DDM**), `SYSDDM` prompts you for additional information.

> **Note:** The actual DBID assignments for VSAM is made with `NTDB` macros when assembling the Natural parameter module; see *Installing Natural for VSAM* in the *Installation for z/OS* and *Installation for z/VSE* documentation.

Additional Options for VSAM Files

If the DDM is to access a VSAM file, an additional screen, requiring the entry of additional VSAM options, is displayed:

```
11:24:04            ***** NATURAL SYSDDM UTILITY *****            2006-05-25
                    - Catalog a VSAM file/DDM -


    DBID 254   FNR  12    DDM AUTOMOBILES-VS                      Def seq
    -------------------------------------
    VSAM file information

    VSAM file name ............ AUTO
    VSAM View ...........(Y/N) N
    Logical related to FNR ....
    User defined prefix .......

    VSAM file organization

    KSDS, ESDS, RRDS, VRDS (K,E,R,V) .. K

    Compress file ........(Y/N) N
    Zones X'0C' / X'0F' (C/F) F
```

The additional options for VSAM files consist of two parts: **VSAM File Information** and **VSAM File Organization**.

VSAM File Information Options

Option	Explanation	
VSAM file name	The DDNAME/FCT entry as defined to the TP monitor or when using batch mode, for example: `//PERSON DD ...` where PERSON would be entered under **VSAM file name**.	
VSAM View (DDM)	Indicates whether this DDM represents a logical user DDM or a physical DDM.	
	Y	Indicates that the DDM represents a logical DDM, which means that it does not necessarily correspond to the physical layout of the VSAM file. A logical DDM must use the same file number as the physical DDM from which it is derived, and the corresponding physical DDM must exist at the time the user DDM is invoked during execution. The short names of the logical DDM must be identical to those defined in the physical DDM. The sequence of fields within the DDM can be different from the physical sequence. The primary-key field must not be deleted from the DDM. Since the logical DDM is a subset of the physical DDM, the corresponding subsets of the underlying VSAM file appear to the user as independent files with different record layouts. When processing a logical DDM, the user obtains records only from the corresponding subset and not from any other subset contained in the same physical VSAM file.
	N	Indicates that the DDM represents a physical DDM. Only one DDM with a given file number can be used as the physical DDM for a VSAM file. This physical DDM is used internally by Natural to calculate field offsets.

Logical DDMs are used to define different record types in a physical VSAM file. At DDM generation, these record types are identified by specifying a prefix for the primary key.

If a logical DDM is read, only records whose key begins with the specified prefix are returned from the VSAM file. Records beginning with any other prefix are ignored. If not specified otherwise, the prefix corresponds to the logical file number.

A different prefix must be assigned to each logical DDM. Natural automatically links the prefix with the logical key. The field layout in the logical DDM need not be the same as in the physical DDM.

The following two options are used only if the DDM represents a logical file which is to be derived from a physical VSAM file.

Option	Explanation
Logical related to FNR	This option is used to enter the file number of the physical DDM from which the logical file or DDM is derived. A logical DDM corresponds to a record type which is controlled by a prefix. Several logical record types can be contained in a physical VSAM file. The record types are distinguished by a prefix which determines which records are to be processed. See the **example** below.
User defined prefix	The prefix value which is to be assigned for the logical file. The default prefix value is the logical file number (length 3).

Example of Logical Related to FNR

Physical Data Set				
Key				
{ X1234 X2345 X3456 }	DDM1	PREFIX ↵ = X	FNR ↵ = 10	
{ Z1234 Z1209 Z9000 }	DDM2	PREFIX ↵ = Z	FNR ↵ = 11	
Read DDM1 with key				
Display key				
results in:				
	Key 1234 2345 3456			

VSAM File Organization Options

Option	Explanation		
KSDS, ESDS, RRDS, VRDS	The type of VSAM file:		
	K	KSDS file (default)	
	E	ESDS file	
	R	RRDS file	
	V	VRDS file	
Compress file	Specifies whether the file is to be compressed or not.		
	N	Indicates that the file is not to be compressed. The file is written in the maximum length (that is, the length of all fields within this file) as defined in `SYSDDM` or Predict. `N` is the default value.	
	Y	Indicates that the file is to be written in variable record length. During compression, the record is scanned backwards for default values, which are blank for alphanumeric fields, low values for binary fields, low values with a zone for packed fields and `X'F0'` for numeric fields. Compression stops as soon as the first non-default value is detected or the first descriptor is found. The new computed length is used to write the record to the file; this applies to KSDS and ESDS files only. Compression of trailing null values in VSAM records minimizes the space required for VSAM records. The application programming interface `USR0100N` in the library `SYSEXT` is provided to be able to maintain the logical record length by a Natural program.	
Zones X'0C' / X'0F'	In Adabas all positive packed values have `X'0F'` as a zone. This value could be different in VSAM. `F` Indicates that all packed data are written to the VSAM file with the zone `X'0F'`. This is the default. `C` Indicates that all packed values are written to the VSAM file with the zone `X'0C'`.		

Edit DDM

To edit the DDM currently loaded in the work area, you can use the DDM editor of the `SYSDDM` utility. If no DDM has been read into the work area, an empty screen is displayed, allowing the manual entry of a DDM definition.

Instead of entering a complete DDM manually, you can read an existing DDM definition into the work area, by entering `EDIT` *ddm-name* in the DDM editor **Command** line. This DDM can be modified and cataloged under a different name.

📄 **Note:** When you modify a DDM, all objects which reference this DDM have to be cataloged again.

DDM Editor

Example:

```
11:26:09                      ***** EDIT DDM (VSAM) *****                    2007-02-25
DDM Name EMPLOYEES-VS                         Def.Seq.        DBID   254 FNR      1
Command
I T L DB Name                                 F Leng  S D Remark
- - - -- ------------- top ------------- - ----- - - -----------------------
   1 AA PERSONNEL-ID                          A 8.0      P
 *      CNNNNNNN
 *      C=COUNTRY
 G 1 AB FULL-NAME
   2 AC FIRST-NAME                            A 20.0  N
   2 AD MIDDLE-NAME                           A 20.0  N
   2 AE NAME                                  A 20.0    A
   1 AF MAR-STAT                              A 1.0   F
 *      M=MARRIED
 *      S=SINGLE
 *      D=DIVORCED
 *      W=WIDOWED
   1 AG SEX                                   A 1.0   F
   1 AH BIRTH                                 N 6.0
 G 1 A1 FULL-ADDRESS
 M 2 AI ADDRESS-LINE                          A 20.0  N
   2 AJ CITY                                  A 20.0  N
```

If you enter the HELP command or a question mark (?) in the **Command** line, the editor help information is displayed.

The header information of the DDM editor is:

DDM Name	The name used to reference the DDM in a Natural program. The name must be unique within the specified Natural system file.
Def. Seq.	The default sequence by which the file is read when it is accessed with a READ LOGICAL statement in a Natural program.
DBID	The database in which the file to be accessed with the DDM is contained. With Natural for VSAM, a DBID assigned to VSAM must be specified. If 0 is specified, the default DBID for the Natural user system file (FUSER) as defined in the Natural parameter module is used.

	Note: The actual DBID assignments for VSAM are made with `NTDB` macros when assembling the Natural parameter module; see *Installing Natural for VSAM* in the *Installation for z/OS* and *Installation for z/VSE* documentation.
FNR	The number of the file being referenced. The specified file number is used internally by Natural for VSAM.

The DDM itself comprises the following field definition attributes which can be entered or modified:

Attribute	Explanation
I	Line indicator. This field is used by the DDM editor to mark lines. `E` Lines containing an error detected during execution of the CHECK command. `S` Lines containing a scanned value. `X/Y` Lines selected for copy/move operation.
T	Field Type: `G` Group header. `M` Multiple-value field. `P` Periodic group header. `*` Comment line. *blank* Elementary field.
L	Level number assigned to the field. Valid level numbers are 1 - 7. Level numbers have to be specified in consecutive ascending order.
DB	For VSAM files, the two-character code which is used in VSAM.
Name	A 3- to 32-character external field name. This is the field name used in Natural programs to reference the field.
F	Field format. For valid formats, refer to User-Defined Variables, Format and Length of User-Defined Variables (in the Natural Programming Guide).
Leng	Standard field length. This length can be overridden in a Natural program. For numeric fields (format N), the length is specified as *nn.m*, where *nn* represents the number of digits before the decimal point and *m* represents the number of digits after the decimal point.
S	This attribute is not applicable to Natural for VSAM.
D	Descriptor Option. `A` Indicates that the field is an alternate index for a VSAM file. `P` Indicates that the field is a primary key. `S` Indicates that the field is a primary subdescriptor or superdescriptor; that is, a primary key for a VSAM file. `X` Indicates that the field is an alternate subdescriptor or superdescriptor; that is, an alternate index for a VSAM file.

Attribute	Explanation
Remark	A comment which applies to a field and/or the DDM.

Most of the editor and line commands available with the Natural program editor also apply to the Natural DDM editor. Special commands, such as `PROFILE`, `RENUMBER`, `SET`, `SHIFT`, etc. and some line commands are not available. Refer to the section *SYSDDM Utility* in the Natural *Editors* documentation and to the section *Program Editor* in the Natural *Editors* documentation for more details on editor commands.

Extended Editing at Field Level

The DDM editor can also be used to enter or modify DDM definitions at field level.

Extended editing mode is used to specify field headers and edit masks to be applied when the field is used in a `DISPLAY` or `INPUT` statement, as well as further specifications for VSAM DDM definitions. All the other information specific to the field (field type, length, name, format, remarks) can also be modified at this point.

The extended editing mode is invoked by entering the line command `.E` in the first positions of the line containing the field.

A range of field definitions can be selected for editing by entering `.E`*nnn* where *nnn* is the number of fields to be selected.

The field level editing mode is terminated when you press ENTER with or without having made any modifications.

The **Extended Field Editing** screen displays special attributes of the field definition if the edited DDM is a VSAM DDM:

```
11:25:26                    ***** EDIT DDM (VSAM) *****                    2007-02-25
                             - Extended Field Editing -
DDM Name AUTOMOBILES-VS                         Def.Seq.        DBID   254 FNR    12

I T L DB Name                                 F Leng  S D Remark
- - - -- ------------- top ------------- - ----- - - ----------------------
    1 GA OWNER-PERSONNEL-NUMBER               N 8.0     A SECONDARY KEY
-----------------------------------------------------------------------------


Field Header .......... OWNER/NUMBER_____
Field Edit Mask ....... _____

Alternate Index Name .. AUTOY___

Maximum Occurrence ....   1

Upgrade Flag .......... _ (X)
Unique Key Flag ....... _ (X)
Null Flag ............. _ (X)

Field GA redefines field __ with offset   0
```

The following attributes can be specified:

Attribute	Explanation
Alternate Index Name	If the field references a VSAM alternate index or a path (denoted by an A in column **D**), the index or path name must be entered here.
Maximum Occurrence	The number of occurrences for a multiple-value field or a periodic group (denoted by an M or P in column **T**).
The following flags only apply to alternate indexes and not to paths:	
Upgrade Flag	Since Natural does not use VSAM paths, upgrading can be performed either by Natural or by VSAM when using a KSDS or ESDS file with alternate indices defined. A blank value indicates that upgrading the alternate index is to be done by VSAM, which is the default. If VSAM is to perform the upgrading, define the VSAM file using IDCAMS with UPGRADE. If you enter an X, upgrading of the alternate index is performed by Natural. If so, the AIX must be defined with the NONUPGRADE option. **Note:** For LSR handling, it is recommended that you specify this option. Under CICS, the FCT entry must also contain the VARIABLE option.
Sort Flag	If this option is marked with an X, the alternate index is to be read in ascending or descending value order. This option only takes effect if the **Upgrade Flag** option is specified, too.

Attribute	Explanation
Unique Key Flag	If this option is marked with an X, Natural ensures that the values of the alternate index field are unique. An attempt to update with a non-unique value results in an error message. The default value is a blank. This option only takes effect if the **Upgrade Flag** option is also specified.
Null Flag	A value of S indicates that null values for the alternate index field are suppressed. The default value is a blank. This option only takes effect if the **Upgrade Flag** option is also specified.

> **Note:** For all DDMs cataloged with Natural which contain alternate indexes and any specifications for the above flags, all flags are nullified during runtime as soon as path processing is activated for these DDMs.

The last two fields on the screen are used to define sub-/superdescriptors for a VSAM file. For example, to define the field S1 as superdescriptor beginning in field BA and ending in field BB, the following would be entered:

```
S1 redefines BA with offset 0
```

The field S1 must have been defined to VSAM as a primary or secondary key.

VSAM superdescriptors can only be constructed from fields which are contiguous. To define the field S2 as a superdescriptor which begins in the 11th position of field BA and ends with the first two positions of field BB, the following would be entered:

```
S2 redefines BA with offset 10
```

In addition, the length of S2 would have to be set to 7. As mentioned above, S2 must have been defined as a primary or alternate index to VSAM.

Restrictions with DDM Generation as Compared to Adabas

- No keys can be defined within periodic groups.
- Descriptors that contain multiple-value fields are not allowed with VSAM.
- Natural DDMs for VSAM cannot contain multiple-value fields or periodic groups *within* periodic groups.
- The same field cannot be defined more than once in the same DDM. A data definition as in the following example would lead to unpredictable results when used with VSAM:

Example:

```
...
G 01 AB FULL-NAME
       02 AC FIRST-NAMEA 20.0  N
       02 AD MIDDLE-I                    A  1.0 N   /* duplicate short name
       02 AE NAME                        A 20.0
       01 AD MIDDLE-NAME                 A 20.0 N   /* duplicate short name
...
```

Natural would treat the field `MIDDLE-I` not as a redefinition of `MIDDLE-NAME` but as a separate field.

Buffers for Memory Management

The `VSIZE` parameter is suballocated into ten different areas whose sizes are determined by the assembly of the Natural parameter module. The different VSAM areas are split into fixed and variable buffer types. If there is insufficient space in the `VSIZE` buffer for all Natural parameter module areas, you receive error message NAT3592 during initialization. At runtime, error message NAT3513 can occur for fixed buffer types. In this case, you must adapt the corresponding the Natural parameter module value. Variable buffers are increased during runtime, NAT3513 does not occur. Some buffer sizes depend on the use of VSAM system files. The relevant buffers are FCT, FWA, TSA and UPD.

The `VSIZE` parameter is suballocated as follows:

- FCT - File Control Table
- FWA - File Work Area
- OPV - Open Table
- SFT - System File Table
- SWT - Switch Table
- TAF - Table of Accessed Files
- ROLL - Table of Session Status Information
- DFB - Table of Decoded Format Buffers
- TSA - Table of Sequential Access
- UPD - Table of Update Records

- VCA - Natural Control Area for VSAM

FCT - File Control Table

FCT contains file-specific information and is a fixed buffer type.

FCT also contains the complete VSAM access control block (ACB), information on existing user exits, and information on the application programming interface `USR0100N`.

The size of the table is determined by using the following formula and then rounding up to a double-word boundary:

(72 + *ACB-length*) (TAFE * 2) + 80

Without VSAM system files, the default setting is:

(72 + 76) (10 * 2) + 80 = 3040

 With VSAM system files, the default setting is:

(72 + 76) (26 * 2) + 80 = 7776

FCT and **SWT** (see below) share a common buffer area.

FWA - File Work Area

FWA contains information on a VSAM request and is a fixed buffer type.

FWA also contains information on the VSAM request parameter list (RPL).

The size of the table is determined by using the following formula and then rounding up to a double-word boundary.

(40 + *RPL-length*) (TAFE * 2) + 80

Without VSAM system files the default setting is:

(40 + 76) (10 * 2) + 80 = 2400

With VSAM system files the default setting is:

(40 + (76*4)) (26 * 2) + 80 = 17968

FWA and OPV (see below) share a common buffer area.

OPV - Open Table

OPV contains information on an `OPRB` string and is a fixed buffer type.

The size of the table is determined by using the following formula and then rounding up to a double-word boundary:

24 * (TAFE * 2) + 48

The default setting is :

24 * (10 * 2) + 48 = 528

OPV and **FWA** share a common buffer area.

SFT - System File Table

This table is only active if VSAM system files are defined. The buffer type is fixed.

This area contains the description of the VSAM system files `FNAT`, `FUSER`, `FDIC`, `FSEC` and `FSPOOL` as well as the system file used by Natural ISPF, if available.

The size of the area is 8192 for `SFILE=ON`. The default setting is 0.

SWT - Switch Table

SWT contains information necessary for the application programming interface `USR1047N` for dynamic DD/DLBL modification. SWT is allocated only if the value specified for the parameter `DDSWITE` in `NTVSAM` is greater than 0.

The SWT buffer type is fixed.

The size of the table is determined by using the following formula and then rounding up to a double-word boundary:

24 * DDSWITE + 48

The default setting is 0.

SWT and **FCT** (see above) share a common buffer area.

TAF - Table of Accessed Files

This area describes the record layout for each file accessed by Natural; it is created by reading the physical or logical DDM for the file. Each TAF entry consists of a header entry and an entry for each field in the DDM. The header entry describes the type of file, file name, primary key, etc. The field entries describe the format, offset, and length of every field in the file. The layouts for the header and field entries are described in the macros NVMTAF and NVMFLD respectively.

The TAF buffer type is fixed.

The size of the table is determined by using the following formula and then rounding up to a double-word boundary:

$(((32 * TAFN) + 112) * TAFE) + 80$

The default setting is:

$(((32 * 50) + 112) * 10) + 80 = 17200$

ROLL - Table of Session Status Information

This table is used to keep track of the position within a file for every active `FIND` or `READ` statement; it is identified by the CID. This allows Natural to release all VSAM resources during a ROLLOUT operation and then reposition itself correctly after a ROLLIN operation.

The ROLL buffer type is fixed.

The size of this area is determined by the subparameter `ROLLSIZ` of macro `NTVSAM` in the Natural parameter module, rounded up to a double-word boundary:

TAXSIZE + 80

The default setting is:

$550 + 80 = 632$

DFB - Table of Decoded Format Buffers

The table is suballocated into two areas, one for global format IDs (GFIDs) and one for command IDs (CIDs).

For any given I/O request, this area describes which fields from the VSAM record area are returned to the Natural record buffer. Each DFB (decoded format buffer) entry consists of one header, identified by the CID or the GFID of the I/O request, plus an entry for each field to be returned to Natural. Each field entry in the DFB contains the format, offset, and length of the field as derived from the associated TAF entry for the file. The layouts of the header and field entries are described in the macros `NVMDFB` and `NVMDFF` respectively.

The DFB buffer type is fixed. If the no-space-condition occurs for GFID-oriented entries, the oldest entries are deleted.

