Natural

Datenbankmanagementsystem-Schnittstellen

Version 9.1.2

April 2023

ADABAS & NATURAL

Dieses Dokument gilt für Natural ab Version 9.1.2.

Hierin enthaltene Beschreibungen unterliegen Änderungen und Ergänzungen, die in nachfolgenden Release Notes oder Neuausgaben bekanntgegeben werden.

Dokument-ID: NATMF-NATDBMS-912-20230424DE

Inhaltsverzeichnis

Datenbankmanagementsystem-Schnittstellen

Diese Dokumentation gibt einen Überblick über die Natural-Datenbankmanagementsystem-Schnittstellen und eine kurze Zusammenfassung ihrer Funktionen.

Die folgenden Themen werden behandelt:

Natural for Db2	Mit der Datenbankmanagementsystem-Schnittstelle zu Db2 können Natural-Anwender auf Daten in einer Db2-Datenbank zugreifen. Natural for Db2 wird unter den TP-Monitoren Com-plete, CICS, IMS TM, im Batch-Modus und unter TSO unterstützt.
Natural for VSAM	Mit der Datenbankmanagementsystem-Schnittstelle zu VSAM können Natural-Anwender auf in VSAM-Dateien gespeicherte Daten zugreifen.
Natural for DL/I	Die Dokumentation ist nur in Englisch verfügbar. 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. **Natural for DL/I will not be supported from version 9.2.2 onwards.**

Anmerkung: Siehe auch Kapitel *Datenbankzugriffe* im *Leitfaden zur Programmierung* . Dort wird auch beschrieben, wie Sie Daten in einer **Adabas**-Datenbank aufrufen und aktualisieren.

1 Über diese Dokumentation

Dokumentationskonventionen

Konvention	Beschreibung
Fettschrift	>Kennzeichnet Elemente auf einem Bildschirm.
`Nichtproportionale Schrift`	Kennzeichnet Namen und Orte von Diensten im Format *`Ordner.Unterordner.Dienst`*, Programmierschnittstellen (APIs), Namen von Klassen, Methoden und Properties in Java.
`Kursivschrift`	Kennzeichnet: Variablen, für die Sie situations- oder umgebungsspezifische Werte angeben müssen. Neue Begriffe, wenn sie erstmals im Text auftreten. Verweise auf andere Dokumentationsquellen.
`Nichtproportionale Schrift`	Kennzeichnet: Text, den Sie eingeben müssen. Meldungen, die vom System angezeigt werden. Programmcode.
{ }	Zeigt eine Reihe von Auswahlmöglichkeiten an, von denen Sie eine auswählen müssen. Geben Sie nur die innerhalb der geschweiften Klammern vorhandenen Informationen ein. Geben Sie nicht die Klammersymbole { } ein.
|	Trennt zwei sich gegenseitig ausschließende Auswahlmöglichkeiten in einer Syntaxzeile voneinander ab. Geben Sie eine der Auswahlmöglichkeiten ein. Geben Sie nicht das Symbol | ein.
[ ]	Zeigt eine oder mehrere Optionen an. Geben Sie nur die innerhalb der eckigen Klammern vorhandenen Informationen ein. Geben Sie nicht die Klammersymbole [ ] ein.
...	Zeigt an, dass Sie mehrere Auswahlmöglichkeiten desselben Typs eingeben können. Geben Sie nur die Informationen ein. Geben Sie nicht die drei Auslassungspunkte (...) ein.

Online-Informationen und Support

Produktdokumentation

Sie finden die Produktdokumentation auf unserer Dokumentationswebsite unter **https://documentation.softwareag.com**.

Zusätzlich können Sie auch über **https://www.softwareag.cloud** auf die Dokumentation für die Cloud-Produkte zugreifen. Navigieren Sie zum gewünschten Produkt und gehen Sie dann, je nach Produkt, zu „Developer Center“, „User Center“ oder „Documentation“.

Produktschulungen

Sie finden hilfreiches Produktschulungsmaterial auf unserem Lernportal unter **https://knowledge.softwareag.com**.

Tech Community

Auf der Website unserer Tech Community unter **https://techcommunity.softwareag.com** können Sie mit Experten der Software AG zusammenarbeiten. Von hier aus können Sie zum Beispiel:

- Unsere umfangreiche Wissensdatenbank durchsuchen.
- In unseren Diskussionsforen Fragen stellen und Antworten finden.
- Die neuesten Nachrichten und Ankündigungen der Software AG lesen.
- Unsere Communities erkunden.
- Unsere öffentlichen Repositories auf GitHub and Docker unter **https://github.com/softwareag** und **https://hub.docker.com/publishers/softwareag>** besuchen und weitere Ressourcen der Software AG entdecken.

Produktsupport

Support für die Produkte der Software AG steht lizenzierten Kunden über unser Empower-Portal unter **https://empower.softwareag.com>** zur Verfügung. Für viele Dienstleistungen auf diesem Portal benötigen Sie ein Konto. Wenn Sie noch keines haben, dann können Sie es unter **https://empower.softwareag.com/register>** beantragen. Sobald Sie ein Konto haben, können Sie zum Beispiel:

- Produkte, Aktualisierungen und Programmkorrekturen herunterladen.
- Das Knowledge Center nach technischen Informationen und Tipps durchsuchen.
- Frühwarnungen und kritische Alarme abonnieren.
- Supportfälle öffnen und aktualisieren.
- Anfragen für neue Produktmerkmale einreichen.

Datenschutz

Die Produkte der Software AG stellen Funktionen zur Verarbeitung von personenbezogenen Daten gemäß der Datenschutz-Grundverordnung (DSGVO) der Europäischen Union zur Verfügung. Gegebenenfalls sind in der betreffenden Systemverwaltungsdokumentation entsprechende Schritte dokumentiert.

I Natural for Db2

Diese Dokumentation beschreibt die Funktionalität und den Einsatz von Natural for Db2, einer Natural-Schnittstelle, die den Zugriff auf Daten in einer Db2-Datenbank ermöglicht.