The size of the TDFB area is determined by using the following formula and then rounding up to a double-word boundary:

(16 * DFBN * 2 + 36) * DFBE * 2 + 128

The default setting is:

(16 * 50 * 2 + 36) * 10 * 2 + 128 = 32848

TSA - Table of Sequential Access

The TSA is used to keep important pointers and information on each READ or FIND statement. There is one TSA entry for each active READ and FIND statement, and each entry is identified by the CID. The layout of the TSA is described in the macro NVMTSA.

The TSA buffer type is variable.

The size of the area is determined by using the following formula and then rounding up to a double-word boundary:

(104 + KEYLGH) * TSAE + 80

Where TSAE = TSA entry.

The default setting is:

(104 + 32) * 10 + 80 = 1440

UPD - Table of Update Records

This area contains an entry for every READ or FIND loop that contains an UPDATE or DELETE statement. These entries are released when an END TRANSACTION or BACKOUT TRANSACTION statement is executed. Each entry contains control information about the record and the values of all the fields that might be updated within the loop. The layout of each UPD entry is described in the macro NVMUPD.

The UPD buffer type is variable.

The size of the UPD area is determined by the subparameter UPDL of macro NTVSAM in the Natural parameter module, rounded up to a double-word boundary.

The default setting is 8272 without VSAM system files and 32848 with VSAM system files.

VCA - Natural Control Area for VSAM

VCA is a fixed length area which contains pointers, addresses, flags, and work areas that are important to a Natural environment for VSAM. The layout for this area is described in the macro `NVMCA`. Within a Natural environment for VSAM, R3 always points to this area.

The size of this area is 6744 bytes.

Application Programming Interfaces

Natural for VSAM provides the following application programming interfaces (APIs) in the Natural system library `SYSEXT`:

API	Function
USR0100N	Controls the VSAM variable record length (LRECL).
USR1047N	Supports dynamic switching of DD/DLBL names defined in a DDM.
USR2008N	Supports dynamic `OPEN` calls for VSAM data sets.

A short description of the APIs is provided in the following section; for more detailed information, log on to the system library `SYSEXT` and display the text object (`USRxxxxT`) that corresponds to the required API.

The section below contains information on the following APIs:

- USR0100N
- USR1047N
- USR2008N

USR0100N

The API `USR0100N` controls the record length of variable VSAM files.

The API is invoked as follows (a sample program called `USR0100P` is provided in the library `SYSEXT`):

```
CALLNAT 'USR0100N' parm1 parm2 parm3 parm4 parm5
```

The parameters are described in the following table:

Parameter	Format/Length	Explanation
parm1	A1	Specifies either of the following function codes: G For retrieval statements; the current record length is determined for *parm5*. P For update/store statements; the length specified in *parm5* becomes the current record length.
parm2	A8	Specifies the DD/DLBL name for the current file (optional); if specified, *parm5* is only valid for this file.
parm3	N5	Specifies the DBID taken from the DDM (optional); is used instead of the DD/DLBL name and only in conjunction with *parm4*.
parm4	N5	Specifies the FNR taken from the DDM (optional).
parm5	N5	Specifies or returns the record length depending on the setting of *parm1*.

Note: If neither *parm2* nor *parm3* and *parm4* are specified, *parm5* is valid for all files.

USR1047N

The application programming interface `USR1047N` enables dynamic modification of DD/DLBL names within a Natural program if the `DDSWITE` subparameter is specified in the `NTVSAM` macro. It can be used if data are spread over several VSAM files which have different DD/DLBL names, but the same record structure.

The API is invoked as follows (a sample program called `USR1047P` is provided in the library `SYSEXT`):

```
CALLNAT 'USR1047N' parm1 parm2 parm3 parm4
```

The various parameters are described in the following table:

Parameter	Format/Length	Explanation
parm1	A1	Specifies either of the following function codes: S For switching of DD names with the next following database calls. R For resetting of DD names; the switch table entry of function S has been deleted (see *SWT - Switch Table*).
parm2	A8	Specifies the old DD name taken from the DDM.
parm3	A8	Specifies the new DD name for the next database calls.
parm4	P4	Return code of Natural for VSAM.

The parameter *parm4* can contain the following response codes:

Code	Explanation
0	Normal return.
4	The switch table (SWT) is too small; increase the `DDSWITE` subparameter in macro `NTVSAM`.
8	The switch table entry has not been found; program error.
12	Invalid function code.
16	The switch table is not allocated; that is, the `DDSWITE` parameter is set to `0`.

USR2008N

This application programming interface (API) is not applicable under Com-plete and CICS.

`USR2008N` supports dynamic `OPEN` calls during a Natural session if `OPSUPP=ON` is specified in the `NTVSAM` macro.

The API is invoked as follows (a sample program called `USR2008P` is provided in the library `SYSEXT`):

```
CALLNAT 'USR2008N' parm1 parm2 parm3 parm4 parm5 parm6
```

The parameters are described in the following table:

Parameter	Format/Length	Explanation
parm1	N5	Specifies the DBID taken from the **NTDB** macro definition; see ***NTDB Macro*** in the section *Natural for VSAM Parameters*.
parm2	A1	Specifies the global `OPEN MODE`; see ***OPEN/CLOSE Processing***.
parm3	A4	Specifies the data management type, for example, VSAM.
parm4	A40/16	Specifies the valid `OPRB` syntax and/or DDM long name instead of the `DD=` or `FNR=` definitions.
parm5	P4	Returns the Natural for VSAM error number.
parm6	A50	Returns the Natural for VSAM error text.

30 Natural Statements and Transaction Logic with VSAM

This section describes special considerations on Natural statements and Natural transaction logic when used with VSAM.

The Natural statements used to access VSAM files are a subset of those provided with the Natural language. No new statements are needed to access a VSAM file, since each Natural statement performs the same function regardless of the database management system or access method used. Therefore, programs written for VSAM files can also be used to access Adabas databases.

The Natural interface to VSAM has no built-in transaction logic and uses the one of the environment it is running in. This leads to different results depending on the environment.

Natural Statements with VSAM

This section mainly consists of information also contained in the Natural *Statements* documentation, where each Natural statement is described in detail, including notes on VSAM usage where applicable. Summarized below are the particular points a programmer has to bear in mind when using Natural statements with VSAM.

> **Note:** Since the Natural compiler does not check if a program adheres to the restrictions imposed by the Natural interface to VSAM, VSAM-specific programming errors concerning the use of Natural statements only occur when the program is executed.

Any Natural statement not mentioned in this section can be used with VSAM without restrictions.

- BACKOUT TRANSACTION
- DELETE
- END TRANSACTION
- FIND
- GET
- GET SAME
- GET TRANSACTION DATA
- HISTOGRAM
- READ
- STORE

- UPDATE

BACKOUT TRANSACTION

The `BACKOUT TRANSACTION` statement is used to back out all database updates performed during the current user logical transaction. This statement also releases all records held during the transaction.

If used with Natural for VSAM, the `BACKOUT TRANSACTION` statement releases records held in the UPD table. It does not back out transactions unless Natural is running under a TP monitor or DFSMStvs which supports dynamic transaction backout (for example, CICS). In this case, a `ROLLBACK` to the last `SYNCPOINT` is issued.

DELETE

The `DELETE` statement is used to delete a record from a VSAM file.

The use of the `DELETE` statement places each record selected in the corresponding `FIND` or `READ` statement in hold status.

The `DELETE` statement is not valid for VSAM entry-sequenced data sets (ESDS).

END TRANSACTION

The `END TRANSACTION` statement is used to indicate the end of a logical transaction. A logical transaction is the smallest logical unit of work (as defined by the user) which must be performed in its entirety to ensure that the information contained in the VSAM file is logically consistent.

The `END TRANSACTION` statement also releases all records placed in hold status during the transaction.

An `END TRANSACTION` only releases records held in the UPD table unless Natural is running under a TP monitor or DFSMStvs which supports dynamic transaction backout (for example, CICS). In this case, an `END TRANSACTION` statement causes a `SYNCPOINT` to be issued.

FIND

The `FIND` statement is used to select a set of records from the VSAM file based on a search criterion consisting of fields defined as descriptors (keys).

The `WITH` clause is used to specify the search criterion consisting of key fields (descriptors) defined in the VSAM file.

Only VSAM key fields can be used.

The number of records to be selected as a result of a WITH clause can be limited by specifying the keyword LIMIT together with a limit value (*operand 1*) expressed as a numeric constant or a user-

defined variable. The limit value is enclosed within parentheses. If the number of records selected exceeds the limit value, the program is terminated with an error message.

The descriptor must be defined in a VSAM file as a VSAM key field. In a DDM, it is marked with `P` for primary key, `S` for primary sub/superdescriptor, `X` for alternate sub/superdescriptor or `A` for alternate key (see *Edit DDM* in the section *Operation*, and the *SYSDDM Utility* as described in the Natural *Editors* documentation).

The formats of the descriptor and the search value must be compatible.

The following Natural system variables are available with the `FIND` statement:

Variable	Content
`*ISN`	The system variable `*ISN` contains the relative byte address of the record currently being processed (ESDS files only). This variable is not available for the `FIND NUMBER` and `FIND FIRST` statements.
`*NUMBER`	The system variable `*NUMBER` contains the number of records which satisfied the basic search criterion specified in the `WITH` clause, and before evaluation of any `WHERE` criterion. `*NUMBER` only contains a meaningful value if the `EQUAL TO` operator is used in the search criterion. With any other operator, `*NUMBER` will be `0` if no records have been found; any other value indicates that records have been found, but the value will have no relation to the number of records actually found. The same applies to `*NUMBER` with the `FIND NUMBER` statement.
`*COUNTER`	The system variable `*COUNTER` contains the number of times the processing loop has been entered. This system variable is not available for the `FIND NUMBER` statement.

The `FIND` statement is only valid for key-sequenced (KSDS) and entry-sequenced (ESDS) VSAM data sets. For ESDS, an alternate index or a path for an alternate index must be defined. Relative record data sets (RRDS) are not allowed, since they do not contain any key fields (descriptors).

GET

The `GET` statement is used to read a record with a given VSAM record number. For an ESDS file, the record number (ISN) would be the relative byte address (RBA); for RRDS and VRDS files, it would be the relative record number (RRN). The `GET` statement does not initiate a processing loop. As a result, a subsequent `UPDATE` or `DELETE` statement will not be processed and Natural returns a corresponding error message.

For ESDS, the RBA must be contained in a user-defined variable (numeric format) or specified as an integer constant. The same rules apply for RRDS and VRDS with the exception that the RRN must be provided instead of the RBA.

GET SAME

The `GET SAME` statement applies to VSAM ESDS, RRDS, and VRDS only (see also the `GET` statement above).

GET TRANSACTION DATA

The `GET TRANSACTION DATA` statement is not applicable to the Natural interface to VSAM.

HISTOGRAM

The `HISTOGRAM` statement is used to read the values of a field which is defined as a descriptor, subdescriptor, or superdescriptor.

The `HISTOGRAM` statement initiates a processing loop, but does not provide access to any fields other than the field specified in the statement.

Only VSAM key fields can be used as descriptors.

The following Natural system variable is available with the `HISTOGRAM` statement:

Variable	Content
`*NUMBER`	When used in conjunction with a KSDS primary key or a unique alternate index, `*NUMBER` is always 1.

> **Note:** The `*ISN` system variable is not available for the Natural interface to VSAM.

When used with VSAM, the `HISTOGRAM` statement is only valid for KSDS and ESDS data sets. For ESDS, an alternate index or a path for an alternate index must be defined.

The values are read directly from the VSAM index and are returned in ascending or descending value sequence.

READ

The `READ` statement is used to read records from a VSAM file. The records can be retrieved in the value sequence (ascending or descending) of a descriptor (key) field. The `READ` sequence initiates a processing loop.

`IN LOGICAL SEQUENCE` is used to read records in the order of the values of a descriptor (key). If `LOGICAL` is specified with a descriptor, the records are read in the value sequence of the descriptor. A descriptor can be used for sequence control. A descriptor within a periodic group cannot be used. If `LOGICAL` is specified without a descriptor, the records are read in the default descriptor sequence, as defined in the DDM.

WITH REPOSITION can be used for skip-sequential processing inside the active loop, the new position must be defined as the new start value for the loop and must reset the system variable *COUNTER.

IN LOGICAL SEQUENCE is only valid for KSDS with primary and alternate keys defined and ESDS with alternate keys defined. A subdescriptor or superdescriptor can be used for sequence control, too.

The following Natural system variables are available with the READ statement:

Variable	Content
*ISN	The system variable *ISN contains either the RRN (for RRDS or VRDS) or the RBA (for ESDS) of the current record.
*COUNTER	This system variable contains the number of times the processing loop has been entered.

Records can also be retrieved IN PHYSICAL SEQUENCE, which is used to read records in the order in which they are physically stored in a database. It is only valid for VSAM ESDS, RRDS and VRDS. This is the default sequence.

STARTING WITH ISN can be used as start value for the loop in ascending or descending physical sequence.

BY ISN is used to read records in RBA and RRN order for ESDS, RRDS and VRDS files, respectively.

STORE

The STORE statement is used to add a record to a database.

A unique value for the primary-key field or the alternate-index field must be provided if the data set is defined with a primary key or a unique alternate index.

The USING/GIVING NUMBER clause is only valid for RRDS or VRDS, in which case the ISN corresponds to the relative record number.

USING/GIVING NUMBER is used to store a record with a user-supplied RRN. If a record with the specified RRN already exists, an error message is returned and the execution of the program is terminated, unless ON ERROR processing was specified.

The Natural system variable *ISN contains the RRN assigned to the new record as a result of the STORE

statement execution. A subsequent reference to *ISN must include the statement number of the related STORE statement. *ISN is available for RRDS or VRDS files only.

UPDATE

The `UPDATE` statement is used to update one or more fields of a record in a database. The record to be updated must have been previously selected using a `FIND` or `READ` statement.

The primary key cannot be updated.

Natural Transaction Logic with VSAM

Natural for VSAM uses the transaction logic of the environment it is running in. Thus, the results of the Natural `END TRANSACTION` and `BACKOUT TRANSACTION` statements (see also the relevant sections in Natural Statements with VSAM) differ depending on the actual environment:

- With Native VSAM
- Under CICS
- Under DFSMStvs

With Native VSAM

Since VSAM itself has no transaction logic, there is no transaction logic available if Natural is working in a native VSAM environment. This is the case under Com-plete, TSO, and in batch mode, which means when `NVSMISC` is the I/O module in use.

With `NVSMISC`, `END TRANSACTION` and `BACKOUT TRANSACTION` statements do not return any error messages, but are ignored by the Natural interface to VSAM.

Under CICS

Under CICS, VSAM files can be defined as "recoverable resources" or for RLS as "recoverable sphere", all of which are synchronized by CICS using the concept of "logical units of work" (LUWs). An LUW ends if a `SYNCPOINT` command is issued or if the CICS task is terminated. For details, refer to the relevant IBM literature on CICS.

Below is information on:

- NVSCICS Module
- Conversational Tasks

> ■ Pseudo-Conversational Tasks

NVSCICS Module

For CICS, the I/O module `NVSCICS` is a normal command-level application program. It transfers `END TRANSACTION` and `BACKOUT TRANSACTION` statements to the `NATCICS` driver which issues the `EXEC CICS SYNCPOINT` and `EXEC CICS ROLLBACK` commands. If an error occurs in a Natural session with uncommitted updates and no error transaction is supplied, Natural itself triggers the interface to VSAM to issue a `ROLLBACK` command.

If a `SYNCPOINT` or `ROLLBACK` command fails (for example, when CICS answers with a `ROLLEDBACK` condition to a `SYNCPOINT` request), error messages NAT3544 or NAT3545 are returned.

Conversational Tasks

If the Natural session runs in CICS conversational mode, the LUW is not ended by a terminal I/O. Natural runs in conversational mode if either the Natural parameter `PSEUDO=OFF` has been specified or Natural itself has determined that pseudo-conversational processing is not possible.

Since terminal I/Os do not disturb the transaction logic of an application as long as Natural is running in conversational mode, a program like the following one would work without problems:

Example:

```
READ vsam-file
 UPDATE
 INPUT ...
END-READ
BACKOUT TRANSACTION
```

Pseudo-Conversational Tasks

If the Natural session is running in pseudo-conversational mode, each terminal I/O terminates the CICS task, thus implicitly performing a `SYNCPOINT`. Therefore, the impact of a `BACKOUT TRANSACTION` statement, that is of an `EXEC CICS SYNCPOINT ROLLBACK` command, only goes back to the most recent terminal I/O. The example program above would, therefore, end with error message NAT3548, because it is not possible to roll back all the updates.

> **Note:** Keep in mind that all messages of the Natural interface to VSAM are issued at runtime only, since the Natural compiler is not able to detect this kind of logical error.

Under DFSMStvs

DFSMS Transactional VSAM Services (DFSMStvs) provides the same features CICS provides: forward and backward recovery logging, backout processing and a two-phase commit process. An LUW ends if the RRS (Resource Recovery Service) call `SRRCMIT` or `SRRBACK` is issued (`END TRANSACTION` or `BACKOUT TRANSACTION`). For details, refer to the relevant IBM literature on DFSMStvs and RRS.

IV Natural for DL/I

This documentation provides information on Natural in a DL/I environment. It describes the operation of Natural for DL/I, as well as special considerations on Natural statements when used with DL/I.

This documentation covers:

General Information	Brief information on features.
Accessing DL/I Data	How to enable access to DL/I databases using Natural statements.
Natural Parameter Modifications for DL/I	Parameters contained in `NDLPARM`, storage estimates, and Natural for DL/I in z/OS environments.
Operation	Describes the procedures `NATPSB`, `NATDBD`, `NATUDF`, and the generation of DDMs from DL/I segment types.
System File Structure	Describes the database structure, the segment data and the processing intent of an application.
Natural Batch Utilities	Describes the system file transfer of NDBs, NSBs and UDFs from one `FDIC` and the use of the batch utility `NDUDFGEN` to generate Natural data areas.
Execution	Describes PSB scheduling, the `CALLNAT` interface, support of IMS TM-specific features, fast path and GSAM, and CICS mode processing under IMS TM.
Programming Language Considerations	Natural versus third generation languages, Natural statements with DL/I, Natural system variables with DL/I.
Problem Determination Guide	Actions required to correct a given problem.
Performance Considerations	How to increase the performance of Natural in a DL/I environment.
DL/I Services	Terminology and maintenance of NDBs and NSBs.

Related Documentation

For installation instructions, see *Installing Natural for DL/I* in the *Installation for z/OS* documentation.

For various aspects of accessing data in a database with Natural, refer to *Database Access* in the *Programming Guide*.

For a list of DL/I status codes and abend codes (under CICS only), refer to *Status Codes and Abend Codes* in the Natural *Messages and Codes* documentation.

31 General Information

With Natural for DL/I, a Natural user can access and update data stored in a DL/I database. The Natural user can be executing in batch mode or under the control of the TP monitor CICS or IMS TM.

A DL/I database is represented to Natural as a set of files, each file representing one database segment type. Each file or segment type must have an associated DDM generated and stored on the Natural system file `FDIC`.

Since Natural for DL/I is an extension to Natural, nearly all of the information contained in the Natural documentation applies to its use in the DL/I environment as well as in the Adabas environment.

The Natural statements used to access DL/I databases are a subset of those provided with the Natural language. No new statements are needed to access a DL/I database.

Applications developed using Natural for DL/I operate as standard DL/I applications. This means that all access to DL/I databases performed by Natural follows the DL/I product conventions. For an online Natural session or batch Natural program to issue a DL/I database call, a PSB must first be scheduled. The PCB in use must have segment sensitivity and the appropriate `PROCOPT` parameter must be specified for Natural, to be able to perform a segment update. Only standard DL/I database calls are issued by Natural.

32 Accessing DL/I Data

Natural for DL/I allows Natural programs to access DL/I databases using Natural statements.

To access DL/I data, Natural requires certain information on these data. This information mainly consists of four types of control blocks:

- the original database descriptions (DBDs) and program specification blocks (PSBs) which are required by DL/I itself;

- suitable copies of DL/I DBDs and PSBs for Natural, called NDBs and NSBs;

- user-defined fields (UDFs);

- Natural DDMs generated from NDBs and UDFs.

All information required by Natural to access DL/I databases is stored and maintained in the Natural system file `FDIC`. The Natural system file `FDIC` can be an Adabas file (if Adabas is installed), or a VSAM file (only in CICS environments).

As is the case with any DL/I application, a DL/I DBDGEN and PSBGEN must be performed to define the data structure the Natural application is to have access to, and the processing intent this application has on these data. This same information, which is contained in the DBD and PSB source statements, must also be defined to Natural.

The Natural batch procedures `NATDBD` and `NATPSB` are used to add this information to the Natural `FDIC` system file. They generate NDBs and NSBs from the respective DBDs and PSBs, using the DBDGEN and PSBGEN source respectively, as input.

It is the administrator's responsibility to ensure that the contents of the DL/I DBDLIB and PSBLIB and the Natural system file `FDIC` are compatible. It is therefore recommended that the DL/I procedures DBDGEN and PSBGEN and the Natural procedures `NATDBD` and `NATPSB` always be executed as a pair.

The DBDGEN source usually does not define all fields within a segment. Additional segment fields, called user-defined fields (UDFs), can be entered as part of creating the DDMs. UDFs in

Natural are added by using either the batch utility `NATUDF`, the *Edit an NDB Segment Description* facility of the `SYSDDM` utility, or Predict.

Once all the necessary information has been stored on the Natural system file `FDIC`, Natural DDMs defining the DL/I database segment types can be created.

33 Natural Parameter Modifications for DL/I

Natural parameter default values for DL/I can be changed to meet your particular requirements. The object module `NDLPARM`, which is used for Natural static parameter assignment in a DL/I environment, must then be appropriately modified and reassembled.

This section covers the following topics:

Parameters in NDLPARM

The following parameters are contained in `NDLPARM`:

- DFBNUM - Maximum Entries in Translated Format Buffer
- DFFNUM - Maximum Fields in Single Entry of Translated Format Buffer
- FLBNUM - Number of Entries in Fast Locate Buffer
- INGSIZE - Initial Size of Buffer to Copy Parameter List
- INGOSIZ - Initial Size of I/O Area for DL/I Calls
- INITCAL - Issues INIT Call at Transaction Start
- PCBLEV - Maximum Number of PCB Levels
- PCBNUM - Maximum Number of PCBs in a PSB
- RELEVNT - Requests Relocation Event
- RESINDB - NDB Resident in Buffer Pool
- RESINSB - NSB Resident in Buffer Pool
- RESIUDF - UDF Resident in Buffer Pool
- SASIZE - Size of Natural Save Area for DL/I
- SEQNUM - Maximum Number of Nested Sequential Accesses
- SEQSSA - Maximum Size of an SSA
- THCSIZE - Table Size to Save Natural Field Values
- TRACE - Trace Options
- TYPCHCK - Numeric/Packed Data Check
- TYPWARN - Issues Data Check Warning
- VALNSB - Validate NSB (against PSB)
- WORKLGH - Size of Work Areas

DFBNUM - Maximum Entries in Translated Format Buffer

Possible Values	Default Value
5 - 200	25

This parameter is used to indicate the maximum number of entries in the table of translated format buffers.

An entry in this table is created for each active Natural input/output statement (`FIND`, `READ`, `UPDATE`, `STORE`).

When increasing `DFBNUM` or `DFFNUM`, take into consideration that the allocated storage area size is obtained by multiplying these values and *not* by adding them.

DFFNUM - Maximum Fields in Single Entry of Translated Format Buffer

Possible Values	Default Value
5 - 1000	10

This parameter is used to indicate the average number of fields contained in each single entry of the table of translated format buffers.

A field entry in this table is created for each field referenced in a Natural input/output statement (`FIND`, `READ`, `UPDATE`, `STORE`).

When increasing `DFFNUM` or `DFBNUM`, take into consideration that the allocated storage area size is obtained by multiplying these values and *not* by adding them.

FLBNUM - Number of Entries in Fast Locate Buffer

Possible Values	Default Value
0 - 32767	50

This parameter is used to indicate the number of entries in the Fast Locate Buffer. This buffer holds absolute addresses of Natural for DL/I objects (that is, NDBs, NSBs, UDFs) in the buffer pool.

The addresses are stored in wrap-around technique.

This buffer is especially useful if Natural for DL/I objects have been marked as "resident" in the buffer pool (see the related parameters `RESINDB`, `RESINSB`, `RESIUDF`).

It allows Natural for DL/I to use the Fast Locate algorithm of the Natural buffer pool manager when locating objects.

INGSIZE - Initial Size of Buffer to Copy Parameter List

Possible Values	Default Value
1000 - 32767 (bytes)	1000

This parameter is used to indicate the initial size of the buffer which is used to copy the DL/I call parameter list and the call parameters below 16 MB if Natural operates in a z/OS environment. If the initial size is not sufficient, Natural automatically increases the size of this buffer accordingly.

INGOSIZ - Initial Size of I/O Area for DL/I Calls

Possible Values	Default Value
1000 - 32767 (bytes)	1000

This parameter is used to indicate the initial size of the I/O area for DL/I calls. This area is re-used for subsequent DL/I calls if no `GET HOLD` call has been issued.

If the initial size is not sufficient, Natural automatically increases the size of this buffer accordingly.

INITCAL - Issues INIT Call at Transaction Start

Possible Values	Default Value
NO/YES	NO

This parameter is used to inform IMS TM that Natural is prepared to accept status codes `BA` or `BB` regarding data unavailability.

The setting of this parameter only applies if Natural runs in a BMP or MPP region.

PCBLEV - Maximum Number of PCB Levels

Possible Values	Default Value
1 - 15	10

This parameter is used to indicate the maximum number of PCB levels which can be processed by Natural.

When increasing `PCBLEV`, take into consideration that the allocated storage area size is obtained by multiplying these values and *not* by adding them.

PCBNUM - Maximum Number of PCBs in a PSB

Possible Values	Default Value
1 - 255	25

This parameter is used to indicate the maximum number of PCBs which can be contained within a single PSB.

When increasing `PCBNUM`, take into consideration that the allocated storage area size is obtained by multiplying these values and *not* by adding them.

RELEVNT - Requests Relocation Event

Possible Values	Default Value
NO/YES	NO

This parameter is used to inform the Natural nucleus whether or not Natural for DL/I requests relocation events.

With `RELEVNT=YES`, Natural for DL/I is called for relocation on every relocation event, that is, even if no DL/I call has been issued since the last relocation event.

With `RELEVNT=NO`, Natural for DL/I is not called for relocation. Instead, it checks itself whether relocation is required before a DL/I call is issued.

RESINDB - NDB Resident in Buffer Pool

Possible Values	Default Value
NO/YES	YES

This parameter is used to indicate whether NDBs are to be kept resident in the buffer pool.

RESINSB - NSB Resident in Buffer Pool

Possible Values	Default Value
NO/YES	YES

This parameter is used to indicate whether NSBs are to be kept resident in the buffer pool.

RESIUDF - UDF Resident in Buffer Pool

Possible Values	Default Value
NO/YES	YES

This parameter is used to indicate whether UDFs are to be kept resident in the buffer pool.

SASIZE - Size of Natural Save Area for DL/I

Possible Values	Default Value
1000 - 3000 (bytes)	1000

This parameter is used to indicate the size of the save area.

Do not increase the default value, unless you receive an error message which indicates that a save area overflow has occurred.

SEQNUM - Maximum Number of Nested Sequential Accesses

Possible Values	Default Value
5 - 100	20

This parameter is used to indicate the maximum number of nested sequential accesses which can be processed by Natural.

When increasing the values for the `SEQNUM` and `SEQSSA` parameters, remember that the storage area allocated is dependent on the product of these areas, *not* their sum.

SEQSSA - Maximum Size of an SSA

Possible Values	Default Value
10 - 500 (bytes)	50

This parameter is used to indicate the maximum size of an SSA related to sequential access.

When increasing the values for the `SEQNUM` and `SEQSSA` parameters, remember that the storage area allocated is dependent on the product of these areas, *not* their sum.

THCSIZE - Table Size to Save Natural Field Values

Possible Values	Default Value
2000 - 32000 (bytes)	3000

This parameter only applies under IMS TM or under CICS in pseudo-conversational mode.

This parameter is used to indicate the size of the table which is used to save field values in hold status when running under IMS TM or under CICS in pseudo-conversational mode.

TRACE - Trace Options

Possible Values	Explanation
ALL	Trace all modules
CMD	Trace command execution
REQ	Trace request modules
ROU	Trace routines
SER	Trace service modules
OFF	Trace is not active. Default value.

This parameter is used to indicate whether Natural trace information is to be created and printed or not.

The options CMD, REQ, SER and ROU can be combined.

TYPCHCK - Numeric/Packed Data Check

Possible Values	Default Value
NO/YES	NO

This parameter is used to indicate whether numeric or packed segment fields from DL/I are to be checked for valid data and repaired, if necessary.

With TYPCHCK=NO, no data check is performed. Natural for DL/I would abend with data exception if, for example, a packed field contained blanks.

With TYPCHCK=YES, a data check is performed. If the field does not contain format compatible data, it is filled with zeroes. In addition, a message is issued, depending on the setting of the parameter TYPWARN (see below).

TYPWARN - Issues Data Check Warning

Possible Values	Default Value
NO/YES	NO

This parameter only applies if TYPCHCK has been specified (see above).

This parameter is used to indicate whether a message is to be issued if a data check and repair has been performed.

With TYPWARN=NO, no message is issued if a data repair has been performed.

With TYPWARN=YES, a message is issued if a data repair has been performed. This message displays the short name of the field in error. The message is issued as a warning (only), which means that:

- The message is not issued via the Natural error exit but is directly inserted into the page buffer.

- The message(s) is (are) only issued when the page buffer is full.

- There is no backout transaction.

- The program flow is not interrupted.

VALNSB - Validate NSB (against PSB)

Possible Values	Default Value
NO/YES	YES

This parameter is used to indicate whether the NSB is to be validated (against the PSB).

With `VALNSB=YES`, the NSB is validated. The NSB name in the `NATPSB` command is checked against the PSB name in the `EXEC` statement of the batch environment.

If the names are not identical, a message is issued.

With `VALNSB=NO`, NSB validation is bypassed. This allows for multiple business units sharing the same application. Improper use may cause mismatch problems or abends.

WORKLGH - Size of Work Areas

Possible Values	Default Value
1000 - 3000	1000

This parameter is used to indicate the size of the work areas. Natural allocates six work areas of this size.

Do not increase the default value, unless you receive an error message which indicates that a work area overflow has occurred.

Storage Estimates

The memory size required by Natural for DL/I is determined by the following items:

1. Object code: 90 KB.

2. Save areas: 3 KB.

3. Work areas: 6 KB.

4. Fast Locate Buffer: 12 bytes for each entry.

5. XRST buffer: 2 KB.

6. Internal tables: the amount of storage allocated depends on parameters specified in the module `NDLPARM`. The following formula can be used to compute the amount of storage required for initial table allocation:

Amount of Storage =

```
SEQNUM * (SEQSSA + 64) + 32 +
DFBNUM * (28 + (DFFNUM * 12)) + 20 +
PCBNUM * (24 + 12 + (PCBLEVL * 5)) + 20 +
TCHSIZE
```

The above formula can be described as follows:

Term	Computational Expression
Sequential Access Table	SEQNUM * (SEQSSA + 64) + 32
Field Table	DFBNUM * (28 + (DFFNUM * 12)) + 20
PCB Map	PCBNUM * (24 + 12 + (PCBLEVL * 5)) + 20
Table of Fields in Hold	TCHSIZE

If the standard values of these `NDLPARM` parameters are used in the above formula, 14 KB of storage is allocated.

7. Segment I/O areas are to be added on additionally.

> **Note:** The object code is shared among all Natural sessions. There is a copy of all other areas for each active Natural session.

The storage required for save areas, work areas, Fast Locate Buffer, XRST buffer and internal tables is allocated from the thread at the initialization of the Natural session. Six GETMAINs are performed, the sizes of which are determined by the values of the parameters in the `NDLPARM` module. If the default values of the `NDLPARM` parameters are used, the total size required is 27 KB.

The total size available is determined by the profile parameter `DLISIZE` in the Natural parameter module; see the Natural *Parameter Reference* documentation.

The `BUS` (Buffer Usage Statistics) system command can be used to obtain information on the sizes of the buffers allocated by Natural for DL/I. The following information is provided:

Buffer	Content
`DLISIZE0`	Contains the Fast Locate Buffer, the XRST buffer and the save areas.
`DLISIZE1`	Contains the work areas.
`DLISIZE2`	Contains the sequential access table.
`DLISIZE3`	Contains the field table.
`DLISIZE4`	Contains the PCB map.

Buffer	Content
`DLISIZE5`	Contains the table of fields in hold status.

Natural for DL/I in z/OS Environments

Before Natural issues a DL/I call in a z/OS environment, it checks whether the call parameter list or any of the call parameters reside above the 16 MB line. This is the case if the Natural threads have been placed above this line. If so, the parameter list and all parameters are copied into a buffer which has been allocated below the line via GETMAIN. The pointers in the parameter list are modified accordingly to point to the new parameters.

The initial size of this buffer is set by the `INGSIZE` parameter of `NDLPARM`. If the initial size is not sufficient, Natural automatically increases the size of this buffer accordingly.

This overhead is required because DL/I terminates programs abnormally if parameter addresses passed in DL/I calls do not refer to code or storage areas below the 16 MB line.

34   Operation

Natural for DL/I operates as a standard DL/I application.

Prior to running a Natural application, a PSB must be scheduled. The method for scheduling PSBs varies depending on the actual environment (see the relevant sections under *PSB Scheduling*), but as for any other DL/I application, PSB scheduling is a requirement.

This section covers the following topics:

Procedure NATPSB

Every PSB required by DL/I to accommodate Natural requests must be processed by the Natural batch utility `NDPBNSB0`. This utility stores DL/I PSB information, in a form suitable for Natural, on the `FDIC` system file. This information is referred to as NSB control block. A batch procedure called `NATPSB` has been established for this purpose.

A sample `NATPSB` job has been included in the source library from the installation medium. The information used to create NSB control blocks comes from the actual PSBGEN source. It is essential that the same input is used for the `NATPSB` procedure as was used for the DL/I PSBGEN. Otherwise, unpredictable results are likely.

The `NATPSB` job is a three step procedure:

- The first step executes the normal DL/I PSBGEN procedure. This step is included to guarantee compatibility between DL/I and Natural.
- The second step performs another assembly and link of the PSBGEN source, this time using macros supplied by Natural.
- The final step executes the Natural batch utility `NDPBNSB0`, which uses the linked PSB module from the previous step to create NSB control blocks which are stored on the `FDIC` system file. `NDPBNSB0` dynamically loads the Natural module `NDLB0002`, which therefore must be present in an allocated load library.

Natural requires one or more PSBs for batch and/or online processing. Depending on application requirements, the PSB can be switched during a Natural session. Each PSB describes all user views that can be used to access DL/I databases from Natural programs if this PSB is active. A PSB must contain one or more program communication blocks (PCBs) for each DBD to be accessed. Since Natural only uses the single positioning option on PCBs, Natural programs that maintain two or more independent positions in a database require a PCB (of the appropriate type) for each separate position.

If this requirement is not fulfilled, Natural for DL/I issues the runtime error message:

```
NAT3789 Active PSB contains too few PCBs for program execution
```

The PCB in use must have segment sensitivity and the appropriate `PROCOPT` parameter specified for Natural, to be able to perform a segment update.

Nested I/O loops (`FIND` or `READ`) in Natural programs frequently require separate positions in the same database to be maintained. To reduce the number of PCBs needed, as many I/O loops as possible should be closed before opening subsequent I/O loops.

Consider the following sample DL/I database:

Sample Education Database ED00DBD:
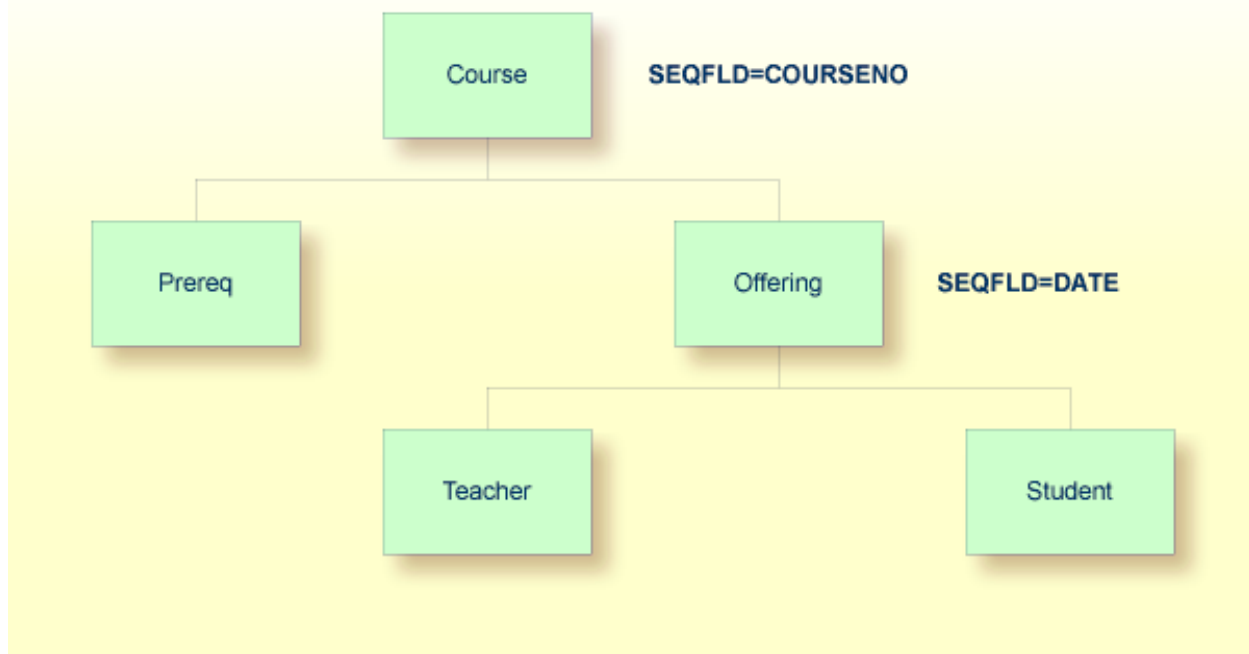
The following Natural program based on the above database requires two PCBs:

```
READ ED00DBD-COURSE BY COURSENO
   FIND ED00DBD-PREREQ WITH COURSENO-COURSE = COURSENO
     FIND ED00DBD-OFFERING WITH COURSENO-COURSE = COURSENO
     LOOP
   LOOP
LOOP
END
```

The first PCB is used to maintain position on the `COURSE` and `PREREQ` segments. A second PCB is required for the `OFFERING` segment since the `FIND` loop has not been terminated for the `PREREQ` segment prior to invoking a `FIND` on the `OFFERING` segment. By closing the first `FIND` loop prior to opening the second one, this program would only require one PCB.