Allgemeine Informationen	Verwendungszweck, umgebungsspezifische Aspekte, Integration mit Predict, dem Data Dictionary der Software AG, Inkompatibilitäten und Einschränkungen, Fehlermeldungen im Zusammenhang mit Db2 und die in dieser Dokumentation verwendeten Begriffe.
Zugriff auf eine Db2-Tabelle	Wie Sie den Zugriff auf eine Db2-Tabelle mit einem Natural-Programm ermöglichen.
Natural Tools for Db2 benutzen	Wie Sie die Natural Tools for Db2 aufrufen, um Db2-spezifische Objekte und SQL-Anweisungen zu verwalten.
Anwendungspläne verwalten	Wie Sie die Db2-Anwendungspläne online verwalten.
Katalog verwalten	Wie Sie den Db2-Katalog verwalten.
Interaktives SQL	Wie Sie SQL-Statements, die nicht eingebettet sind, verarbeiten.
Systemtabellen abrufen	Wie Sie Db2-Objekte und -Benutzerberechtigungen anzeigen oder drucken.
Environment Setting-Funktionalität benutzen	Wie Sie SQL-Statements ausführen und die Werte spezieller Register anzeigen können.
Explain PLAN_TABLE-Funktionalität benutzen	Wie Sie Ihre `PLAN_TABLE` interpretieren.
File Server-Statistiken	Wie Sie Statistiken über die Generierung und Nutzung des Dateiservers anzeigen.
Db2-Kommandos aus Natural absetzen	Wie Sie Db2-Statements aus Natural heraus absetzen.
Natural-Systemkommandos für Db2 benutzen	Wie Sie Natural-Systemkommandos benutzen, die in die Natural Tools for Db2 integriert wurden.
Natural-Datendefinitionsmodule (DDMs) generieren	Generation of Natural data definition modules (DDMs) using the *SQL Services* function of the Natural `SYSDDM` utility.
Dynamische und statische SQL-Unterstützung	Interne Behandlung dynamischer Statements, Erstellung und Ausführung statischer Database Request Modules (DBRM),

	gemischter dynamischer/statischer Modus und Anwendungsplanwechsel in den verschiedenen unterstützten Umgebungen.
Natural-Statements und Systemvariablen benutzen	Behandelt besondere Aspekte von Natural-DML-Statements, Natural-SQL-Statements und Natural-Systemvariablen bei Db2. Darüber hinaus wird die erweiterte Fehlerbehandlung von Natural for Db2 besprochen.
Verarbeitung von Natural Stored Procedures und UDFs	Beschreibt die Verarbeitung von Natural Stored Procedures und Natural User-Defined Functions (UDFs).
Interface-Subprogramme	Beschreibt mehrere Natural- und Nicht-Natural-Subprogramme, die für verschiedene Zwecke verwendet werden können.
Natural File Server für Db2	Beschreibt, wie ein Natural File Server für Db2 in den verschiedenen unterstützten Umgebungen verwendet wird.

Verwandte Dokumentation

Hinweise zur Installation und eine Beschreibung der Natural for Db2-Parametermodule siehe unter *Installing Natural for DB2* in der *Installation for z/OS*-Dokumentation.

Informationen zu verschiedenen Aspekten des Zugriffs auf Daten in einer Datenbank mit Natural siehe unter *Datenbankzugriffe* im Natural-*Leitfaden zur Programmierung*.

Informationen zur Protokollierung von SQL-Statements, die in einem Natural-Programm enthalten sind, siehe *DBLOG Trace-Bildschirm für SQL-Statements* in der *Debugger und Dienstprogramme (Utilities)*-Dokumentation.

2 Allgemeine Informationen

Verwendungszweck

Natural for DB2 ist eine Natural-Schnittstelle für den Zugriff auf Daten in einer DB2-Datenbank.

Im Großen und Ganzen gibt es keinen Unterschied zwischen der Verwendung von Natural mit Db2 und der Verwendung von Natural mit Adabas, VSAM oder DL/I. Die Natural-Schnittstelle zu Db2 ermöglicht Natural-Programmen den Zugriff auf Db2-Daten mit denselben nativen Natural-Anweisungen zur Datenmanipulation (DML), die auch für Adabas, VSAM und DL/I verfügbar sind. Daher können Programme, die für Db2-Tabellen geschrieben wurden, auch für den Zugriff auf Adabas-, VSAM- oder DL/I-Datenbanken verwendet werden. Darüber hinaus sind Natural SQL DML-Anweisungen verfügbar.

Alle Operationen, die eine Interaktion mit Db2 erfordern, werden von Natural for DB2 ausgeführt.

Umgebungsspezifische Aspekte

Natural for Db2 wird in den folgenden Umgebungen unterstützt:

- Natural for DB2 unter Com-plete
- Natural for DB2 unter CICS
- Natural for DB2 unter IMS TM
- Natural for DB2 unter TSO
- Natural for DB2 mit CAF
- Natural for Db2 mit Db2 DL/I Batch-Unterstützung

Natural for DB2 unter Com-plete

Db2 wird von Com-plete unterstützt. Programme, die unter Com-plete laufen, können auf Db2-Datenbanken über die Db2 Call Attachment Facility (CAF) zugreifen. Zusammen mit der Com-plete-Schnittstelle zu Db2 ermöglicht diese Einrichtung einen vollständig konversationellen Zugriff auf Db2-Tabellen.

Wenn der während des Installationsprozesses erstellte Db2-Plan nicht in Ihrer Db2-SERVER-Parameterliste für Com-plete angegeben ist, müssen Sie NATPLAN vor dem ersten SQL-Aufruf explizit aufrufen, um diesen Plan zuzuweisen.

Natural for DB2 unter CICS

CICS/Db2 Attachment Facility

Unter CICS verwendet Natural die CICS/Db2 Attachment Facility für den Zugriff auf Db2. Stellen Sie daher sicher, dass dieses Attachment gestartet ist. Ist dies nicht der Fall, wird die Natural-Sitzung mit dem CICS-Abend-Code `AEY9` vorzeitig beendet, was zur Natural-Fehlermeldung NAT0954 führt, wenn der Natural-Profilparameter `DU` auf `OFF` gesetzt ist.

CICS-Db2-Plan-Auswahl

Wenn die Natural-CICS-Transaktions-ID keinem Db2-Plan in der RCT durch `DB2ENTRY`- und `DB2TRAN`-Definitionen zugeordnet ist, müssen Sie vor dem ersten SQL-Aufruf explizit `NATPLAN` ausführen, um den gewünschten Db2-Plan anzugeben und `NDBUEXT` als dynamischen Planauswahl-Exit (Attribut `PLANExit`) zu definieren. Die eigentliche Planzuweisung wird durch den dynamischen Planauswahl-Exit durchgeführt.