Natural selects the PCB to be used for a database request in the following manner:

1. Natural selects the first PCB in the PSB with the correct DBD name and the appropriate `PROCSEQ` parameter (if applicable).

2. Natural then determines if the PCB can be used for the request or if there is a conflict due to current database positioning.

3. If there was a positioning conflict or the PCB did not contain the correct DBD name or `PROCSEQ` parameter, Natural would continue scanning the PSB.

4. If the database search request refers to a secondary index, Natural attempts to use a PCB with the corresponding PROCSEQ parameter. If there is no PCB of this type in the PSB, Natural tries to use a PCB without the PROCSEQ parameter. In this case, it is assumed that the INDICES parameter has been coded in the appropriate SENSEQ statement.

5. If no eligible PCB could be found, an error message would be generated.

In general, PCBs for use by Natural can have different PROCOPT parameters. However, if there are two or more PCBs in the PSB referring to the same DBD, these PCBs must appear consecutively in the PSB source and they must specify the same SENSEG statements and same PROCOPT parameters. They can, however, have different PROCSEQ parameters.

When locating an eligible PCB, Natural disregards the PROCOPT parameter of the PCB. The first free PCB is selected independently of the PROCOPT parameter, so that if the chosen PCB has a PROCOPT that does not support the request, an error message that corresponds to a DL/I status code is returned.

Natural assumes that all PCBs with the same DBD name and the same PROCSEQ parameter contain the same SENSEG statements as the first PCB. If this is not true and a PCB is selected that does not contain a SENSEG statement for the segment being referenced, an error message that corresponds to a DL/I status code is returned.

The following example PSB and Natural program demonstrate that the sequence of the PCBs, referring to the same DBD, may affect Natural programs if the PROCOPT parameters are different:

```
PCB     TYPE=DB,DBDNAME=ED00DBD,PROCOPT=GO,...
SENSEG NAME=COURSE
SENSEG NAME=OFFERING,PARENT=COURSE
PCB     TYPE=DB,DBDNAME=ED00DBD,PROCOPT=A,...
SENSEG NAME=COURSE
```

The following program requires two PCBs: the first PCB is used for the READ loop (which reads all COURSE segments) and the second nested FIND loop (which finds one offering to a given course); the second PCB is used for the first FIND loop (which updates a specific COURSE segment). The program does not work if the order of the two PCBs is reversed.

```
READ COURSE BY COURSENO
   FIND (1) COURSE WITH COURSENO = '120'
      UPDATE WITH TITLE = 'Natural'
   LOOP
   FIND (1) OFFERING WITH COURSENO-COURSE = COURSENO (0010)
      DISPLAY COURSENO-COURSE
   LOOP
LOOP
END
```

The following figure shows the logical connections between DL/I PSBs, PCBs, sensitive segment types and Natural DDMs:
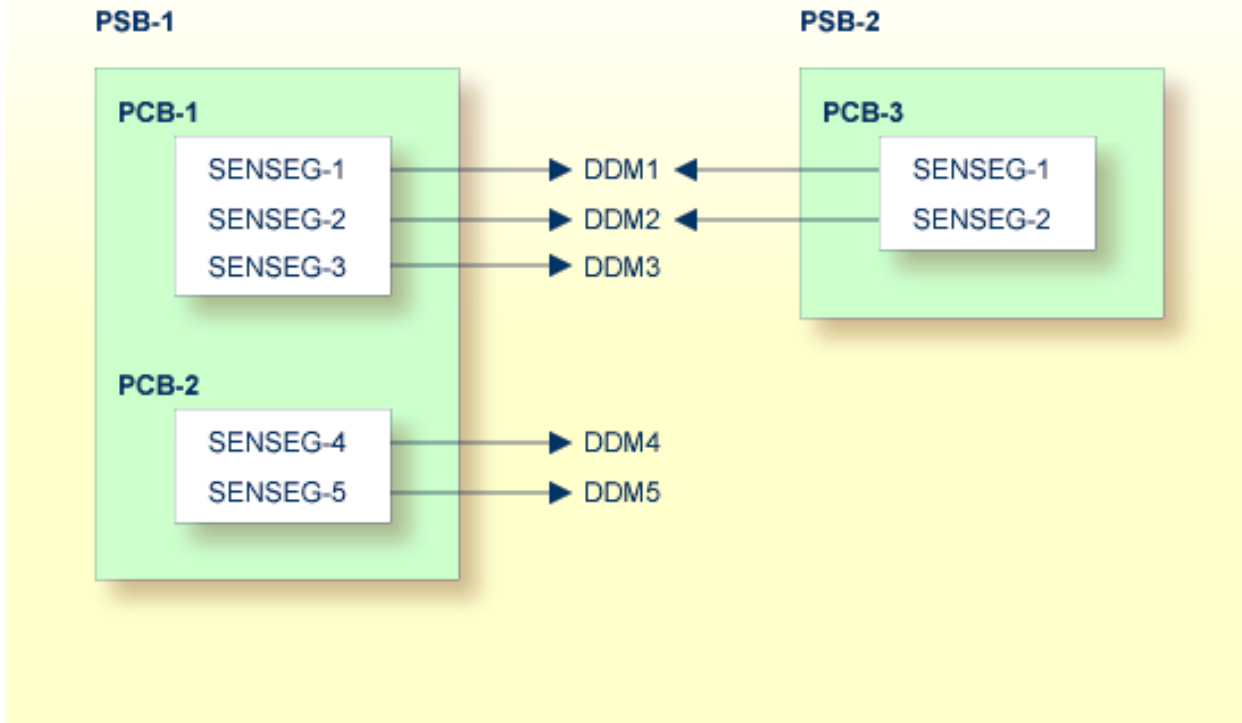
Natural DDMs which are derived from segment descriptions in the DBD correspond to DL/I segment types.

Since each DL/I application program requires the specification of its sensitive segment types, an appropriate PSB must be scheduled before Natural program execution. A PSB can be scheduled at the start of a Natural session or at any time during the session.

If, in the configuration shown in the diagram above, PSB-2 has been scheduled, only the DDMs DDM1 and DDM2 are accessible to Natural application programs. If an attempt is made to use DDM5, for example, Natural for DL/I returns the error message:

```
NAT3768 PCB with requested DBD not found in NSB
```

Procedure NATDBD

Every DL/I database structure, both physical and logical, which is supposed to be used by Natural, must be processed by the Natural batch utility `NDPBNDB0`.

This utility stores DL/I database information on the `FDIC` system file, in a form suitable for Natural. This information is referred to as *NDB control block*. A batch procedure called `NATDBD` has been established for this purpose.

A sample `NATDBD` job has been included in the source library on the installation medium. The information used to create NDB control blocks comes from the actual DBDGEN source. It is essential that the same input is used for the `NATDBD` procedure as was used for the DL/I DBDGEN. Otherwise, unpredictable results are likely.

The `NATDBD` job is a three step procedure:

- The first step executes the normal DL/I DBDGEN procedure. This step is included to guarantee compatibility between DL/I and Natural.

- The second step performs another assembly and link of the DBDGEN source, this time using macros supplied by Natural.

- The final step executes the Natural batch utility `NDPBNDB0`, which uses the linked DBD module from the previous step to create NDB control blocks which are stored on the `FDIC` system file. `NDPBNDB0` dynamically loads the Natural module `NDLB0001`, which therefore must be present in an allocated load library.

The `NATDBD` procedure assigns a short name of two bytes to each DL/I field; that is, to each field defined in the DBD. All field short names are generated in the range from `NA` to `Z9`, which means that up to 13 * (26 + 10) = 468 DL/I fields can be managed per DBD. DL/I short names are generated uniquely within an NDB.

When replacing an NDB, `NATDBD` reassigns short names in a consistent way; that is, the same short name to the same field name. In addition, the UDFs are maintained, where the new NDB contains the new DL/I layout followed by the old UDF layout, which means that UDFs are not deleted by `NATDBD`. It is the administrator's responsibility to edit the segment description after `NATDBD` has been executed, in order to modify the UDFs accordingly.

Using Logical Databases with Natural

The following information must be considered when using logical databases with Natural:

- Execute the `NATDBD` procedure for a logical database only after successful execution of the procedure for the physical databases referred to. In other words, if the input DBD is a "logical" DBD, the NDBs generated from the "physical" DBDs must already be stored in the Natural `FDIC` system file to correctly generate the NDB control blocks related to this segment.

- When a segment specifying the `SOURCE=`*keyword* is processed by the `NATDBD` procedure, the related "physical" DBD must already be stored in the Natural `FDIC` system file.

  If the `SOURCE=`*keyword* is specified (in one or more segments) in a "physical" DBD, which means that one or more logical virtual child segments are involved (recursively or not), the NATDBD procedure run against this DBD stores the NDB structure on the Natural FDIC system file even if one or more physical DBDs referred to by the `SOURCE=`*keyword* specifications have not already been stored.

In this case, the logical virtual child segments whose source DBD is not yet in the Natural FDIC system file as well as their descendants are not accessible to the user since Natural has marked these segments as inhibited. An appropriate Natural error message is issued indicating the name(s) of the related physical DBD(s) that need to be stored into the Natural system file.

If the logical relationship is the result of a recursive database structure, the `NATDBD` procedure for the physical DBD must be run at least twice: the first time, the NDB is stored on the Natural system file with the undefined segment marked as inhibited; the second time, the reference to the `SOURCE` segment is resolved.

If multiple physical databases are logically related, the `NATDBD` procedure must be run for each of these physical databases and then rerun for any database that contained logical child segments marked as inhibited.

- If the `SOURCE=keyword` is specified in a "logical" DBD and one or more source DBDs are not found in the Natural `FDIC` system file while running the `NATDBD` procedure, the NDB structure is not stored and an appropriate error message is returned.

- If an attempt is made to generate a DDM for a segment whose NDB control blocks are not in the Natural `FDIC` system file, a Natural error message is returned.

Using Index Databases with Natural

The following information must be considered when using index databases with Natural:

- To access a secondary index database as data, the secondary index database must be defined as an independent physical database to both DL/I and Natural.

- The NATDBD procedure need not be executed for primary or secondary index DBDs.

Procedure NATUDF

The DBDGEN source usually does not define all fields within a segment. Additional segment fields called User-Defined Fields (UDFs) can be entered as part of creating the DDMs. UDFs define the additional data in the segment that can be referenced by a Natural program. UDFs can be generated online using Predict or the Natural *SYSDDM Utility*, or they can be generated in batch mode using the `NATUDF` procedure.

The `NATUDF` procedure invokes the batch utility `NDPBCUDF`, which stores segment description layout information on an `FDIC` system file.

⚠️ **Important:** Before `NDPBCUDF` can be executed, the DL/I DBD must have been stored as an NDB on the `FDIC` system file, and a DBID and FNR must have been assigned (with Predict or `SYSDDM`) to each segment concerned. Otherwise, `NDPBCUDF` cannot read the segments concerned.

The input for this utility is provided by the segment description read from a work file. This work file contains segment identification statements and segment field descriptions.

You can format data by using either delimiter mode (`IM=D`) or forms mode (`IM=F`); see also the `IM` profile parameter in the Natural *Parameter Reference* documentation. In delimiter mode, the delimiter character can be used. In forms mode (for example, if input is passed from other programs), input data fields are assumed to be in contiguous storage and must be filled up to the internally defined full length.

One line is required for the segment identification statement, and two lines are required for each segment field description.

The section below covers the following topics:

- Segment Identification Statement
- Segment Field Description

Segment Identification Statement

One line has to be supplied for each segment being defined. The following syntax is used (the parameters must be specified in the sequence shown below):

```
FUNC=(function),DBD=dbd-name,SEGM=segment-name
```

function	Function to be applied to the segment:
	ADD to create a new segment layout;
	REP to replace an existing segment layout;
	MOD to add or modify fields without deleting existing fields not present in the input file;
	END to indicate termination of the UDF redefinition.
dbd-name	A 1 to 8 character alphanumeric DBD name; that is, the name of the DL/I DBD which owns the segment to be defined.
segment-name	A 1 to 8 character alphanumeric name of the DL/I segment to be defined.

Segment Field Description

The segment identification statement has to be followed by at least one segment field description. The following syntax is used for each field to be defined (the parameters must be specified in the sequence shown below):

```
FUNC=FLD,NAME=fnam,TYPE=type,LEVEL=lev,LENGTH=lgh,MAXOCC=moc,VAR=var
FUNC=STR,BEGIN=begin
```

After each FLD card, a STR card must be coded, except for the last FLD card, which is specified with four dollar signs ($$$$) in the field name. After this last FLD card, an END card must be coded.

fnam	The name of the field being defined. This must be an alphanumeric value of 1 to 19 bytes. The value $$$$ closes the definition of the current segment.
type	The UDF field format (1 character). The following formats can be specified: A, B, F, P, U, N, S.
lev	The field level (1 digit).
lgh	The field length (4 digits).
moc	The maximum occurrence (3 digits) of the field (only applicable for a multiple-value field or a periodic group).
var	Possible values: V variable field length N fixed field length
begin	The starting position of the field being redefined. This can be specified either in terms of bytes relative to the beginning of the segment or as a field name of the DL/I field being redefined. The value must be alphanumeric and 1 to 19 characters long (32 bytes in forms mode, as the field is 32 characters long in this mode).

The short name is automatically assigned by the utility in the range from AA to G9, excluding EA to E9. The range from HA to M9 is reserved for UDFs of logical child segments. Thus, up to 216 fields can be provided as input, which is the maximum number of UDF fields.

For further information on UDF field parameters, please refer to *DL/I Services*.

Delimiter Mode (IM=D) Example:

```
FUNC=REP,DBD=ED02DBD,SEGM=COURSE
FUNC=FLD,NAME=GENG1,TYPE=N,LEVEL=1,LENGTH=5
FUNC=STR,BEGIN=11
FUNC=FLD,NAME=DUM1,TYPE=A,LEVEL=1,LENGTH=6
FUNC=STR,BEGIN=TITLE
FUNC=FLD,NAME=DUM2,TYPE=A,LEVEL=1,LENGTH=6
FUNC=STR,BEGIN=DESCRIPN
FUNC=FLD,NAME=GENG3,LEVEL=1,MAXOCC=2
FUNC=STR,BEGIN=GENG1
```

```
FUNC=FLD,NAME=GRU21,TYPE=N,LEVEL=2,LENGTH=1
FUNC=STR
FUNC=FLD,NAME=GRU22,TYPE=A,LEVEL=2,LENGTH=2
FUNC=STR
FUNC=FLD,NAME=GRU23,TYPE=N,LEVEL=2,LENGTH=3
FUNC=STR
FUNC=FLD,NAME=$$$$
FUNC=REP,DBD=ED02DBD,SEGM=COURSE
FUNC=FLD,NAME=DUM41,TYPE=B,LEVEL=1,LENGTH=9
FUNC=STR,BEGIN=DESCRIPN
FUNC=FLD,NAME=DUN2,LEVEL=1,MAXOCC=2
FUNC=STR,BEGIN=TITLE
FUNC=FLD,NAME=GRU21,TYPE=N,LEVEL=2,LENGTH=1
FUNC=STR
FUNC=FLD,NAME=GRU22,TYPE=A,LEVEL=2,LENGTH=2
FUNC=STR
FUNC=FLD,NAME=GRU23,TYPE=N,LEVEL=2,LENGTH=3
FUNC=STR
FUNC=FLD,NAME=$$$$
FUNC=END
```

Forms Mode (IM=F) Example:

```
ADDDBD1    SEGM1
FLD               1FIELD-1                    000A0012N000
STR
FLD               1FIELD-ANY                  000A    N000
STRFIELD-1
FLD               2FIELD-ANY2                 000A0024N000
STR
FLD                $$$$
STR
REPDBD2    SEGM2
FLD               1NEW-FIELD-NAME             000A0012N000
STR
FLD                $$$$
END
```

Sample JCL:

```
//NATUDF   JOB  ......
//NATUDF   EXEC PGM=NATBATCH,PARM='...'
//STEPLIB DD  DSN=...
//        DD  DSN=...
//SYSUDUMP DD  DUMMY
//CMPRINT  DD  SYSOUT=Y
//DDCARD   DD  DSN=NAT23n.SRCE(ADAPARM),DISP=SHR
//CMSYNIN  DD *
LOGON SYSDDM
NDPBCUDF
FUNC=REP,DBD=ED02DBD,SEGM=COURSE
```

```
FUNC=FLD,NAME=DUM1,TYPE=A,LEVEL=1,LENGTH=6
FUNC=STR,BEGIN=TITLE
FUNC=FLD,NAME=DUM2,TYPE=A,LEVEL=1,LENGTH=6
FUNC=STR,BEGIN=DESCRIPN
FUNC=FLD,NAME=$$$$
FUNC=END
FIN
```

Generation of DDMs from DL/I Segment Types

DDMs that represent DL/I segment types are generated from information contained in the NDB and UDF control blocks. These DDMs contain all fields that have been defined for the segment, both in the NDB and in the UDF.

In addition, the DDMs contain the fields from the ancestor segments that have been defined in the DBDGEN for these segments. Ancestor segments are defined as segments that form the hierarchical path from the root segment down to the current segment. Ancestor segment fields that might have been defined in the DBDGEN for a segment include sequence fields, secondary index fields and search fields.

The DDM for a DL/I segment contains all fields that could be specified in the segment search argument (SSA), all fields that are available as part of the key feedback area and any segment I/O fields as well. Each DDM, therefore, contains all the fields that Natural requires to automatically build the concatenated key for the segment.

Once all fields have been defined for a specific segment DDM, the corresponding Natural DDM can be generated and cataloged (stored) on the Natural `FDIC` system file. This is done either with Predict or with the Natural *SYSDDM Utility*.