Pseudo-Conversational Mode unter CICS

Unter CICS läuft ein Natural-Programm in der Regel im Pseudo-Conversational Mode (Natural-Profilparameter `PSEUDO` auf `ON` gesetzt; dies ist der Standardwert). In diesem Fall wird am Ende der CICS-Transaktion die Db2-Transaktion festgeschrieben und alle offenen Db2-Cursor werden implizit geschlossen. Es gibt normalerweise keine Möglichkeit, offene Natural-`FIND`/`SELECT`-Datenbankzugriffsschleifen nach der Terminal-E/A wieder aufzunehmen.

Um das Problem zu umgehen, dass CICS eine pseudo-konversationelle Transaktion während der Schleifenverarbeitung beendet und Db2 dadurch alle Cursor schließt und alle Selektionsergebnisse verliert, benutzt Natural for Db2 den File Server zur Unterstützung der Natural-Transaktionslogik. Wenn Sie im Pseudo-Conversational Mode unter CICS arbeiten wollen, geben Sie im Natural-Profilparameter `DB2` den Schlüsselwort-Subparameter `FSERV=ON` an und stellen Sie eine File Server-Datei in der CICS-Region bereit.

Pseudo-Conversational Mode unter CICS

Wenn Sie keine File Server-Datei in der CICS-Region bereitstellen und im Natural-Profilparameter `DB2` der Schlüsselwort-Subparameter `CONVERS` auf `ON` gesetzt ist, schaltet Natural for Db2 immer dann in den Conversational Mode, wenn während einer offenen Datenbankschleife eine Terminal-E/A stattfindet. Das bedeutet, dass die CICS-Transaktion über Terminal-Ein-/Ausgaben erzeugt wird, solange es offene Datenbankschleifen gibt. Dies kann zu Deadlocks bei Db2 führen, da Db2-Ressourcen über Terminal-Ein-/Ausgaben zugeordnet werden.

Conversational Mode 2 unter CICS

Um Anwendungen zu unterstützen, die das implizite Commit bei CICS Terminal-Ein-/Ausgaben nicht einsetzen und stattdessen ein explizites `ROLLBACK` oder `COMMIT` codieren, um ihre Datenbanktransaktion zu beenden, wurde ein Conversational Mode 2 eingeführt.

Conversational Mode 2 bedeutet, dass eine Db2-Update-Transaktion über Terminal-Ein-/Ausgaben erzeugt wird, bis ein explizites `COMMIT` oder `ROLLBACK` ausgegeben wird.

Der Conversational Mode 2 kann angefordert werden , indem im Natural-Profilparameter `DB2` der Schlüsselwort-Subparameter `CONVRS2=ON` gesetzt wird, oder er kann durch den Aufruf des `CALLNAT`-Programms `NDBCONV` dynamisch gesetzt oder zurückgesetzt werden.

Vorsicht: Diese Art von Anwendungen neigen dazu, CICS- und Db2-Ressourcen zu binden, da die Ressourcen über Terminal-Ein-/Ausgaben nicht freigegeben werden!

File Server unter CICS

Die Verwendung des File Servers wird gesteuert durch den Schlüsselwort-Subparameter `FSERV` im Makro `NTDB2`.

In einer CICS-Umgebung ist der File Server ein optionales Feature, um die Probleme bei der Umstellung auf konversationelle Verarbeitung zu lösen. Vor einer Bildschirm-Ein-/Ausgabe erkennt Natural, ob offene Cursors vorhanden sind, und speichert die in diesen Cursors enthaltenen Daten in den File Server. Mit dem File Server können Datenbankschleifen über Terminal-Ein-/Ausgaben hinweg fortgesetzt werden, aber Datenbankänderungen, die vor einer Terminal-Ein-/Ausgabe vorgenommen wurden, können nicht mehr rückgängig gemacht werden.

Ausführliche Informationen zum File Server siehe unter *Natural File Server für DB2*.

Natural for DB2 unter IMS TM

Unter IMS TM verwendet Natural die IMS Db2 Attachment Facility für den Zugriff auf Db2. Stellen Sie daher sicher, dass dieses Attachment gestartet ist.

In IMS TM-Transaktionsverarbeitungsumgebungen schließt Db2 alle Cursors und verliert dadurch alle Selektionsergebnisse, wenn das Programm zum Terminal zurückkehrt, um eine Antwortnachricht zu senden. Dieser Betriebsmodus unterscheidet sich von der Art und Weise, wie Db2 im Conversational Mode unter CICS oder in TSO-Umgebungen arbeitet, wo Cursor über die Terminal-Kommunikation hinweg geöffnet bleiben können und daher ausgewählte Zeilen für längere Zeit beibehalten werden können.

File Server unter IMS TM MPP

Die Verwendung des File Servers wird gesteuert durch den Schlüsselwort-Subparameter `FSERV` im Makro `NTDB2`.

Der File Server wird benötigt, um das Natural for Db2-Cursor-Management zu unterstützen, während IMS TM nach jeder Terminal-Ein-/Ausgabe-Operation ein implizites End-of-Transaction an Db2 ausgibt. Mit dem File Server können Datenbankschleifen über Terminal-Ein-/Ausgaben hinweg fortgesetzt werden, aber Datenbankänderungen, die vor einer Terminal-Ein-/Ausgabe vorgenommen wurden, können nicht mehr rückgängig gemacht werden.

Ausführliche Informationen zum File Server siehe unter *Natural File Server für DB2*.

Natural for DB2 unter TSO

Natural for Db2 kann unter TSO ausgeführt werden, ohne dass Änderungen an der Natural/TSO-Schnittstelle erforderlich sind.

Neben z/OS Batch kann die Batch-Umgebung für Natural auch der TSO Background sein, der das TSO-Terminal-Monitorprogramm durch eine `EXEC PGM=IKJEFT01`-Anweisung in einem JCL-Stream aufruft.

Sowohl TSO-Online- als auch Batch-Programme können entweder unter der Kontrolle des DSN-Kommandos oder unter Verwendung der Call Attachment Facility (CAF) ausgeführt werden. Die CAF-Schnittstelle ist erforderlich, wenn Plan-Switching verwendet werden soll.

File Server unter TSO

Die Verwendung des File Servers wird gesteuert durch den Schlüsselwort-Subparameter `FSERV` im Makro `NTDB2`.

In einer TSO-Umgebung ist der File Server ein optionales Feature, durch das während des Entwicklungsstatus eine zukünftige CICS- oder IMS TM-Produktionsumgebung emuliert werden kann.