If you do not have Predict installed, use the `SYSDDM` function **DL/I Services** to generate Natural DDMs from DL/I segment types. This function is invoked from the main menu of `SYSDDM`.

35 System File Structure

As described in section *Accessing DL/I Data*, certain information must be stored and maintained on the Natural `FDIC` system file in order to access DL/I data. This information describes the database structure, the segment data and the processing intent of an application. Four elements on the Natural `FDIC` system file contain this information. One of these elements, the Natural DDM, is common to all DBMS environments. The remaining three elements, however, are used only by Natural for DL/I; they are NDB control blocks, NSB control blocks and UDF control blocks. Therefore, the Natural `FDIC` system file used by Natural for DL/I contains three subfiles.

This section covers the following topics:

NDB Subfile

The NDB subfile contains the NDBs. The NDB, or Natural DBD, control blocks contain most of the information present in the DL/I DBD, combined with additional data used by Natural, such as the file number (FNR) and database identification (DBID) of the segment, and short names for fields defined in the DBD. The NDB control blocks are created and stored on the Natural `FDIC` system file by the `NATDBD` procedure.

An NDB consists of the following fields:

Field	Description
`ND`	DBD name (8 characters) combined with sequence number (1 byte, binary).
`NC`	The first two bytes contain the number of `NZ` fields in the record times 20. The second two bytes contain the total number of `NZ` fields in the NDB multiplied by 20.
`NZ`	NDB data.

NSB Subfile

The NSB subfile contains the NSBs. The NSB, or Natural PSB, control blocks contain most of the information present in the DL/I PSB. These control blocks are created and stored on the Natural `FDIC` system file by the NATPSB procedure.

An NSB consists of the following fields:

Field	Description
NP	PSB name (8 characters) combined with sequence number (1 byte, "binary").
NC	The first two bytes contain the number of NZ fields in the record times 20. The second two bytes contain the total number of NZ fields in the NSB multiplied by 20.
NZ	NSB data.

UDF Subfile

The UDF subfile contains the UDFs. The UDF, or User-Defined Field, control blocks contain information on segment fields which have been specified by the user, either through the online **DL/I Services** function of the *SYSDDM Utility*, the NATUDF procedure, or by using Predict.

The fields are as follows:

Field	Description
NS	Database identification (1 byte, "binary"), file number (1 byte, "binary") and sequence number (1 byte, "binary"). The DBID and FNR are those of the segment being described by this record.
NC	The first two bytes contain the number of NZ fields in the record times 20. The second two bytes contain the total number of NZ fields in the UDF multiplied by 20.
NZ	Field description as specified by the user using Predict, the EDIT segment layout facility of the *SYSDDM Utility* or the procedure NATUDF.
NW	The long field name.

Natural for DL/I Objects

Natural for DL/I objects are created during execution of the NATPSB procedure (NSB), during execution of the NATDBD procedure (NDB)), or when **assigning DBID/FNR** to a segment type (UDF). Consequently, at least one UDF block for each segment type with an assigned DBID/FNR is always present on FDIC whether or not user-defined fields (UDF fields) have been defined by the user.

When displaying type definitions in the *SYSDDM Utility*, the NDB and its related UDF are combined automatically. The only way to display an UDF separately (for debugging purposes) is by using NDLBLOCK.

Displaying Keys of UDF Blocks

The utility program `NDLULIST`, cataloged in the library `SYSDDM`, is provided for listing the keys of all UDF blocks and for checking for duplicates.

For each duplicate found the following warning is issued:

```
More than one record with same DBID/FNR
```

Displaying the Size of Natural for DL/I Objects

The following utility programs, cataloged in library `SYSDDM`, are provided for displaying the sizes of the various Natural for DL/I objects:

- `NDLSIZED` displays the sizes of all NDBs stored on `FDIC`.

- `NDLSIZEP` displays the sizes of all NSBs stored on `FDIC`.

- `NDLSIZEU` displays the sizes of all UDFs stored on `FDIC`.

Displaying Natural for DL/I Objects

The utility program `NDLBLOCK`, cataloged in library `SYSDDM`, is provided for displaying the NDBs, NSBs and UDFs stored on `FDIC`. The utility displays the objects in hexadecimal format.

Control Blocks in Separate Buffer Pool

The Natural for DL/I control blocks NDB, NSB and UDF are read from `FDIC` and loaded into a buffer pool - resident or not, depending on the `NDLPARM` parameters `RESINDB`, `RESINSB`, and `RESIUDF`. This allows a given object to be shared by several users.

By means of the `NTBPI` macro (as described in the Natural *Parameter Reference* documentation) it is possible to have a buffer pool for NDB, NSB and UDF control blocks which is different from the buffer pool for Natural programs, thus allowing for better isolation between the different Natural objects.

If a separate buffer pool is allocated, Natural for DL/I locates its control blocks in this buffer pool. Otherwise, they are located in the Natural buffer pool.

The **List Objects** function of the SYSBPM utility displays the NDB, NSB and UDF control blocks kept in the buffer pool as follows:

	Library	DBID	FNR
NDB	SYSDLIND	255	253
NSB	SYSDLINS	255	253
UDF	Udddfff	255	253

> **Notes:**

1. The library names of NDB and NSB are fixed internal names and are not related to any Natural library.

2. The DBID/FNR values are fixed internal values and are not related to any Natural system file.

3. *ddd* is the DBID of the corresponding segment, *fff* is the FNR of the corresponding segment.

The **Hexadecimal Display** function of the SYSBPM utility also allows you to display Natural for DL/I objects. This function might be useful when in doubt if the expected object has been read from FDIC, or if the object has been read from the expected FDIC (test/production).

Control Blocks in Buffer Pool Blacklist

The Natural for DL/I control blocks NDB, NSB and UDF can be added to the buffer pool blacklist.

This is done by the **Blacklist Maintenance** function of the SYSBPM utility.

As "Library" you enter SYSDLIND for NDBs, SYSDLINP for NSBs, and SYSDLINS for UDFs.

As **Object** you enter the NDB name for NDBs, the NSB name for NSBs, and *Udddfff* for UDFs where *ddd* is the DBID and *fff* the FNR of the corresponding segment.

This feature allows you to modify NDBs, NSBs or UDFs without causing unpredictable results for active users.

If an attempt is made to load a locked object into the buffer pool, Natural for DL/I will issue error message NAT3935.

Natural for DL/I Objects and Natural DDMs

When referencing a DDM in a Natural program, Natural translates the DDM name into the corresponding DBID/FNR pair. If this DBID identifies the DDM as a DL/I DDM (by means of the `NTDB` macro), the Adabas control block is passed to Natural for DL/I for further processing.

Natural for DL/I takes the DBID from the control block and tries to locate an UDF with this DBID/FNR in the buffer pool. If it is not found there, it is read from `FDIC` and loaded into the buffer pool.

The UDF contains the name of the related NDB in its header. Using this name, Natural for DL/I tries to locate the NDB in the buffer pool. If it is not found there, it is read from `FDIC` and loaded into the buffer pool.

The segment description including all DL/I fields is part of the NDB.

From this it is clear that:

- The NDB/UDF is required during runtime.
- The relation between the Natural program and the related NDB is established by means of DBID/FNR only.

This implies that the DBA has to ensure that DDMs and NDBs are always kept in synchronization. For example, it is not sufficient to transfer only the Natural programs from test to production.

36   **Natural Batch Utilities**

This section covers the following topics:

Transfer of NDBs/NSBs/UDFs from one System File to Another

- Unloading the NDBs, NSBs and UDFs
- Loading NDBs, NSBs and UDFs
- Selecting NDBs, NSBs and UDFs from a Data Set

The transfer of NDBs, NSBs and UDFs from one `FDIC` system file to another is performed either online using the utility `SYSMAIN` (as described in the Natural *Utilities* documentation) or in two batch steps, using two Natural batch utilities provided for this purpose:

- With the `ULDDLI` unload utility, the NDBs, NSBs and UDFs are transferred from one `FDIC` system file to a sequential work file.

- With the `INPLDLI` load utility, the NDBs, NSBs and UDFs are transferred from the sequential work file to another `FDIC` system file.

Both programs, `ULDDLI` and `INPLDLI`, are contained in the library `SYSDDM`.

Unloading the NDBs, NSBs and UDFs

The utility `ULDDLI` is used to unload NDBs, NSBs and UDFs from an `FDIC` system file to a sequential work file.

`ULDDLI` requires the following input:

- the specification of the `FDIC` system file to be unloaded (either in the Natural parameter module or dynamically) and

- one or more parameter lines containing the following:

  - **Function code** (A1); the following function codes can be specified:

A	All NSBs, NDBs and UDFs are unloaded.
D	All NDBs with valid object names and their UDFs are unloaded. If no object names are specified, all NDBs and their UDFs are unloaded.
P	All NSBs with valid object names are unloaded. If no object names are specified, all NSBs are unloaded.
U	All UDFs with valid object names are unloaded. If no object names are specified, all UDFs are unloaded.
. (period)	Terminate `ULDDLI`; at least one parameter card with function code "." is required.

  - **Object name** (A8); 0 - 6 occurrences.

> **Note:** With UDFs, the object name must be in the form *nnn**nnn*; that is, a 3-digit database ID, followed by 2 asterisks, followed by a 3-digit file number.

Work files: `CMWKF01` DD card must be provided with:

```
DCB=(RECFM=VB,LRECL=4624,BLKSIZE=4628)
```

When `ULDDLI` is executed, the specified NDBs, NSBs and UDFs are written from the `FDIC` system file to the `CMWKF01` data set.

> **Note:** DL/I fields of a segment are part of the NDB block and not of the UDF block, which means that you must still transfer the entire NDB block if you have modified a DL/I field in a segment.

Example 1 - Unload the NDBs TESTDB1 and TESTDB2:

```
LOGON SYSDDM
ULDDLI D TESTDB1 TESTDB2
  .
```

Example 2 - Unload all UDF Blocks:

```
LOGON SYSDDM
ULDDLI U
 .
```

Example 3 - Unload UDF Blocks with DBID 10/FNR 150 and DBID 246/FNR 3:

```
LOGON SYSDDM
ULDDLI U 010**150 246**003
  .
```

Loading NDBs, NSBs and UDFs

The utility `INPLDLI` is used to load NDBs, NSBs and UDFs - previously unloaded with `ULDDLI` - from the work file to an `FDIC` system file.

`INPLDLI` requires the following input:

- the specification of the `FDIC` system file into which the NDBs, NSBs and UDFs are to be loaded (either in the Natural parameter module or dynamically);

- (optionally) the parameter `DEL=Y`:

  If you specify `DEL=Y`, all existing NDBs and UDFs found on the `FDIC` system file are first deleted. The ones contained on the input work file are added to the file. NSB definitions contained on the work file replace any identically named NSBs on the `FDIC` system file.

If you do not specify `DEL=Y`, existing identically named NDBs and NSBs are not replaced. Existing UDFs which have been allocated identical DBID/FNR combinations are not replaced either. Non-existent definitions are added. If you do not specify `DEL=Y`, it may occur that an NDB is loaded but all or some of its segments (UDFs) are not, or that segments (UDFs) are loaded without the corresponding NDB being loaded.

- (optionally) the parameter `REP=Y`: If you specify `REP=Y`, NDBs, NSBs and UDFs contained on the work file replace any identically named NDBs, NSBs and UDFs on the `FDIC` system file.

`DEL=Y` and `REP=Y` are mutually exclusive. If neither `DEL=Y` nor `REP=Y` is specified, existing NDBs, NSBs and UDFs are neither deleted nor replaced.

Work files: `CMWKF01` DD card must be assigned to the work file which was created by the utility program `ULDDLI`.

When `INPLDLI` is executed, the NDBs, NSBs and UDFs are loaded from the work file into the specified `FDIC` system file, depending on whether they already exist and on whether `DEL=Y` was specified.

Example:

```
LOGON SYSDDM
INPLDLI REP=Y
```

Selecting NDBs, NSBs and UDFs from a Data Set

The utility `SELDLI` allows you to select Natural for DL/I objects (NDBs, NSBs, UDFs) from a data set created by the `ULDDLI` utility. The output of `SELDLI` can be used as input for `INPLDLI`. Since `INPLDLI` does not allow to select objects from a data set created by `ULDDLI`, you can use `SELDLI` to perform this function on desired objects prior to running `INPLDLI`.

`SELDLI` can, therefore, be used for backup/recovery or transfer of selected objects from test to production.

`SELDLI` also supports a `SCAN` (command `SCN`) feature that will list all of the objects on the input data set without selecting any for output.

`SELDLI` can be used in batch mode only.

`SELDLI` requires the following input:

- the specification of the output data set `CMWKF01` from `ULDDLI`
- up to 30 parameter lines containing the following:
  - Object type (A3); the following types can be specified:
    - NSB - Select specified NSB
    - NDB - Select specified NDB

- NDU - Select specified NDB and related UDF
- UDF - Select specified UDF
- SCN - List input data set `CMWKF01`
- terminate `SELDLI`
- Object name (A8); 1 occurrence

> **Notes:**

1. With NDB/NSB, a wildcard (*) can be specified at the end of the name to select a range of names.

2. With UDFs, the object name must be in the form *nnn**nnn*; that is, a 3-digit database ID, followed by 2 asterisks, followed by a 3-digit file number.

`SELDLI` provides the following output:

- Data set containing selected objects to be used as input to `INPLDLI`. It is specified with DDNAME `CMWKF02`.

When `SELDLI` is executed, the specified NDBs, NSBs and UDFs are copied from `CMWKF01` to `CMWKF02`.

Example 1 - Select all NDBs:

```
LOGON SYSDDM
SELDLI
NDB,*
.
FIN
```

Example 2 - Select NSB "ORDPSB" and UDF for DBID 151, FNR 3:

```
LOGON SYSDDM
SELDLI
NSB,ORDPSB
UDF,151**003
.
FIN
```

Example 3 - Select NDB "CUSTDBD" and its related UDFs:

```
LOGON SYSDDM
SELDLI
NSB,ORDPSB
NDU,CUSTDBD
.
FIN
```

Example 4 - List all objects on the input data set:

```
LOGON SYSDDM
SELDLI
SCN
FIN
```

Utility NDUDFGEN for Natural Data Areas

The batch utility `NDUDFGEN` can be used to generate Natural data areas.

Input is provided by a UDF definition read from a work file.

Two kinds of data areas can be generated:

- a Natural view,
- a data structure (local data area).

A view in a local data area is generated from all fields contained in the input work file. The utility normalizes the data to the requirements of a view according to the Natural syntax. The field lengths are adapted to Natural field lengths, multiple-value fields and periodic groups are generated from record data structures. Arrays are generated by `NDUDFGEN` with the maximum length allowed by Natural. Field definitions are collected into a redefinition and the redefined field is generated according to the length of the individual fields collected. The generated field can then be used in the segment description as UDF; this means that not all UDFs need to be defined in the segment description, but only the generated fields.

A data structure as local data area is generated of all input fields. A level increment value can be specified for the fields. No other modifications to the input file data are permitted, so that the data are generated as specified in the input file.

Input for NDUDFGEN

The input layout is similar to the one for the `NDPBCUDF` utility.

The first card is the definition card; it contains the definition which is valid for all of the UDF definitions.

The FLD cards contain the actual field definitions and are separated from each other by STR cards.

The END card indicates the end of the field definitions. The input is required in forms mode (`IM=F`) as follows:

Definition Card

Definition	Explanation
Bytes 1 - 3	The first 3 bytes are not used.
Bytes 4 - 11	These 8 bytes contain the DBD name.
Bytes 12 - 19	These 8 bytes contain the segment name.
Bytes 20 - 27	These 8 bytes contain a prefix (generated for fields).
Bytes 28 - 30	These 3 bytes contain the maximum occurrence (default is 191).
Byte 31	This byte contains either S if a data structure is to be generated or V if a view is to be generated.
Byte 32	This byte contains the level increment.

Field Card

Definition	Explanation
Bytes 1 - 3	The first 3 bytes contain FLD.
Bytes 4 - 19	These 16 bytes are not used.
Byte 20	This byte contains the field level.
Bytes 21 - 39	These 19 bytes contain the name of the field being defined. This must be an alphanumeric value.
Bytes 40 - 42	These 3 bytes are not used.
Byte 43	This byte contains the format of the field.
Bytes 44 - 47	These 4 bytes contain the byte length of the field.
Byte 48	This byte is not used.
Byte 49 - 52	This byte contains the length as required by Natural (if this length is specified, the byte length is ignored).
Byte 53 - 57	These 4 bytes contain the maximum size of the 1st dimension of an array.
Byte 58 - 62	These 4 bytes contain the maximum size of the 2nd dimension of an array.
Byte 63 - 66	These 4 bytes contain the maximum size of the 3rd dimension of an array.

Example 1 - View Generation:

```
    DBDNAME SEGMENT PREFIX   191V
 FLD             1VAR1                000A0745
 STR
 FLD             1GROUP               000A0000N0000000200020000
 STR
 FLD             2VAR2                000A0006N00060005
 STR
 FLD             2VAR3                000A0030
 STR
 END
```

The above input generates the following view:

```
13:38:41              *****  E D I T  DATA  *****              2006-05-25
Library: XYZ1      Name:        LOCAL              DBID:  10 FNR:   5
Command:                                                        > +
I T L Name                            F Leng  Index/Init/EM/Name/Comment
- - - ------------------------------ - ---------------------------------
    1 VAR1                            A  149 (5)
    1 GROUP                                  (4)
    2 VAR2                            A    6 (5)
    2 VAR3                            A   30
```

Example 2 - Structure Generation:

```
    DBDNAME SEGMENT PREFIX   191S
FLD               1VAR1                000A0745
STR
FLD               1GROUP               000A0000N0000000200020000
STR
FLD               2VAR2                000A0006N00060005
STR
FLD               2VAR3                000A0030
STR
END
```

The above input generates the following data structure:

```
13:41:20              *****  E D I T  DATA  *****              2006-05-25
Library: XYZ1      Name:        LOCAL              DBID:  10 FNR:   5
Command:                                                        > +
I T L Name                            F Leng  Index/Init/EM/Name/Comment
- - - ------------------------------ - ---------------------------------
  V 1 DBDNAME-SEGMENT-VIEW                     DBDNAME-SEGMENT
  M 2 VAR1                            A  149 (5)
  P 2 GROUP                                  (4)
    3 PREFIX-1                        A   60 /*PREFIX-1
  R 3 PREFIX-1
    4 VAR2                            A    6 (5)
    4 VAR3                            A   30
```

37 Execution

This section covers the following topics:

PSB Scheduling

In all environments, Natural must know the name of the scheduled PSB, not only the address of the PCB list. In the online environments, the application developer must have the ability to change the scheduled PSB during a Natural session. This is accomplished by the Natural command `NATPSB` (in batch or CICS environments) or by calling `CMDEFSWX`/`CMDIRSWX` (in IMS TM environments).