Bei jeder Terminal-Ein-/Ausgabe gibt Natural ein `COMMIT WORK`-Kommando aus, um CICS- oder IMS TM-Syncpoints zu simulieren. Daher können Datenbankänderungen, die vor einer Terminal-Ein-/Ausgabe vorgenommen wurden, nicht mehr rückgängig gemacht werden.

Ausführliche Informationen zum File Server siehe unter *Natural File Server für DB2*.

Natural for DB2 mit CAF

Wenn Sie Natural for Db2 unter TSO oder im Batch-Modus ausführen und die CAF-Schnittstelle verwenden, müssen Sie `NATPLAN` vor dem ersten SQL-Aufruf explizit aufrufen, um den erforderlichen Db2-Plan zuzuordnen.

`NATPLAN` kann bearbeitet werden, um die entsprechende Db2-Subsystem-ID anzugeben.

Natural for Db2 mit Db2 DL/I Batch-Unterstützung

Wenn Sie in derselben Natural-Sitzung im Batch-Modus (nicht BMP) auf Db2 und DL/I zugreifen möchten, können Sie die Db2 DL/I Batch-Unterstützung nutzen, mit der Sie die Wiederherstellung von Db2- und DL/I-Datenbanksystemen koordinieren können.

Wenn Sie diese Möglichkeit nutzen wollen, müssen Sie die Prozedur `DLIBATCH` ausführen, um das Modul `DSNMTV01` als Anwendungsprogramm zu starten. `DSNMTV01` wiederum führt den Natural-Batch-Nukleus aus, der mit der Db2-Schnittstelle `DFSLI000` verlinkt sein muss.

Wenn Ihr PSB mit `CMPAT=YES` generiert wird, werden alle Syncpoints ausgeführt, und Sie müssen ein `END TRANSACTION`-Statement absetzen, bevor Sie Ihre Natural-Sitzung beenden. Andernfalls gehen alle Datenbankänderungen verloren, weil Natural am Ende der Sitzung implizit ein `BACKOUT TRANSACTION`-Statement absetzt.

Wenn Ihr PSB mit `CMPAT=NO` generiert wird, werden alle Syncpoints ignoriert.

Integration mit Predict

Predict, das offene, operationale Data Dictionary der Software AG für die Entwicklung mit der 4GL-Sprache Natural, ist ein zentrales Repository für Anwendungsmetadaten und bietet Dokumentations- und Cross-Reference-Funktionen. Mit Predict können Sie automatisch Code aus Definitionen generieren und so die Produktivität bei Entwicklung und Wartung steigern.

Da Db2 von Predict unterstützt wird, ist ein direkter Zugriff auf den Db2-Katalog über Predict möglich. Informationen aus dem Db2-Katalog können in das Predict-Datendiktionär übertragen werden, um sie in Datendefinitionen für andere Umgebungen zu integrieren.

Db2-Datenbanken, -Tabellen und -Sichten (Views) können eingebunden und verglichen werden, neue Db2-Tabellen und -Sichten können generiert werden, und Natural-DDMs können erzeugt und verglichen werden. Alle Db2-spezifischen Datentypen und die referenzielle Integrität von Db2 werden unterstützt. Details finden Sie in der entsprechenden Predict-Dokumentation.

Darüber hinaus unterstützen die aktiven Predict-Referenzen statisches SQL für Db2, wie unter *WITH XREF Option* in *Programme für die statische Ausführung vorbereiten* beschrieben.

Integration mit Natural Security

Wenn das Programm in einer Umgebung ausgeführt wird, die von Natural Security kontrolliert wird, kann die Verwendung bestimmter Funktionen von Natural for Db2 durch den Security-Administrator eingeschränkt werden, z. B:

- **Natural Tools for DB2**

  Zugriff auf die Natural System-Library `SYSDB2`

  Einzelne Funktionen
- **Statisches SQL**

  Die statische Generierung kann wie folgt unterbunden werden:

  - Einschränkung des Zugriffs auf die Natural-System-Library `SYSDB2`,
  - Nichtzulassung des Moduls CMD,
  - Beschränkung des Zugriffs auf die Libraries, die die entsprechenden Natural-Objekte enthalten,
  - Nichtzulassung eines der Natural-Systemkommandos `CATALOG` oder `STOW` für eine Library, die relevante Natural-Objekte enthält.

  Wenn eine Library in Natural Security definiert ist und die Datenbankkennung (DBID) und Dateinummer (FNR) dieser Library von den Standardangaben abweichen, schaltet das statische Generierungsverfahren automatisch auf die in Natural Security definierten DBID- und FNR-Angaben um.

Weitere Informationen erhalten Sie von Ihrem Security-Administrator.

Inkompatibilitäten und Einschränkungen

In diesem Abschnitt werden die bekannten Inkompatibilitäten und Einschränkungen gegenüber Db2 aufgeführt, die auftreten können, wenn Natural for Db2 für den Zugriff auf Daten aus Db2 verwendet wird.

- **Datentyp DECIMAL oder NUMERIC**
  Die meisten SQL-Datenbanksysteme unterstützen gepackte Dezimalzahlen mit einer maximalen Genauigkeit von 31 Ziffern und einer Skalierung (Bruchteil der Zahl) von bis zu 31 Ziffern. Die Skalierung muss positiv sein und darf nicht größer als die Genauigkeit sein. Natural erlaubt eine Genauigkeit und Skalierung von bis zu 29 Ziffern.

Meldungen mit Bezug zu Db2

Die Nummernbereiche von Natural-Systemmeldungen im Zusammenhang mit Db2 sind 3275 - 3286, 3700-3749 und 7386-7395.

Eine Liste der Fehlermeldungen, die bei der statischen Generierung ausgegeben werden können, finden Sie unter *Static Generation Messages and Codes Issued under NDB* in der Natural-*Messages and Codes*-Dokumentation.

In dieser Dokumentation verwendete Begriffe

Begriff	Erläuterung
Db2 (vormals: DB2)	Db2 bezieht sich auf IBMs Db2 UDB für z/OS.
DBRM	Datenbankabfrage-Modul.
DDM	Natural-Datendefinitionsmodul.
DML	Datenmanipulationssprache (Natural).
File Server	In diesem Dokument bezieht sich der Begriff „File Server“ auf den Natural File Server für Db2.
NDB	Dies ist der Produktcode von Natural for Db2. In dieser Dokumentation wird der Produktcode häufig als Präfix in den Namen von Dateien (Datasets), Modulen usw. verwendet.

3 Zugriff auf eine Db2-Tabelle

Um den Zugriff auf eine Db2-Tabelle mit einem Natural-Programm zu ermöglichen:

1 Benutzen Sie die **Natural Tools for Db2**, um eine Db2-Tabelle zu definieren. Siehe *Natural Tools for Db2 benutzen*.

2 Verwenden Sie Predict oder die Funktion **SQL Services** des Natural-Dienstprogramms SYSDDM, um ein Natural-Datendefinitionsmodul (DDM) für die definierte Db2-Tabelle zu erstellen.

3 Sobald Sie ein DDM für eine Db2-Tabelle definiert haben, können Sie mit einem Natural-Programm auf die in dieser Tabelle gespeicherten Daten zugreifen.

Natural for Db2 setzt die Statements eines Natural-Programms in SQL-Statements um.

Natural sorgt automatisch für die Vorbereitung und Ausführung der einzelnen Statements. Im dynamischen Modus wird ein Statement nur einmal vorbereitet (wenn möglich) und kann dann mehrmals ausgeführt werden. Zu diesem Zweck führt Natural intern eine Tabelle mit allen vorbereiteten Statements (siehe Statement Table in *Interne Behandlung dynamischer Statements*).

Für die Entwicklung von Natural-Anwendungen, die auf Db2-Tabellen zugreifen, kann nahezu die gesamte Bandbreite der Möglichkeiten der Programmiersprache Natural genutzt werden. Für eine Reihe von Natural DML-Statements gibt es jedoch gewisse Einschränkungen und Unterschiede, was die Verwendung mit Db2 betrifft, siehe *Verwendung von nativen Natural DML-Statements*. In der *Statements*-Dokumentation finden Sie Hinweise zur Verwendung von Natural mit Db2 bei den Beschreibungen der betreffenden Natural DML-Statements.

Da es keine Db2-Entsprechung zu den internen Sequenznummern (ISNs) von Adabas gibt, sind alle Natural-Funktionen, die ISNs verwenden, beim Zugriff auf Db2-Tabellen mit Natural nicht verfügbar.

Für SQL-Datenbanken bietet Natural zusätzlich zu den nativen Natural DML-Statements spezielle Natural-SQL-Statements. Siehe ***Verwendung von Natural-SQL-Statements***. Sie sind in der *Statements*-Dokumentation aufgeführt und erläutert.

4 Natural Tools for Db2 benutzen

Dieser Abschnitt beschreibt, wie Sie die Natural Tools for Db2 aufrufen und Db2-spezifische Objekte und SQL-Statements verwalten. Außerdem enthält dieser Abschnitt Informationen zu globalen PF-Tasten-Einstellungen und globalen Verwaltungskommandos in Natural Tools for Db2.

Anmerkungen:

1. Siehe auch *Special Requirements for Natural Tools for DB2* in *Installing Natural for DB2 on z/OS*.
2. Wenn Sie bei der Installation von Natural for Db2 eine neue SYSDB2-Library erstellt haben, stellen Sie sicher, dass diese alle Predict-Schnittstellenprogramme enthält, die zur Ausführung der **Natural Tools for Db2** erforderlich sind. Diese Programme werden zum Zeitpunkt der Predict-Installation in SYSDB2 geladen (siehe die entsprechende Predict-Dokumentation).

Natural Tools for Db2 aufrufen

Um die Natural Tools for Db2 aufzurufen:

- Geben Sie das Natural-Systemkommando `SYSDB2` ein.

Das Hauptmenü (**Main Menu**) der **Natural Tools for Db2** wird angezeigt. Es bietet Ihnen die unten aufgeführten Funktionen.

```
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  ↩
```

Funktionen des Hauptmenüs

Funktion	Beschreibung
Application Plan Maintenance	Wie Sie Db2-Anwendungspläne online verwalten.
Catalog Maintenance	Wie Sie den Db2-Katalog verwalten.
Interactive SQL	Wie SQL-Statements verarbeitet werden, die nicht eingebettet sind.
Retrieval of System Tables	Wie Sie Db2-Objekte und Benutzerberechtigungen anzeigen/drucken.
Environment Setting	Wie Sie SQL-Statements ausführen und spezielle Registerwerte anzeigen.
Explain PLAN_TABLE	Wie Sie Ihre PLAN_TABLE interpretieren.
File Server Statistics	Wie Sie Statistiken über die Generierung und über die Nutzung des File Servers anzeigen.
DB2 Commands Execution	Wie Sie Db2-Kommandos von Natural aus absetzen.

Mit den Natural Tools for Db2 editieren

Der in den **Natural Tools for Db2** verfügbare Free-Mode-Editor setzt voraus, dass der **Software AG Editor** installiert ist. Die Zeilenkommandos und Hauptkommandos, die in den **Natural Tools for Db2** zur Verfügung stehen, sind eine Untermenge der im **Software AG Editor** verfügbaren Befehle.

Sowohl die Hauptkommandos als auch die Zeilenkommandos werden ausführlich in der Online-Hilfe von **Natural Tools for Db2** beschrieben, die Sie durch Drücken von PF1 (Help) aufrufen können. Weitere Einzelheiten finden Sie in der Dokumentation zum **Software AG Editor**.

Übersicht über die Hauptkommandos des Editors

Die Hauptkommandos werden in der Kommandozeile des Editorbildschirms eingegeben. Die wichtigsten Hauptkommandos sind:

Kommando	PF-Taste	Beschreibung
BOTTOM (++)		Sprung an das Ende der Daten.
CHANGE		Sucht nach einer bestimmten Zeichenfolge und ersetzt jede gefundene Zeichenfolge durch eine angegebene Zeichenfolge.
CLEAR		Löscht den Quellcodebereich des Editors.
DELETE		Löscht die Zeile(n), die eine bestimmte Zeichenfolge enthalten, entsprechend den angegebenen Auswahloperanden.
DOWN (+)	PF8	Blättert um den angegebenen Betrag nach unten.
FIND		Sucht eine durch Kommandooperanden angegebene Zeichenkette an der/den durch Auswahloperanden angegebenen Stelle(n).

Kommando	PF-Taste	Beschreibung
LEFT	PF10	Mit diesem Kommando können Sie in den Daten um den angegebenen Betrag nach Links blättern.
LIMIT *n*		Legt ein Limit für das FIND-Kommando fest. Es werden *n* Zeilen verarbeitet.
PRINT		Druckt die angezeigten Daten aus.
RESET		Setzt alle anstehenden Zeilenkommandos zurück.
RFIND	PF5	Wiederholt das letzte FIND-Kommando.
RIGHT	PF11	Mit diesem Kommando können Sie in den Daten um den angegebenen Betrag nach Rechts blättern.
TOP (--)		Sprung an den Anfang der Daten.
UP (-)	PF7	Mit diesem Kommando können Sie in den Daten um den angegebenen Betrag nach oben blättern.