- The NATPSB Command
- PSB Scheduling in a Batch Environment
- PSB Scheduling in a CICS Environment
- PSB Scheduling in an IMS TM Environment

The NATPSB Command

The `NATPSB` command handles PSB scheduling status and can be invoked with one of the following three options:

Option	Description
`INQ`	Performs an inquiry on PSB scheduling status.
`ON psbname`	Issues a PSB schedule of the PSB `psbname`.
`OFF`	Issues a `SYNCPOINT` to commit all updates and terminate the PSB.

> **Note:** The `NATPSB INQ` command is valid in an IMS TM environment, too.

The following command, for example, issues a PSB schedule of `ED00PSB`:

```
NATPSB ON ED00PSB
```

A PSB scheduling operation is allowed only if there is no active PSB. If a PSB is active and another PSB is to be scheduled, the `ON` request for this new PSB must be preceded by an `OFF` request. Otherwise, the following message is issued:

```
NAT3900 PSB ... scheduled, but PSB ... already active
```

Since `NATPSB` is actually a Natural program, it can also be invoked with a `FETCH` or `FETCH RETURN` statement. The options described above should then be passed in the `FETCH` statement as two parameters. The first parameter would be an alphanumeric field of three bytes for `INQ`, `ON` or `OFF`. If the first parameter is `ON`, the second parameter must also be passed. It is an alphanumeric field of eight bytes and contains the name of the PSB to be scheduled.

Execution time errors of NATPSB can be intercepted by an ON ERROR statement. The error messages from NAT3900 to NAT3903 and from NAT3817 to NAT3820 are generated by NATPSB.

Example:

```
FETCH RETURN 'NATPSB' 'ON' 'PBNDL01'
ON ERROR
  IF *ERROR = 3900                         /* PSB already scheduled
    STACK TOP COMMAND 'NATPSB' 'ON' PBNDL01'
    STACK TOP COMMAND 'NATPSB' 'OFF'
    STOP
  END-IF
END-ERROR
END
```

PSB Scheduling in a Batch Environment

To execute a batch program that accesses a DL/I database, it is necessary to use the DL/I batch procedure which executes an application program under DL/I control. Therefore in the JCL/JCS used to execute Natural batch accessing DL/I databases, the first program in the step is a DL/I system program (DFSRRC00).

PSB scheduling is performed by DL/I before control is passed to Natural. Since Natural requires the name of the scheduled PSB, it is necessary to invoke the Natural PSB scheduling program NATPSB before executing a Natural application program. This can be achieved by specifying the command NATPSB ON *psbname* as the first command in the batch input stream to Natural.

Batch Execution under z/OS

Under z/OS, the DL/I region controller program (DFSRRC00) invokes the NDLSINIB bootstrap module for Natural for DL/I by specifying MBR=NDLSINIB in the PARM field of the EXEC card. NDLSINIB reads two statements from the NDINPUT DD card:

■ Statement 1 contains the name of the Natural module to be executed.

■ Statement 2 contains the dynamic Natural parameters.

Before executing the user program, the command NATPSB ON *psbname* must be specified in the input stream to pass the name of the current PSB to Natural.

Example 1 - z/OS with Adabas System File:

```
//          EXEC DLIBATCH,PSB=psbname,MBR=NDLSINIB
//G.STEPLIB DD ...            Steplibs
//G.NDINPUT DD *              Input for NDLSINIB
natbatch                      Natural load module name
STACK=(LOGON user),DU=ON      Any Natural parameters
//DDCARD    DD *              Primary input file
ADARUN MODE=MULTI,PR=USER     ADARUN cards
//G.CMSYNIN DD *              Primary input file
NATPSB ON psbname             Mandatory Natural PSB scheduling
pgmname                       Natural user program name
/*                            End of Natural commands
```

Example 2 - z/OS with VSAM System File:

```
//          EXEC DLIBATCH,PSB=psbname,MBR=NDLSINIB
//G.STEPLIB DD ...            Steplibs
//G.NDINPUT DD *              Input for NDLSINIB
natbatch                      Natural load module name
STACK=(LOGON user),DU=ON      Any Natural parameters
//G.CMSYNIN DD *              Primary input file
NATPSB ON psbname             Mandatory Natural PSB scheduling
pgmname                       Natural user program name
/*                            End of Natural commands
```

In both examples, *natbatch* is assumed to be the load module produced by the respective link-edit procedure.

PSB Scheduling in a CICS Environment

Under CICS, the PSB must be scheduled using the NATPSB command, which actually invokes the appropriate scheduling or termination calls.

The active PSB can be changed dynamically during the Natural session using the NATPSB command. Therefore, more than one PSB can be used during a Natural session. Only one PSB, however, can be active for a CICS task at a time.

The NATPSB command can be entered in the Natural Command line or passed to Natural dynamically with the Natural STACK statement when starting a Natural session.

Examples:

```
MOVE 'STACK=(NATPSB ON EDOOPSB)'
    TO DYNAMIC-PARM-KEYWORD-LIST.
EXEC CICS
    XCTL PROGRAM('NATvrs')
END-EXEC.
```

This example taken from a COBOL/CICS program assumes that `NATvrs` is the value supplied for the `PROGRAM` keyword in the CICS PPT; where `vrs` is the current Natural version number.

Another possibility is to assign `NATPSB` commands to one or more PF keys when starting a Natural session as illustrated in the following example:

```
NATD STACK=(KEY PF1 = EDOOPSB)
```

This example assumes that `NATD` is the value supplied for the `TRANSID` keyword in the CICS PCT. `EDOOPSB` is the following Natural program (cataloged in the library `SYSTEM`):

```
STACK TOP COMMAND 'NATPSB ON EDOOPSB'
STACK TOP COMMAND 'NATPSB OFF'
END
```

Whenever PF1 is pressed, the commands `NATPSB OFF` and `NATPSB ON EDOOPSB` are executed.

PSB Scheduling in an IMS TM Environment

Under IMS TM, Natural for DL/I runs as a conversational transaction. It has the ability to perform direct or deferred message switching. This means that several different Natural transactions and PSBs can be invoked during a single Natural session. It is also possible to invoke multiple PSBs and provide the user with access to databases defined in different PSBs. This is accomplished by calling `CMDEFSWX` or `CMDIRSWX`.

Under IMS TM, PSB scheduling is performed by the IMS Control Region before control is passed to the Natural transaction running as an MPP (Message Processing Program) or BMP (Batch Message Processing). As in the batch environment, Natural needs to know the name of the scheduled PSB. This is accomplished internally at Natural session start by the driver which stores the pointer to the PCB address list and the name of the PSB into IOCB fields. The `NATPSB INQ` command can be issued in this environment but the `NATPSB ON`/`NATPSB OFF` commands cannot.

CALLNAT Interface

The Natural subprograms `NDLPCBAD` and `NDLPSBSC` are provided, which can be invoked with a `CALLNAT` statement from within a Natural program.

See the following sections:

- NDLPCBAD Subprogram
- NDLPSBSC Subprogram

NDLPCBAD Subprogram

The Natural subprogram `NDLPCBAD` provides the calling Natural program with the name of the currently scheduled PSB and the pointer to the PCB address list.

Example:

```
DEFINE DATA LOCAL
01 PSBNAME  (A8)
01 PCBADDR  (B4)
END-DEFINE
CALLNAT 'NDLPCBAD' PSBNAME PCBADDR
DISPLAY PSBNAME PCBADDR
END
```

This pointer can then be used by non-Natural programs to obtain the individual PCB addresses and to establish addressability to the PCBs. For example, move these addresses to the BLL cells (COBOL/VS) or use the `SET ADDRESS` instruction (COBOL II).

NDLPSBSC Subprogram

The Natural subprogram `NDLPSBSC` allows for scheduling a PSB in CICS or batch environments. It performs the same functions as the `NATPSB` command.

Using `CALLNAT 'NDLPSBSC'` (instead of `FETCH RETURN 'NATPSB'`) avoids the NAT1108 error message, which is issued if a PSB is scheduled in an `INPUT` loop as follows:

```
INPUT ...
FETCH RETURN 'NATPSB' 'ON' 'psbname'
REINPUT ...                        /* returns NAT1108
```

Example:

```
DEFINE DATA LOCAL
01 COMMAND  (A3)
*  'ON'
*  'OFF'
*  'INQ'
01 PSBNAME  (A8)
01 RETCODE  (B1)
*  01: Command invalid
*  02: PSB name missing
*  03: PSB psbname active
*  04: PSB psbname not active
*  05: Not used
*  06: No PSB active
END-DEFINE
MOVE 'ON' TO COMMAND
MOVE 'psbname'TO PSBNAME
CALLNAT 'NDLPSBSC' COMMAND PSBNAME RETCODE
DISPLAY PSBNAME RETCODE
END
```

Under IMS TM, `NDLPSBSC` can only be used with parameter `'INQ'`, because PSB scheduling is performed by the IMS control region before control is passed to Natural.

Support of IMS-Specific Features

This section covers the following topics:

- Symbolic Checkpoint/Restart Functions - CHKP, XRST
- The INIT Call to Enable Data Availability Status Codes

Symbolic Checkpoint/Restart Functions - CHKP, XRST

A Natural program can make use of the IMS TM symbolic checkpoint and restart facilities by using the statements `GET TRANSACTION DATA` and `END TRANSACTION`.

The executing program can checkpoint user data on the IMS system log data sets by supplying an 8-byte checkpoint ID as the first operand in the `END TRANSACTION` statement and by specifying the areas to be checkpointed as additional operands.

To ensure that the checkpoints are written to the IMS log data set, the Natural profile parameter `ETDB` (see the Natural *Parameter Reference* documentation) must be specified, and the database specified with the `ETDB` parameter must be a DL/I database.

If no operands are specified with the `END TRANSACTION` statement, Natural uses `NATDLICK` as the default checkpoint ID.

This checkpoint data are retrieved by executing the `GET TRANSACTION DATA` statement. The first operand of this statement must also be an 8-byte checkpoint ID. The remaining operands must be listed in the same sequence, length and format as in the corresponding `END TRANSACTION` statement.

Example:

```
RESET CKPID(A8) KEY(A10) AREA1(A20) AREA2(N6) AREA3(A120)
GET TRANSACTION DATA CKPID KEY AREA1 AREA2 AREA3
IF CKPID NE ' '                                    /* checkpoint restart
  MOVE KEY TO START-KEY(A10)
ELSE
  RESET START-KEY                                  /* normal restart
MOVE *PROGRAM-ID TO CKPID
  :
READ DLI-DB BY XKEY > START-KEY
  :
  UPDATE
  :
  END TRANSACTION CKPID XKEY AREA1 AREA2 AREA3
  :
END
```

Normal Restart:	Simply run the job. The checkpoint ID parameter in the program's `GET TRANSACTION DATA` statement is set to blanks by the DL/I call handler `NDLSIOBA`.
Checkpoint Restart:	To restart after an abnormal termination, specify one of the following checkpoint IDs in the `PARM` field of the `EXEC` statement in your program's JCL:
	`CKPTID=LAST` to restore data areas written to the log by the job at the last successful checkpoint; or
	`CKPTID=cccccccc` to restore data areas written with checkpoint ID *cccccccc*.

These are the usual IMS TM restart procedures. Each checkpoint ID used in an `END TRANSACTION` statement is displayed in the job output once the extended checkpoint has been successfully executed by IMS.

The checkpoint ID parameter of the program's `GET TRANSACTION DATA` statement is set to the actual checkpoint ID used by IMS.

The data areas are restored into the areas you specify in your `GET TRANSACTION DATA` statement.

Ensure that the `//IMSLOGR` DD statement specifies the correct IMS log data set.

When Natural is started in a BMP region, the initialization routine issues an `XRST` call, to ensure that symbolic checkpointing is available. This is done whether the Natural user programs to be executed make use of IMS symbolic checkpoint logic or not. If the `XRST` was unsuccessful, Natural returns the following error message:

```
NAT3959 XRST call failed with DL/I status code xx ↵
```

When a `GET TRANSACTION DATA` statement is directed to the Natural call handler and the initial `XRST` call has been flagged as successfully executed, the restart checkpoint ID and contents of this buffer are copied into the program's user fields.

When an `END TRANSACTION` statement is directed to the Natural call handler, the user fields to be checkpointed are copied into the buffer before a symbolic checkpoint call (`CKPT`) is issued.

If the database specified with the profile parameter `ETDB` (see the Natural *Parameter Reference* documentation) is not the same as the database affected by the transaction, the first operand of the `END TRANSACTION` statement will be used as checkpoint ID for the `ETDB` database, while `NATDLICK` will be used as checkpoint ID for the other database *not* specified with the `ETDB` parameter.

The total area to be checkpointed must not exceed 1992 bytes.

The INIT Call to Enable Data Availability Status Codes

If the `INITCAL` parameter of `NDLPARM` is set to `YES`, Natural issues an `INIT` call during session initialization and during each MPP transaction start. The character string in the I/O area is `STATUS GROUPA`. This informs IMS that Natural is prepared to accept status codes regarding data unavailability. IMS returns status codes `BA` or `BB` when the DL/I call requires access to unavailable data (for example, if the accessed database has been stopped).

The corresponding error messages of Natural for DL/I are:

```
NAT3897 DL/I status code 'BA'
NAT3898 DL/I status code 'BB'
```

For compatibility reasons, the default setting of `INITCAL` is `NO`.

The `INIT` call is issued only if Natural runs in a BMP or MPP region.

Fast Path Support

Natural supports Fast Path databases.

Fast Path database types include Main Storage Databases (MSDB) and Data Entry Databases (DEDB).

■ MSDB:

MSDBs have root only segments that are fixed-length. There are two types of MSDBs: terminal-related and non-terminal-related.

To read segments in an MSDB, `GU` and `GN` are used.

To update segments in an MSDB, `REPL`, `DLET`, `ISRT`, and `FLD` are used.

- DEDB:

  DEDBs use the design concept that database content can be physically partitioned by ranges of root keys or by groupings produced by a randomizing algorithm.

As a basic requirement, the non-conversational `NATIMS` driver must be used. This is because Fast Path programs cannot be conversational programs, that is, they cannot use an SPA.

For DEDB databases, no special processing is required by Natural for DL/I.

For MSDB databases, the (one and only) SSA is built without command codes because DL/I does not allow for it (not even the null command code must be used in case of MSDB databases).

When updating segments in an MSDB database, Natural for DL/I uses the `REPL` call (rather than the `FLD` call) because the `UPDATE` statement of the Natural language does not provide a search condition that indicates which segments must be updated (searched update).

Support of GSAM

Natural for DL/I supports the Generalized Sequential Access Method (GSAM), with which a sequential data set can be handled as a sequential non-hierarchic database by IMS.

Although GSAM databases have no segments, keys or parentage, they are handled internally by Natural as root-only databases with fixed or variable-length segment types. Thus, it is possible to use DDMs instead of work files for GSAM record types.

For variable-length GSAM records, Natural maintains the record length; you need not reserve a field for the record length in the DDM.

A `FIND` or `READ` statement generates a `GN` (get next) call sequence for GSAM. Due to GSAM restrictions, `UPDATE` and `DELETE` statements are not allowed. Due to GSAM restrictions, a `STORE` statement must insert records at the *end* of the database.

IMS repositions GSAM databases for sequential processing, which means that the position need not be re-established by the application program after checkpoint calls. Therefore, Natural performs no repositioning after checkpoint calls in the case of PCBs for GSAM.

In order to use the extended restart feature of IMS, the Natural job has to terminate abnormally. This can be accomplished by calling the Natural IMS TM service module `CMSVC13D`. If the job terminates either normally or with a condition code, IMS does a clean-up and no restart is possible.

Every GSAM database structure which is to be used by Natural must be processed by the `NATDBD` procedure. The assembly step of this procedure extracts the relevant information from the DBD source and simulates an appropriate `SEGM` statement as shown in the following examples.

Example 1 - Segment Description of Fixed-Length GSAM Records:

```
DBD     NAME=TESTDB,ACCESS=(GSAM,BSAM)
DATASET DD1=INPUT,DD2=OUTPUT,RECFM=F,RECORD=80
DBDGEN
END
```

From the above source statements, `NATDBD` would simulate a segment with the name of the DBD and the length as specified with the `RECORD` keyword:

```
SEGM NAME=TESTDB,BYTES=80
```

Example 2 - Segment Description of Variable-Length GSAM Records:

```
DBD     NAME=TESTDB,ACCESS=(GSAM,BSAM)
DATASET DD1=INPUT,DD2=OUTPUT,RECFM=VB
DBDGEN
END
```

From the above source statements, `NATDBD` would simulate a segment with the name of the `DBD`, a maximum length of 32760 and a minimum length of 8:

```
SEGM NAME=TESTDB,BYTES=(32760,8)
```

In both examples, the NDB name and the segment name are `TESTDB`, and the generated DDM name would be `TESTDB-TESTDB`.

The Natural program to read this GSAM database would be as simple as:

```
READ TESTDB-TESTDB
  DISPLAY FIELDS-OF-TESTDB
LOOP
END
```

Processing in CICS Pseudo-Conversational Mode or under IMS TM

When Natural is running under CICS in pseudo-conversational mode (that is, with the parameter `PSEUDO=ON` set in the Natural parameter module) or under IMS TM, the Natural task/transaction is terminated following each write to a terminal, and a new task/transaction is started when new input is entered through the terminal. Because a Syncpoint is forced at the end of the task/transaction, all resources are released when the message is sent to the terminal. Therefore, the DL/I PSB is no longer active, nor are any DL/I `GET HOLD` calls in effect.

To avoid consistency problems on the DL/I databases, Natural performs additional processing when it is running in CICS pseudo-conversational mode or under IMS TM:

1. If a DL/I `GET HOLD` call is still active at the end of the task/transaction, the values of the fields read by the program that issued the corresponding `READ` or `FIND` (only the fields used, not the whole segment) are saved in an internal table of Natural for DL/I.

2. When a new task/transaction resumes the Natural session and the program issues an `UPDATE` or `DELETE` statement, Natural checks whether the field contents have been changed. If the check shows that the field contents have not been changed, the `UPDATE`/`DELETE` is executed. If they have been changed, an error message is returned by Natural notifying the user that the field values just read were changed by another user in the system and that, therefore, the `UPDATE`/`DELETE` operation is not carried out.

Natural also performs automatic PSB repositioning following resumption of the task/transaction. A Natural application is, therefore, not affected by pseudo-conversational mode, unless it uses conventional programming techniques, for example COBOL or PL/1.

If the task/transaction is terminated due to a screen I/O while a `READ` or `FIND` loop is being executed on a segment without a unique sequence field, Natural is not able to reposition the PSB in the database when the task/transaction is resumed. The same may occur when using secondary indices with non-unique key fields in pointer segments. Natural is not able to reposition the PSB in these instances because DL/I does not provide a method of re-establishing position in the middle of non-unique keys or non-keyed segments.