Der Blätterbetrag für die Kommandos UP, DOWN, LEFT, and RIGHT wird im Feld SCROLL in der oberen rechten Ecke des Auflist-Bildschirms angegeben. Gültige Werte für den Blätterbetrag sind:

Wert	Erläuterung
CSR	Der Blätterbetrag wird durch die Cursorposition bestimmt.
DATA	Der Blätterbetrag entspricht der Seitengröße abzüglich einer Zeile.
HALF	Der Blätterbetrag entspricht der Hälfte der Seitengröße.
MAX	Der Blätterbetrag entspricht der Datenmenge nach unten/oben.
PAGE	Der Blätterbetrag ist gleich der Seitengröße.
n	Der Blätterbetrag entspricht *n* Zeilen.

Übersicht über die Zeilenkommandos des Editors

Zeilenkommandos werden in den Editor-Präfixbereich der entsprechenden Statement-Zeile eingegeben. Die wichtigsten Zeilenkommandos sind:

Kommando	Beschreibung
A	Fügt die zu verschiebende(n) oder zu kopierende(n) Zeile(n) hinter der aktuellen Zeile ein.
B	Fügt die zu verschiebende(n) oder zu kopierende(n) Zeile(n) vor der aktuellen Zeile ein.
C	Kopiert die aktuelle Zeile.
CC	Markiert den Anfang und das Ende eines Blocks von zu kopierenden Zeilen.
D	Löscht die aktuelle Zeile.
DD	Markiert den Anfang und das Ende eines zu löschenden Zeilenblocks.
I*nn*	Fügt *nn* neue Zeilen nach der aktuellen Zeile ein.
M	Verschiebt die aktuelle Zeile.
MM	Markiert den Anfang und das Ende eines zu verschiebenden Zeilenblocks.

Globale PF-Tastenbelegungen

Innerhalb der **Natural Tools for Db2** gelten die folgenden globalen PF-Tastenbelegungen:

Taste	Belegung	Beschreibung
PF1	Help (Hilfe)	Durch Drücken von PF1 rufen Sie die Online-Hilfe von einem beliebigen Bildschirm der Natural **Tools for Db2** aus auf.
PF3	Exit (Ende)	Wenn Sie PF3 drücken, gelangen Sie immer zum vorherigen Bildschirm oder zur vorherigen Funktion. Wenn Sie die Taste auf einem Editorbildschirm drücken, auf dem Änderungen vorgenommen wurden, wird das Fenster **Exit Function** angezeigt (wie unter *Exit-Funktion* beim Natural-Programm-Editor beschrieben). Wenn Sie PF3 im Hauptmenü (**Main Menu**) drücken, verlassen Sie die **Tools for Db2**.
PF12	Canc (Abbruch)	Wenn Sie PF12 drücken, kehren Sie immer zu dem Menü zurück, von dem aus Sie den aktuellen Bildschirm aufgerufen haben. Wenn Sie PF12 im Hauptmenü (**Main Menu**) drücken, verlassen Sie die **Tools for Db2**.

Globale Verwaltungskommandos

Innerhalb der Natural Tools for Db2 gelten die folgenden globalen Verwaltungskommandos:

Kommando	Beschreibung
COPY *name*	Kopiert das angegebene Member aus der aktuellen Library in den Editor, nach (A) oder vor (B) der aktuellen Zeile.
LIBRARY *name*	*name* legt den Namen der Natural Library als aktuelle Library fest.
LIST *name**	Listet alle Member der aktuellen Library auf, deren Namen mit *name* beginnen. Aus der Liste können Sie ein Member auswählen, indem Sie es mit S markieren.
PURGE *name*	Löscht das angegebene Member aus der aktuellen Library.
READ *name*	Liest das angegebene Member aus der aktuellen Library in den Editor ein. Der aktuelle Name wird auf *name* gesetzt.
SAVE [*name*]	Speichert den generierten Code als Member *name* in der aktuellen Library. Wenn kein Name angegeben wird, wird der aktuelle Name verwendet. Die aktuellen Library- und Member-Namen werden oberhalb der Kommandozeile angezeigt.

Member- und Library-Namen müssen den Natural-Namenskonventionen entsprechen. Siehe *Namenskonventionen für Objekte* und *Namenskonventionen für Libraries* in *Natural benutzen*. Member können JCL-Member, SQL-Member oder Output-Member sein.

5 Anwendungspläne verwalten

Allgemeines zur Funktion Application Plan Maintenance

Der Application Plan Maintenance-Teil der **Natural Tools for Db2** dient zur Generierung von JCL-Code,

- um Datenbankanforderungsmodule (DBRMs) aus Ihren Natural-Programmen zu erstellen,
- um Db2-Anwendungspläne und -Packages aus Ihrer Natural-Umgebung heraus zu verwalten.

Es stehen zwei Betriebsarten zur Verfügung: Fixed Mode und Free Mode.

Fixed Mode

Im Fixed Modus werden Sie durch Verwaltungsbildschirme mit Syntaxgraphen bei der Angabe der richtigen Kommandos unterstützt. Vollständige JCL-Members können anhand vordefinierter Jobprofile generiert werden. Sie geben einfach die erforderlichen Daten in Eingabemasken ein. Die Daten werden auf die Einhaltung der korrekten Syntax geprüft. Dann werden aus diesen Daten JCL-Members generiert. Die Members können direkt durch Drücken von PF4 (Submi) übergeben werden. Sie können aber auch in den Free Mode wechseln, indem Sie PF5 (Free) drücken.

Free Mode

Wenn Sie im Fixed Mode PF5 drücken, wird der Free-Mode-Editor aufgerufen, mit dem Sie den im Fixed Mode erzeugten JCL-Code ohne die syntaktischen Einschränkungen ändern können. Im Free Mode können Sie das JCL-Member, das sich gerade im Quellcodebereich befindet, durch Drücken von PF4 (wie im Fixed Mode) übermitteln.

Funktion Application Application Plan Maintenance aufrufen

Um die Funktion Application Plan Maintenance aufzurufen:

- Geben Sie im Hauptmenü **Natural Tools for DB2 Main Menu** den Funktionscode A ein.