38 Programming Language Considerations

This section covers the following topics:

Natural versus Third Generation Languages

With a few exceptions Natural provides all of the functionality of third generation language programming in the DL/I environment.

However, accessing DL/I data using Natural is significantly different from programming techniques used in a third generation language. Natural application programmers do not have to code specific DL/I calls or build the segment search arguments (SSAs). They do not need to concern themselves with PCB mask information or keep track of PCB positioning between Syncpoints.

Natural for DL/I operates as a standard DL/I application and although most of the DL/I call processing is done internally, it is important to realize that all of the required DL/I processing is still performed:

PSBs are scheduled and terminated, PCBs are selected for use, database positioning is maintained, SSAs are created, the most efficient DL/I calls are issued, PCB mask information is evaluated, `GET HOLD` calls are issued before update or delete operations.

These tasks are all being performed for the application by Natural.

It is important to note that Natural is performing these tasks based on the information available in the application program. If, for example, a `READ` or `FIND` statement in a program is lacking essential segment search information, Natural selects a PCB, builds an SSA and issues a certain DL/I call based on this lacking information.

The Natural programmers use the same Natural statements to manipulate data in DL/I as they would for VSAM, Adabas or DB2.

Natural accesses DL/I segments based on the Natural DDM which is being referenced. Since the data access is always for one specific segment type (the one defined by the DDM), Natural neither issues path calls nor unqualified calls; that is, calls where the segment name is not specified.

> **Notes:**

1. Due to the structure of the Natural programming language, application control over DL/I call command codes is not available.

2. The `LOG`, `STAT` and `GSCD` call functions are not supported for the IMS TM environment.

Natural Statements with DL/I

This section mainly consists of information also contained in the *Natural Statements* documentation, where each Natural statement is described in detail, including notes on DL/I usage where applicable. Summarized below are the particular points a programmer has to bear in mind when using Natural statements with DL/I.

Any Natural statement not mentioned in this section can be used without restrictions with DL/I.

- BACKOUT TRANSACTION
- DELETE
- DISPLAY
- END TRANSACTION
- FIND
- GET TRANSACTION DATA
- READ
- RELEASE
- STORE
- UPDATE
- WRITE
- Statements not Available for DL/I

BACKOUT TRANSACTION

The Natural statement `BACKOUT TRANSACTION` is used to back out all database updates performed during the current logical transaction.

How the statement is translated and which command is actually issued depends on the TP-monitor environment:

- Under CICS, the `BACKOUT TRANSACTION` statement is translated into an `EXEC CICS ROLLBACK` command. However, in pseudo-conversational mode (`PSEUDO=ON`), only changes made to the database since the last terminal I/O are undone. This is due to CICS-specific transaction processing.

- In batch mode and under IMS TM, Natural for DL/I issues `ROLB` calls without checking the `CMPAT` setting in the corresponding NSB. However, under IMS TM, only changes made to the database since the last terminal I/O are undone. This is due to IMS TM-specific transaction processing.

Because PSB scheduling is terminated by a Syncpoint/checkpoint request, Natural saves the PCB position before executing the `BACKOUT TRANSACTION` statement. Before the next command execution, Natural reschedules the PSB and tries to set the PCB position as it was before the backout.

> **Note:** The PCB position might be shifted forward if any pointed segment had been deleted in the time period between the `BACKOUT TRANSACTION` and the following statement.

DELETE

The Natural statement `DELETE` is used to delete a segment from a DL/I database, which also deletes all descendants of the segment.

DISPLAY

The DL/I AIX fields can be displayed with the Natural statement `DISPLAY` only if a PCB is used with the AIX specified in the parameter `PROCSEQ`. If not, an error message is returned by Natural for DL/I at runtime.

END TRANSACTION

The Natural statement `END TRANSACTION` indicates the end of a logical transaction and releases all DL/I data locked during the transaction. All data modifications are committed and made permanent.

How the statement is translated and which command is actually issued depends on the TP-monitor environment:

- Under CICS, the `END TRANSACTION` statement is translated into an `EXEC CICS SYNCPOINT` command.

- In batch mode and non-message-driven BMP environments, Natural for DL/I issues `CHKP` calls without checking the `CMPAT` setting in the corresponding NSB.

- In MPP and message-driven BMP environments, the `END TRANSACTION` statement is not translated into a `CHKP` call, but is ignored, because `CHKP` calls imply GU calls. As Natural is a conversational transaction, you must reply to the terminal before requesting the next message (that is, before issuing the next `GU` call). An implicit end-of-transaction is issued after each terminal I/O.

Because PSB scheduling is terminated by a `SYNCPOINT`/`CHECKPOINT` request, Natural saves the PCB position before executing the `END TRANSACTION` statement. Before the next command execution, Natural reschedules the PSB and tries to set the PCB position as it was before the `END TRANSACTION` statement.

> **Note:** The PCB position might be shifted forward if any pointed segment had been deleted in the time period between the `END TRANSACTION` and the following command.

With batch-oriented BMP regions, user data can be checkpointed on the IMS system log data sets. This is done by supplying an 8-byte checkpoint ID as the first operand in the `END TRANSACTION` statement, and by specifying the areas to be checkpointed as additional operands.

If the database specified with the Natural profile parameter `ETDB` is not the same as the database affected by the transaction, the first operand of the `END TRANSACTION` statement will be used as checkpoint ID for the ETDB database, while `NATDLICK` will be used as checkpoint ID for the other database *not* specified with the `ETDB` parameter.

The total area to be checkpointed must not exceed 1992 bytes; see also *Symbolic Checkpoint/Restart Functions*.

FIND

With DL/I, the Natural `FIND` statement is typically used when a specific search criterion is known and specific segments are to be retrieved. This issues a DL/I `GET UNIQUE` call. However, if the `FIND` statement specifies a lower level segment and is within an active `READ` or `FIND` loop for an ancestor segment, it generally results in a DL/I `GET NEXT WITHIN PARENT` call.

The `FIND` statement initiates loop processing, which is active until all segment occurrences which match the search criterion have been read.

When accessing a field starting after the last byte of the given segment occurrence, the storage copy of this field is filled according to its format (numeric, blank, etc.).

`FIND FIRST`, `FIND NUMBER` and `FIND UNIQUE` are not permitted. The `PASSWORD`, `CIPHER`, `COUPLED` and `RETAIN` clauses are not permitted either.

In the `WITH` clause, you can only use descriptors that are defined as key fields in DL/I and marked with "D" in the DDM.

When connecting search criteria, the following has to be observed:

```
[NOT]  { basic-search-criterion }  [ { OR  }  search-expression ]  ...
       { (search-expression)    }    { NOT }
```

Connecting search criteria for segment type A results in multiple qualification statements within one DL/I segment search argument (SSA). Connecting search criteria for segment types A and B results in multiple SSAs. Therefore, the Boolean operator OR cannot be used to combine search criteria for different segment types.

GET TRANSACTION DATA

The Natural statement `GET TRANSACTION DATA` retrieves checkpoint data saved by an `END TRANSACTION` statement. The first parameter of this statement must be an 8-byte checkpoint ID. The remaining operands must be listed in the same sequence, length and format as in the corresponding `END TRANSACTION` statement; see also *Symbolic Checkpoint/Restart Functions*.

READ

The Natural statement `READ` should be used to process a set of segment occurrences in sequential order and usually results in a DL/I `GET NEXT` call.

When the `READ` statement is used, segments are retrieved based on the sequence field of the root segment or based on a secondary index field. Since the `READ` statement initiates sequential access of the database, it is important to understand that the `EQUAL TO` clause means the same thing as the `STARTING FROM` clause; it initiates a sequential read loop beginning with the key value specified.

The `READ` statement initiates loop processing. A loop is active until all segment occurrences which match the search criterion have been read.

The `PASSWORD` and `CIPHER` clauses are not permitted.

`IN PHYSICAL SEQUENCE` is used to read records in the order in which they are physically stored in a database. The physical sequence is the default sequence.

> **Note:** This is only valid when using Natural with HDAM databases.

`BY ISN` is not valid when using Natural with DL/I.

For Natural, the descriptor used must be either the sequence field of the root segment or a secondary index field. If a secondary index field is specified, it must also be specified in the `PROCSEQ` parameter of a PCB. Natural uses this PCB and the corresponding hierarchical structure to process the database.

RELEASE

The Natural statement `RELEASE` is not applicable for DL/I usage, since it releases sets of records retained by a `FIND` statement that contained a `RETAIN` clause, which is not valid when using Natural with DL/I.

STORE

The Natural statement `STORE` can be used to add a segment occurrence.

If the segment occurrence is defined with a primary key, a value for the primary key field must be provided.

In the case of a GSAM database, records must be added at the end of the database (due to GSAM restrictions).

The `USING/GIVING NUMBER` clause is not valid when using Natural with DL/I.

If the `SET/WITH` clause is used, the following applies with Natural for DL/I:

■ Values must be provided for the segment sequence field and for all sequence fields of the ancestors.

■ Only I/O (sensitive) fields can be provided.

■ A segment of variable length is stored with the minimum length necessary to contain all fields as specified with the STORE statement. The segment length will never be less than the minimum size specified in the SEGM macro of the DBD.

■ If a multiple-value field or a periodic group is defined as variable in length, at the end of the segment only the occurrences as specified in the STORE statement are written to the segment and define the segment length.

UPDATE

The Natural statement UPDATE can be used to update a segment in a DL/I database. The segment length is increased (if necessary) to accommodate all fields specified with the UPDATE statement. If a multiple-value field or a periodic group is defined as variable in length, only the occurrences as specified in the UPDATE statement are written to the segment.

The DL/I AIX field name cannot be used in an UPDATE statement. AIX fields, however, can be updated by referring to the source field which comprises the AIX field.

DL/I sequence fields cannot be updated because of DL/I restrictions.

If the SET/WITH clause is used, only I/O (sensitive) fields can be provided. A segment sequence field cannot be updated (DELETE and STORE must be used instead).

Due to GSAM restrictions, the UPDATE statement cannot be used for GSAM databases.

WRITE

With the Natural statement WRITE, the DL/I AIX fields can be displayed only if a PCB is used with the AIX specified in the parameter PROCSEQ. If not, an error message is returned by Natural for DL/I at runtime.

Statements not Available for DL/I

The following Natural statements are not available for DL/I users:

■ GET

■ GET SAME

■ HISTOGRAM

■ PASSW

■ RELEASE

Natural System Variables with DL/I

With DL/I, the following restrictions apply to the following Natural system variables:

*ISN

As there is no DL/I equivalent to Adabas internal sequence numbers (ISNs), the system variable `*ISN` is not available with Natural for DL/I.

*NUMBER

With Natural for DL/I, the Natural system variable `*NUMBER` does not contain the number of segment occurrences found. It contains `0` if no segment occurrence satisfies the search criterion and a value of `8,388,607=X'7FFFFF'` if at least one segment occurrence satisfies the search criterion.

39 Problem Determination Guide

The items listed below are cross-referenced by Natural for DL/I error messages. They are supplied to advise Natural programmers, DL/I database administrators and system support personnel of actions required to correct a given problem.

Item Number	Corresponding Action

1	**Activate Natural Trace Facility for DL/I**
	
	Note: The Natural trace facility for DL/I is available in all Natural for DL/I environments.
	
	To activate the Natural trace facility for DL/I (Dynamic Trace Activation):
	
	■ Execute the command `NDLTRACE` in library `SYSDDM` as follows:
	
	`NDLTRACE ON `*`parm1 parm2 parm3`*
	
	Permitted values for trace parameters are either `CMD`, `SER` and `ROU` (according to the specifications in the given error message) or `ALL` to trace all events of Natural for DL/I.
	
	To activate the Natural trace facility for DL/I (Initial Trace Activation):
	
	1. Code the `TRACE` parameter in the `NDLPARM` module according to the specifications in the given error message.
	
	  Or:
	
	  Specify `TRACE=ALL` to trace all events of Natural for DL/I.
	2. Assemble the `NDLPARM` module.
	3. Link-edit the load module that contains Natural for DL/I.
	
	To create and display the Natural trace for DL/I:
	
	1. Start the Natural session with `DSIZE=64` (or smaller).

Item Number	Corresponding Action
	This is required because the trace data is written into the `DSIZE` buffer. 2. Activate the trace facility (see above) and specify the following commands: `TEST DBLOG D` Start `DBLOG` for DL/I. ... Reproduce your problem here. `TEST DBLOG D` Display the data logged.
2	**Obtain the Program Listing**
3	**Obtain the View Listing**
4	**Obtain the DBD Macros**
5	**Obtain the PSB Macros**
6	**Obtain the NDB Description Printout** To obtain the printout, execute the Natural module `NDLBLOCK` in the library `SYSDDM` with the following parameters: ■ block type (3 bytes alphanumeric) = `NDB` ■ block name (8 bytes alphanumeric) = *`dbd-name`*
7	**Obtain the NSB Description Printout** To obtain the printout, execute the Natural module `NDLBLOCK` in the library `SYSDDM` with the following parameters: ■ block type (3 bytes alphanumeric) = NSB ■ block name (8 bytes alphanumeric) = *`psb-name`*
8	**Obtain the UDF Description Printout** To obtain the printout, execute the Natural module `NDLBLOCK` in the library `SYSDDM` with the following parameters: ■ block type (3 bytes alphanumeric) = `UDF` ■ block name (8 bytes alphanumeric) = *`db-id**file-number`* (that is, 3 digits for the database ID, a literal separator "**" and 3 digits for the *`file-number`*)
9	**Obtain a DUMP**
10	**Obtain the NDLPARM Listing**
11	**Obtain the NATDBD Procedure Output**
12	**Obtain the NATPSB Procedure Output**

40 Performance Considerations

This section lists some special considerations which may help you increase the performance of your Natural for DL/I environment.

Parameters

Set the `DLISIZE` parameter to `0` if no DL/I database is to be accessed.

Do not modify `NDLPARM` parameters, unless requested by a corresponding Natural for DL/I error message. Unused buffers are compressed by the Natural compression algorithm.

DBID

Use the same DBID for all segment types (DDMs) of a given NDB, because an `OPEN` command is generated for each DBID.

Global and Local Data Areas

Keep global and local data areas as small as possible, because the format buffer contains *all* fields of the global and local data areas, not only those which are referenced by a Natural I/O statement.

FIND Statements

If the sequence field is unique, use a `FIND (1)` statement instead of a `FIND` statement to prevent an unnecessary second DL/I call.

Direct Access to Lower Levels

Access segments on lower levels directly (by using the field sequence of the parent); that is, access ancestor segments only if their contents are required by the application program.

In such cases, UDFs of ancestor segments as well as DL/I fields of ancestor segments which are not sequence fields are not available to the application program.

DBLOG Utility

Use the Natural utility DBLOG (TEST DBLOG D) to tune your application; see *Logging Database Calls (DBLOG)* in the Natural *Utilities* documentation.

41 DL/I Services

When you invoke **DL/I Services** from the SYSDDM main menu, the **DL/I Services Main Menu** is displayed which offers you the following functions:

- **NDB Maintenance**
  An NDB is a DL/I DBD (database description) which is defined to Natural.

- **NSB Maintenance**
  An NSB is a DL/I PSB (program specification block) which is defined to Natural.

NDB Maintenance

This section covers the following topics:

- Menu and Functions
- Select an NDB from a List
- Select an NDB Segment from a List
- Edit an NDB Segment Description
- Generate DDM from Segment Description

Menu and Functions

When you select **NDB Maintenance** on the **DL/I Services Main Menu**, the NDB Maintenance menu is displayed:

```
 14:37:12                     **** DL/I Services ****                  2006-05-25
                               - NDB Maintenance -

                  Code Functions
                  ---- ------------------------------------
                    S  Select an NDB from a List
                    P  Purge an NDB
                    L  Select an NDB Segment from a List
                    E  Edit an NDB Segment Description
                    G  Generate DDM from Segment Description
                    ?  Help
                    .  Back
                    M  End
                  ---- ------------------------------------
            Enter Code: ?
              NDB Name:
          Segment Name:

 ENTER  PF1  PF2  PF3  PF4  PF5  PF6  PF7  PF8  PF9  PF10  PF11  PF12
        Help      Back                                          End
```

The individual NDB maintenance functions are listed below:

Function	Explanation
Select an NDB from a List	List the NDBs which are defined on the Natural system file. You can then select NDBs from this list by entering the following function codes: P to purge an NDB, L to list the segments of an NDB. For details, see *Select an NDB from a List*.
Purge an NDB	Purge an NDB and its related segment descriptions from the Natural system file. The name of the NDB to be purged must be specified. Before this function is executed you are prompted to confirm the purge request. For details, see *Select an NDB from a List*.
Select an NDB Segment from a List	List the segments of the specified NDB. You can then select segments from this list for further processing. For details, see *Select an NDB from a List*.
Edit an NDB Segment Description	Edit a segment description. The segment name and its corresponding NDB name are required when invoking this function. A database ID (DBID) and file number (FNR) must have been assigned to the segment description (function code A on the **Segment List** display) before it can be edited. For details, see *Edit an NDB Segment Description*.
Generate DDM from Segment Description	Generate a DDM from a segment description. The DDM definition is a Natural DDM of the segment. Prior to execution of this function, a DBID and FNR must have been assigned to the segment (function code A on the **Segment List** display). For details, see *Generate DDM from Segment Description*.

Select an NDB from a List

When you select an NDB from a list, a list containing all NDBs defined on the Natural system file is displayed. In addition to the NDB name the following is displayed:

L/P	Indicates if `ACCESS=LOGICAL` or not.
length	Length of the NDB.
NoSGMS	Number of the segment types in the NDB.
ACCESS	The access specification taken from the DBD.

From the list, you can select NDBs for further processing by entering the following function codes in the **Func** column next to the NDB names:

Code	Function
P	**Purge NDB**
	This function is identical to the **Purge NDB** function available on the **NDB Maintenance** menu. It deletes an NDB and its related segment descriptions from the Natural system file. Before the function is executed you are prompted to confirm the purge request.
L	**List NDB Segments**
	This function is identical to the **Select NDB Segment from a List** function available on the **NDB Maintenance** menu. It lists the segments of the selected NDB.
	For details, see *Select an NDB Segment from a List*.

Select an NDB Segment from a List

When you select an NDB segment from a list, a list containing all segments of the specified NDB is displayed. If you do not know the NDB name, use the **Select an NDB from a List** function.

```
 10:50:48                    **** DL/I SERVICES ****                    2006-05-25
                              - Segment  List -
 DBD Name = ED00DBD
   Func   Level  Segment     DBID   FNR    Seg-Lgh       UDF-Lgh    Response
 -------------------------- Top of Data --------------------------------
     _        1    COURSE     246    _10    75-80          100
     _        2    PREREQ     246    _11    36-36           40
     _        2    OFFERING   246    _12    41-41           40
     _        3    TEACHER    246    _13    24-24           60
     _        3    STUDENT    246    _14    40-40           40
     _                              ___    ___
     _                              ___    ___
     _                              ___    ___
     _                              ___    ___
     _                              ___    ___
 --------------------------- Bottom ------------------------------------
     Code .. _ ( ? Help   . Back   M End )
   Func = E  Edit Segment Description    A  Assign DBID and FNR
          F  Free DBID and FNR           ' ' Change DBID and FNR
          G  Generate DDM                N  Take New Copy of UDF

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
 Exec  Help       Exit                                                    Canc
```

Next to each segment you can enter one of the function codes listed below. You can mark several segments at the same time with a function code. If you do not enter any code, the list is scrolled forward until the bottom of the list is reached.

You can enter one of the following codes next to a segment on the segment list to perform one of the following functions:

Code	Function
A	**Assign DBID/FNR** Assign a DBID and a FNR to the selected segment. The DBID is a number in the range from 1 to 254. It must be contained in the database ID list (`NTDB` macro) of the Natural parameter module. All Natural DDMs which refer to a DL/I segment must have a DBID belonging to this range. If a DBID has not been entered, a default value is assigned, which is the entry with the lowest value in the DBID list specified in the Natural parameter module. For a given NDB, all segments should be assigned the same DBID. Otherwise, Natural assumes different databases and generates an `OPEN` command (which, however, is ignored by the DL/I call handler). The FNR is a number in the range from 1 to 254. The FNR must be specified; no default value is assumed by Natural. The segments of a logical NDB must have file numbers different from those assigned to the segments of the physical NDB. The DBID/FNR combination must be unique on the Natural system file. It is used by Natural to uniquely determine the NDB and the segment within the NDB.
E	**Edit Segment Description** Edit the description of a segment within a given NDB. The function is the same as the **Edit an NDB Segment Description** function which you can invoke from the **NDB Maintenance** menu. Before you can edit a segment description, a DBID and FNR must have been assigned to the segment (see above). For details, see *Edit an NDB Segment Description*.
F	**Free DBID/FNR** Release the DBID and FNR which have previously been assigned to the segment. Once a DBID and FNR have been released, they are available for assignment to another segment.
G	**Generate DDM** Generate a DDM definition from a segment description. The function is the same as the **Generate DDM from Segment Description** function which you can invoke from the **NDB Maintenance** menu. Before you can generate a DDM from a segment description, a DBID and FNR must have been assigned to the segment (see above). The generated DDM is a Natural view of the segment. Once it has been generated, the DDM can be modified and cataloged. For details, see *Generate DDM from Segment Description*.
N	**Take New Copy of UDF** Refresh the user-defined fields (UDFs) of a segment when the UDFs of the source segment have been changed. This applies to UDFs of segments belonging to a logical NDB and to UDFs of logical virtual children. Before you can execute this function, a DBID and FNR must have been assigned to the segment **(see above)**.

Code	Function
blank	**Change DBID/FNR**
	Change a previously assigned DBID and/or FNR.
	For changing a DBID or FNR, the same rules concerning DBID and FNR specification apply as for assigning a DBID/FNR **(see above)**.

Edit an NDB Segment Description

Additional segment fields, so-called user-defined fields (UDFs), can be defined.

This function is invoked either by entering function code E, an NDB name and a segment name on the **NDB Maintenance** menu, or by selecting the segment from the **Segment List** (by marking it with function code E). A DBID and a FNR must have been assigned to a segment description (function code A on the **Segment List** display) before it can be edited.

```
EDIT command:       DBD   ED00DBD   SEGMENT STUDENT   SEGLGH 40-40
  ALL   LEV  SN        FIELD NAME      START  DLI MAXOCC FOR LGH   V
        -----------------------------------------------------------
        1   PM  EMPNO               00001  SQU          A    6
        1   PN  NAME                00007  SRC          A   33
        1   PO  GRADE               00040  SRC          A    1
        1   AA  BIRTHDATE           00025               A
        2   AB  DATE-DD                                 N    2
        2   AC  DATE-MM                                 N    2
        2   AD  DATE-YY                                 N    2
        1   AE  BIRTHPLACE                              A   10
        1   AF  STUDENT-NAME        PN                  A   18
```

The following information is displayed on the status line at the top of the screen:

DBD	Name of the DBD which contains the edited segment.
SEGMENT	Name of the edited segment.
SEGLGH	Minimum and maximum length of the edited segment, separated by a hyphen.

DL/I fields and user-defined fields are displayed as shown above. You can add, delete or modify UDFs. DL/I fields, however, can neither be added nor deleted. If the specification TYPE=P is included in the FIELD statement of the DL/I DBD, the format of the field can be changed from P (decimal packed unsigned) to S (decimal packed signed) on the edit segment description screen. FOR (format) is the only attribute of a DL/I field you can modify. In particular, it is not possible to change the name of a DL/I field, because it is used by Natural to build the segment search arguments (SSA). If the name of a DL/I field is to be changed, the field can be redefined as an UDF.

Edit commands are available to copy or delete single lines or to insert a group of empty lines. In addition, commands for scrolling forward or backward are provided. For details you can enter a question mark in the "command" field to display the corresponding help information.

After modification of segment field attributes you can save the description by entering SAVE in the command field.

The following field definition attributes are displayed and can be modified for user-defined fields:

Attribute	Description
LEV	Level number used to define a group of fields.
SN	Short name of the field as used internally by Natural.
FIELD NAME	Name of the field as used in the application programs.
START	Start position of the field in the segment.
DLI	Type of the DL/I field, as follows: SIX secondary index field SQU sequence field (unique) SQM sequence field (multiple) SRC search field
MAXOCC	Maximum number of occurrences of a multiple field or periodic group.
FOR	Format of the field.
LGH	Length of the field.
V	Variable field length indicator.

Each user-defined field can be defined as follows:

Field Type	Description	
Elementary Field	A field that contains only one value in a single segment. Example: Personnel number	
Multiple Field	A field that can contain more than one value in a single segment. Reference to a particular value of a multiple field can be made by appending a one to three-digit subscript (value 1 - 191) to the field name. Example: Languages - English, German, Italian	

Field Type	Description
Group	A series of one or more adjacent fields that can be referenced with a single name (the group name). You can also refer to a single field of a group by specifying its name. Example: 01 Address Group field 02 City Elem. field 02 Street " " 02 Number " "
Periodic Group	A group which is repeated in multiple adjacent occurrences in a single segment. For a periodic group it is possible to refer to a range of occurrences (or a field within a periodic group) by specifying the first and the last occurrence number to be referenced (connected by a hyphen (-)) after the name and in ascending order. Multiple-value fields or periodic groups are not allowed within a periodic group. Example: Several addresses

Since DL/I fields cannot be modified as described above (with the exception of FORMAT), they cannot be directly defined as a group. To define a DL/I field as a group, it is necessary to redefine it as a user-defined field which then can be redefined as a group. In a DDM, these user-defined fields must not be specified as descriptor fields. When a DDM is generated, the UDFs are marked as non-descriptor fields.

Example - Redefinition of a DL/I Sequence Field as a Group:

The description of the segment STUDENT within the DBD named ED00DBD is used as shown in the **Segment List** screen above:

```
 LEV  SN    FIELD NAME        START  DLI   NOCC  FOR LGH  V
 ----------------------------------------------------------
  1  PM   EMPNO            00001  SQU         A    6
  .
  .
  .
```

If the DL/I sequence field PM is to be "structured", it must be redefined as a user-defined field (AAAAA in the figure below). This UDF can then be structured as required.

```
 LEV  SN    FIELD NAME        START  DLI   NOCC   FOR LGH  V
 -----------------------------------------------------------------
  1   PM   EMPNO             00001  SQU          A    6
  .
  .
  .
  1   AA   AAAAA             PM
  2   AB   BBBBB                                 A    3
  2   AC   CCCCC                                 A    3
  .
  .
  .
```

The group field `AAAAA` has no format/length (`FOR`/`LGH`) specified. The length of a group is set equal to the sum of all fields belonging to the group.

UDF Parameters

For each user-defined field on the above screen, parameters can be specified as listed and described in the following table. The total length of all DL/I fields and user-defined fields must not exceed the segment length.

When attributes of a UDF are modified and an old copy of this UDF is contained in the shared UDF buffer pool, the old copy is marked "invalid". If the UDF is referred to again by a Natural program, the modified UDF is read from the Natural system file. Therefore, it is not necessary to restart the Natural session if a UDF has been modified. However, this applies only to physical UDFs; that is, to UDFs of a physical NDB. If a physical UDF is modified and a logical NDB refers to the appropriate segment type, the logical UDF is not marked "invalid" in the buffer pool. To invalidate a logical UDF it is necessary to restart the TP monitor or to execute function `N` (**Take New Copy of UDF**) of the **Segment List** screen on the appropriate segments in the logical NDB.

Field	Description
`LEV` (level number)	A one-byte value used to define a group. A field is a group only if the subsequent field has a higher level number. The field immediately after the last group element must have a lower level number. A group can be defined within another group. The level number of the first user-defined field must be 1.
`SN` (short name)	The name used internally by Natural to identify the field. It must be two bytes in length, the first character must be alphabetic in the range from A to G (E is not permitted). The second character can be alphanumeric (that is, up to 216 UDF fields can be defined). If the segment is a logical child, the first character must be alphabetic in the range from H to M. Short names must be unique among a segment type.
`FIELD NAME`	External field name, up to 19 bytes long.
`START`	The start position of the field in the segment. The position can be specified as absolute by giving a three-digit number or it can be specified as relative, by giving the short name of a previously defined field which is being redefined.

Field	Description
	It is important to specify the start position for the first user-defined field; otherwise, a default of 1 is used, which may cause overlapping with previous DL/I fields. The default for all other user-defined fields is the position immediately after the previous field. The redefinition of fields is possible only for fields which have the same level number. When the level is higher than 1 (that is, for a field inside a group), only the last field can be redefined with the same level number. An absolute position must not be specified for a field within a group.
MAXOCC	The maximum number of occurrences of a multiple-value field or periodic group in a segment.
FOR (format)	Standard field formats are: A Alphanumeric B Binary F Fixed Point P Packed decimal unsigned; that is, the zone half-byte of the last byte is X'F'. S Packed decimal signed; that is, the zone half-byte of the last byte is X'C' (positive) or X'B' (negative). N Unpacked
LGH (length)	Field length is a three-digit number; it must not exceed the maximum length permitted. These are as follows: 253 bytes for alphanumeric fields (A), 126 bytes for binary fields (B), 4 bytes for fixed point (F), 14 bytes for packed decimal unsigned (P), 14 bytes for packed decimal signed (S), 27 bytes for unpacked decimal (N) In addition, the length specified must not exceed the segment length. Length must not be specified for a group. The length of packed fields is the field length in bytes.
V (variable)	Depending on its value, V or blank, this parameter indicates whether a field has a variable length. Fields can be specified as variable only if the segment is a segment of variable length. Only one field can be defined as variable within a given segment description. An elementary field can be specified as variable in length only if it is the last field in the segment. A multiple field or a periodic group can be specified as variable in length regardless of its position in the segment. When applied to a multiple field or a periodic group, a setting of VARIABLE means that the number of occurrences is not known at definition time; therefore, MAXOCC should be specified using the maximum expected value.

Generate DDM from Segment Description

This function is invoked either by using the `G` function code of the **NDB Maintenance** menu - then an NDB name and a segment name must be specified -, or by selecting the segment from the **Segment List**, by marking it with function code `G`.

A DBID and a FNR must have been assigned to a segment description (function code `A` on the **Segment List** display) before a DDM can be generated.

The DDM is generated from a segment description and represents a Natural view of the segment. It must be generated and cataloged before the corresponding segment can be referenced by a Natural program. After generation, default options for field headers or edit masks (decimal positions) can be modified in the DDM. See *Catalog DDM* and *Edit DDM* in the Natural *Utilities* documentation for corresponding information.

It should be noted, however, that default options for field headers or edit masks (decimal positions) are stored with the DDM and not with the NDB or UDF. The data in the NDB or UDF reflects what is allowed by the DL/I FIELD macro in which the length can be specified only in bytes (decimals are not allowed). Consequently, when regenerating the DDM, prior modifications in the DDM must be applied again by the user.

In DL/I a program must be able to reference search fields, sequence fields and secondary index fields of ancestor segments in order to build a certain search criterion; therefore, DDMs for DL/I segments can also include fields which are not part of the actual physical segment.

To satisfy the requirements for DL/I processing, a DDM must contain all the fields which can be referenced. Therefore, the generated DDM can contain the following fields:

- DL/I sequence fields, search fields and secondary index fields of the current (physical) segment. These fields have been defined in the `DBDGEN` source for this segment. When the DDM is generated, information on these fields is obtained from the NDB control block for this segment. DL/I sequence fields and secondary index fields are marked as descriptor (`D`), search fields are marked as non-descriptor (`N`). All of these fields can be used to qualify search requests.

- DL/I sequence fields and secondary index fields of all the ancestor segments. These fields have been defined in the `DBDGEN` source for the ancestor segments. When the DDM is generated, information on these fields is obtained from the NDB control blocks for the ancestor segments. These fields are marked as descriptor (`D`). They can be used to qualify search requests.

- DL/I search fields of all the ancestor segments. These fields have also been defined in the `DBDGEN` source for the ancestor segments. When the DDM is generated, information on these fields is also obtained from the NDB control blocks for the ancestor segments. However, these fields are marked as superdescriptor (`S`). They can be used to qualify search requests.

- Fields of the current segment defined by the user (UDFs). When the DDM is generated, information on these fields is obtained from the UDF control blocks. These fields cannot be used to qualify search requests.

Fields of format S in the segment description (see *UDF Parameters*) generate format P in the DDM.

The following tables summarize how the various types of fields can be processed using Natural I/O statements. They illustrate which fields can be used to qualify search requests, and which fields can be used with the Natural statements DISPLAY, UPDATE or STORE. In addition, the tables indicate whether the field in the generated DDM is marked as descriptor, superdescriptor or non-descriptor.

Current Segment

Type of field	FIND/READ	DISPLAY	UPDATE	STORE	Marked
DL/I sequence	yes	yes	no	yes	D
DL/I search	yes	yes	yes	yes	D
DL/I SIX	yes	yes	no	no	D
UDF	no	yes	yes	yes	blank

Ancestor Segment

Type of field	FIND/READ	DISPLAY	UPDATE	STORE	Marked
DL/I sequence	yes	yes	no	yes	D
DL/I search	yes	no	no	no	S
DL/I SIX	yes	yes	no	no	D
UDF	no	no	no	no	blank

> **Notes:**

1. Using the Natural statement DISPLAY, the DL/I SIX fields can be displayed only if a PCB is used with this SIX specified in the PROCSEQ parameter. If not, an error message is returned by Natural at runtime.

2. The DL/I SIX field name cannot be used in an UPDATE or STORE statement. SIX fields, however, can be updated/stored by referring to the source fields which comprise the SIX.

3. The READ statement returns records in ascending sequence. The possible sequences for DL/I segments are root sequence or the sequence of any secondary index.

As mentioned above, the generated DDM contains all fields of the current segment and all DL/I fields of the ancestor segment(s), marked either as D or S. The UDFs of the ancestor segments are not included in the generated DDM because a DDM refers only to one segment.

The generated external name of the DDM is equal to the segment name prefixed by the DBD name.

Example:

Name of DBD:	ED00DBD
Name of segment:	STUDENT
Name of generated DDM:	ED00DBD-STUDENT

The generated external name of DL/I fields is equal to the name specified in the DL/I FIELD macro during the DL/I DBDGEN procedure.

The generated external name of DL/I fields of ancestor segments is equal to the field name suffixed by the segment name.

Example:

Name of DL/I field:	LOCATION
Name of ancestor segment:	OFFERING
Name of generated field:	LOCATION-OFFERING

The generated external name of the UDFs is equal to the name specified by the user at definition time.

NSB Maintenance

When you select **NSB Maintenance** on the **DL/I Services Main Menu**, the **NSB Maintenance** menu is displayed.

From this menu, you can select the following NSB maintenance functions:

Function	Explanation
Select an NSB from a List	List the DL/I PSBs defined on the Natural system file. You can select NSBs from this list by entering either function code: P Purge an NSB. L List all PCBs and SENSEGs of an NSB.
Purge an NSB	Delete an NSB and its related PCB descriptions from the Natural system file. The name of the NSB to be deleted must be specified. Before this function is executed, you are prompted to confirm the deletion.
List PCBs and SENSEGs of an NSB	For any NSB specified, this function lists the PCBs and their sensitive segments. If an indexed database exists, its name is displayed under the header **PROCSEQ**.

DL/I Services

Select an NSB from a List

```
 10:44:50                    **** DL/I SERVICES ****                   2006-05-25
                               - NSB List -

             Func     NSB Name  CMPAT  Length  NoPCBs   Response
             ----------------- Top of Data -------------------
              _       DFSIVP6   YES     140      3
              _       PBNDL01   NO      160      3
              _       PBNDL02   YES     160      1
              _       PBNDL03   YES     160      3
              _       PBNDL04   YES     160      1
              _       PBNDL05   NO       80      1
              _       PBNDL97   YES     160      3
              _       PBNDL98   YES     200      5
              _       PBNDL99   NO      200      5
              _       PBPQA01   YES      60      5
              _       PBSUP06   NO      440      5
             ------------------- - More - --------------------
         Code .. _ ( ? Help, . Back, M End )


         Func .. P (Purge NSB) L (List PCBs and SENSEGs)

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
 Exec  Help      Exit                                                      Canc
```

List PCBs and SENSECs of an NSB

```
 10:46:57                    **** DL/I SERVICES ****                  2006-05-25
                                - PCB List -
   NSB Name: PBNDL01  (CMPAT=NO ,Length=00160)

        Number of PCB's  NDB Name    Level    SENSEG         PROCSEQ
        ------------------------- Top of Data -------------------------
              3          ED00DBD
                                       1       COURSE
                                       2       PREREQ
                                       2       OFFERING
                                       3       TEACHER
                                       3       STUDENT




        ----------------------- Bottom ----------------------------

        Code .. _ ( ? Help   . Back   M End )

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
  Exec  Help      Exit                                                    Canc
```

```
 10:49:10                    **** DL/I SERVICES ****                     2006-05-25
                               - NDB List -

         Func      NDB Name  L/P  Length  NoSGMs  Access   Response
         --------------------- Top of Data --------------------
           _        CCCBTD00   P    460       6
           _        DNDL01     P    540       5
           _        DNDL02     P    620      10
           _        DNDL03     L    820      10
           _        DNDL04     P     60       1    GSAM
           _        DPQA04     P    480       5
           _        DSUP02     L   1720      15
           _        DSUP05     P    380       5
           _        DSUP09     P    340       2
           _        DSUP10     L    880      10
           _        DUSA01     P    320       5    HDAM
         --------------------- - More - ---------------------
      Code .. _ ( ? Help,  . Back,  M End )



      Func: P (Purge NDB) L (List NDB Segments)

 Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
 Exec  Help       Exit                                                    Canc
```

Index

D

database management system interfaces, ix