  Das Menü **Application Plan Maintenance** wird angezeigt (Beispiel):

```
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
```

Die folgenden Funktionen sind verfügbar:

Code	Beschreibung
PP	Definiert Jobprofile für die DBRM-Erstellung und die Plan/Package-Verwaltung, siehe *Jobprofile vorbereiten*.
CD	Erzeugt JCL, um Datenbankanforderungsmodule zu erstellen.
BI	Erzeugt JCL, um einen Plan oder ein Package zu binden.
RB	Erzeugt JCL, um einen Plan oder ein Package neu zu binden.
FR	Erzeugt JCL, um einen Plan oder ein Package freizugeben.
LJ	Ruft den Free-Mode-Editor auf.
JO	Zeigt die Job-Ausgabe an. **Anmerkung:** Diese Funktion ist nur anwendbar, wenn der Entire System Server installiert ist.

Darüber hinaus stehen Parameter zur Verfügung, die je nach gewählter Funktion angegeben werden müssen:

Parameter	Beschreibung
Object	Gibt an, ob ein Plan (PLAN oder PL) oder ein Package (PACKAGE oder PK) gepflegt werden soll.
Library	Gibt den Namen einer Natural-Source-Library an. Alle vorhandenen Libraries außer denen, die mit SYS beginnen, können angegeben werden. Für die JCL-Pflege muss eine Library angegeben werden. Der Library-Name ist mit Ihrer Natural-Benutzerkennung voreingestellt.
JCL Member	Wenn ein gültiger Member-Name angegeben wird, wird das entsprechende JCL-Member angezeigt. Wird ein Wert gefolgt von einem Stern (*) angegeben, werden alle JCL-Members in der angegebenen Library, deren Namen mit diesem Wert beginnen, aufgelistet. Wenn Sie nur einen Stern (*) angeben, wird eine Auswahlliste mit allen JCL-Members in der angegebenen Library angezeigt. Wenn Sie das Feld JCL-Member leer lassen, wird der leere Editorbildschirm im Free-Mode angezeigt.
Node	Gibt die Nummer des Knotens an, der vom Entire System Server verwendet werden soll. Die Standardnummer ist 148 und kann überschrieben werden.

Kommandos und PF-Tastenbelegungen

Innerhalb der Verwaltungsbildschirme können im Fixed Mode verschiedene Fenster aufgerufen werden. Der Zugriff auf diese Fenster erfolgt über 1-Byte-Steuerfelder.

Um ein Fenster aufzurufen:

- Geben Sie S in das entsprechende Steuerfeld ein.

Zeigt das Kontrollfeld ein X an, so sind bereits Daten in das entsprechende Fenster eingegeben worden.

Darüber hinaus gelten die folgenden PF- Tastenbelegungen im Fixed Mode:

PF-Taste	Funktion
PF4	Generiert JCL-Code und übergeben ihn.
PF5	Generiert JCL-Code und wechselt in den Free Mode.
PF6	Blättert in einem Fenster nach oben.
PF7	Blättert in Fenstern rückwärts.
PF8	Blättert in Fenstern vorwärts.
PF9	Blättert zum unteren Rand eines Fensters.

PF-Taste	Funktion
PF10	Zeigt entweder den vorherigen Bildschirm an (<) oder zeigt ein **Entire System Server Logon**-Fenster an (Logn).
PF11	Zeigt den nächsten Bildschirm an.

Im Free Mode kann JCL-Code bearbeitet und übergeben werden. Die Bearbeitung von JCL-Code erfolgt über Editier- und Zeilenkommandos, siehe *Mit den Natural Tools for Db2 editieren*.

Generierter JCL-Code wird durch Drücken von PF4 übergeben.

JCL-Code kann nicht nur übergeben, sondern auch kopiert, aufgelistet, gelöscht, abgerufen oder in einer Natural-Library gespeichert werden. All dies geschieht über globale Verwaltungskommandos, siehe *Globale Verwaltungskommandos*.

Jobprofil erstellen

Wenn Sie JCL generieren wollen, um einen DBRM zu erstellen oder um einen Plan oder ein Package zu binden, freizugeben oder neu zu binden, müssen Sie einen Jobnamen, Jobkarten und den Namen eines Jobprofils angeben. Daher müssen Sie zunächst die Jobprofile erstellen. Wenn Ihre Jobprofile definiert sind, können Sie stets sofort die entsprechende Funktion auswählen, wenn Sie einen neuen DBRM erstellen oder einen Plan oder ein Package mit Hilfe Ihrer vordefinierten Jobprofile binden, freigeben oder neu binden möchten.

Um ein Jobprofil zu definieren:

1 Geben Sie im Hauptmenü **Natural Tools for DB2 Main Menu** den Funktionscode A ein.

Das Menü **Application Plan Maintenance** wird angezeigt.

2 Rufen Sie im Menü **Application Plan Maintenance** die Funktion **Prepare Job Profile** auf, indem Sie den Funktionscode PP eingeben.

Das Menü **Prepare Job Profile** wird angezeigt.

Menü Prepare Job Profile

```
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	Beschreibung
J	Definiert benutzerspezifische Standard-Jobkarten.
D	Definiert Jobprofile für die DBRM-Erstellungsfunktion.
P	Definiert Jobprofile für die Plan- oder Package-Verwaltungsfunktionen.

Zusätzlich gibt es den Parameter Profile, der nur für die Funktionscodes D und P relevant ist. Beim Funktionscode J entspricht Profile dem USER.

Parameter	Beschreibung
Profile	Gibt den Namen eines bereits vorhandenen Jobprofils an. Wird ein gültiger Profilname angegeben, so wird der Free-Mode-Editor mit dem angegebenen Jobprofil aufgerufen. Dort kann das Profil geändert und gespeichert werden. Wird ein Wert gefolgt von einem Stern (*) angegeben, werden alle vorhandenen Jobprofile aufgelistet, deren Name mit diesem Wert beginnt. Wird nur ein Stern (*) angegeben, wird eine Auswahlliste aller vorhandenen Jobprofile angezeigt. Wird das Feld leer gelassen, wird der entsprechende Fixed-Mode-Profil-Bildschirm aufgerufen, in dem ein neues Jobprofil erstellt werden kann. Um das neue Profil zu speichern, müssen Sie in den Free Mode wechseln.

Jobprofile können mit Hilfe von Verwaltungskommandos verwaltet (d.h. kopiert, aufgelistet, gelöscht, abgerufen oder in einer Natural Library gespeichert) werden, siehe *Globale Verwaltungskommandos*.

Anmerkung: Jobprofile werden in der Natural-Systemdatei FNAT gespeichert.

Standard-Jobkarten

Für alle von der Funktion **Application Plan Maintenance** generierten Jobs werden Jobkarten benötigt.

Mit der Funktion **Default Job Cards** können Sie für jeden Benutzer eine Standard-Jobkarte definieren. Standard-Jobkarten gelten für alle Funktionsbildschirme, auf denen Sie JCL generieren können, und können dort aufgerufen und geändert werden. Mit Stern-Notation (*) können Sie die gewünschte Jobkarte aus einer Liste auswählen.

Um eine Standard-Jobkarte zu definieren:

- Geben Sie im Menü **Prepare Job Profile** den Funktionscode `J` ein und drücken Sie Enter.

  Der Bildschirm **Default Job Cards** wird angezeigt.

```
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
```

  Auf diesem Bildschirm können Sie Ihre benutzerspezifischen Jobkarten erstellen und speichern. Dazu können Sie auch (direkt oder aus einer Liste) eine bereits vorhandene Standard-Jobkarte lesen und ändern. Bereits bestehende Jobkarten können auch gelöscht werden.

Anmerkung: Alle anderen Funktionsbildschirme, die zur Angabe von Jobs verwendet werden, enthalten die gleichen beiden Felder - **Job Name** und **Job Cards** - wie der Bildschirm **Default Job Cards**. Daher ist es möglich, die Standard-Jobkarten auch in jedem dieser Bildschirme zu überschreiben.

Um den Jobnamen zu ändern:

- Geben Sie den neuen Job-Namen in das Feld **Job Name** ein und drücken Sie Enter.

Um die Jobkarten zu ändern:

- Geben Sie im Feld **Default Job Cards** ein `S` ein und drücken Sie Enter.

  Es wird ein Fenster angezeigt, in dem Sie alle Jobkarten ändern können.

Profile für Create DBRM Job definieren

Mit der Funktion **Profile for Create DBRM Job** können Sie Profile für die **Create DBRMs**-Funktionen definieren. Jobprofile für die DBRM-Erstellung bestehen aus JCL, die den folgenden vordefinierten Satz von Ersetzungsparametern enthält:

Parameter	Beschreibung
`@JOBCARDS`	Wird durch die auf dem Bildschirm **Create DBRMs** eingegebenen Jobkarten ersetzt (bis zu fünf Zeilen). Sie können die Jobkarten auch im Profil kodieren und den Modifikator für die Jobkarten weglassen.
`@COMMAND`	Wird durch die Zeichenfolge `CREATE DBRM` ersetzt.
`@DBRMNAME`	Wird durch den Namen des DBRM ersetzt, der bis zu acht Zeichen lang sein kann.
`@CREATE-DBRM`	Wird durch die Kommandoeingabe für den statischen Generierungsschritt ersetzt. Dieser Parameter muss *nach* der `//CMSYNIN`-Karte stehen und den Assembler-Namenskonventionen entsprechen.
`@COMMAN2`	Wird durch die Zeichenkette `MODIFY` ersetzt.
`@MODIFY`	Wird durch die Kommandoeingabe für den statischen Modifikationsschritt ersetzt.
`@XR-START` `@XR-END`	Beide markieren die JCL, die die Natural Assembler XREF-Daten enthalten soll. Wenn keine XREF-Option angegeben ist, wird die JCL wieder gelöscht.

Um ein Jobprofil für die DBRM-Erstellung zu ändern oder umzubenennen:

1. Rufen Sie im Menü **Prepare Job Profile** die Funktion **Profile for Create DBRM Job** auf, indem Sie den Funktionscode `D` eingeben.
2. Geben Sie im Feld **Profile** einen gültigen Profilnamen an und drücken Sie Enter.

Der Free-Mode-Editor mit dem angegebenen Profil wird aufgerufen. Dort können Sie das angezeigte Profil ändern, speichern und umbenennen.

Um ein Jobprofil für die DBRM-Erstellung zu erstellen:

1 Geben Sie im Menü **Prepare Job Profile** den Funktionscode `D` ein.

2 Lassen Sie das Feld **Profile** leer, und drücken Sie Enter.

Der Bildschirm **Profile for Create DBRM Job** wird aufgerufen. Sie können dort ein neues Profil erstellen.

```
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
```

Um das neu erstellte Jobprofil zu speichern:

- Wechseln Sie in den Free Mode, indem Sie PF5 drücken.

Profile für DSN Jobs definieren

Mit der Funktion **Profile for DSN Jobs** können Sie Profile für die Funktionen **Bind**, **Rebind** und **Free** definieren. Für jede der drei Funktionen können die gleichen Profile verwendet werden.

Profile für DSN-Jobs bestehen aus JCL, die den folgenden vordefinierten Satz an Ersetzungsparametern enthält:

Parameter	Beschreibung
`@JOBCARDS`	Wird durch die aktuellen Jobkarten ersetzt. Sie können die Jobkarten auch im Profil codieren und den Jobkarten-Modifikator weglassen.
`@DSNCMD`	Wird durch die Kommandoeingabe für die Funktionen **Bind**, **Rebind** und **Free** ersetzt.
`@PLANNAME`	Wird bei der **Bind**-Funktion durch den Namen des Plans oder des Package ersetzt. Für die Funktionen **Rebind** und **Free** wird er auf Leerzeichen gesetzt.
`@COMMAND`	Wird durch die Zeichenfolge `BIND`, `REBIND` bzw. `FREE` ersetzt.

Um ein Profil für DSN-Jobs zu ändern oder umzubenennen:

1 Geben Sie im Menü **Prepare Job Profile** den Funktionscode `P` ein.

2 Geben Sie im Feld **Profile** einen gültigen Profilnamen ein, und drücken Sie Enter.

Der Free-Mode-Editor mit dem angegebenen Profil wird aufgerufen. Dort können Sie das angezeigte Profil ändern, speichern und umbenennen.

Um ein neues Profil für DSN-Jobs zu erstellen:

1 Geben Sie im Menü **Prepare Job Profile** den Funktionscode `P` ein.

2 Lassen Sie das Feld **Profile** leer, und drücken Sie Enter.

Der Bildschirm **Profile for DSN Jobs** wird aufgerufen. Dort können Sie ein neues Profil für DSN-Jobs erstellen.

```
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
```

Um das neu erstellte Jobprofil zu speichern:

- Wechseln Sie in den Free Mode, indem Sie PF5 drücken.

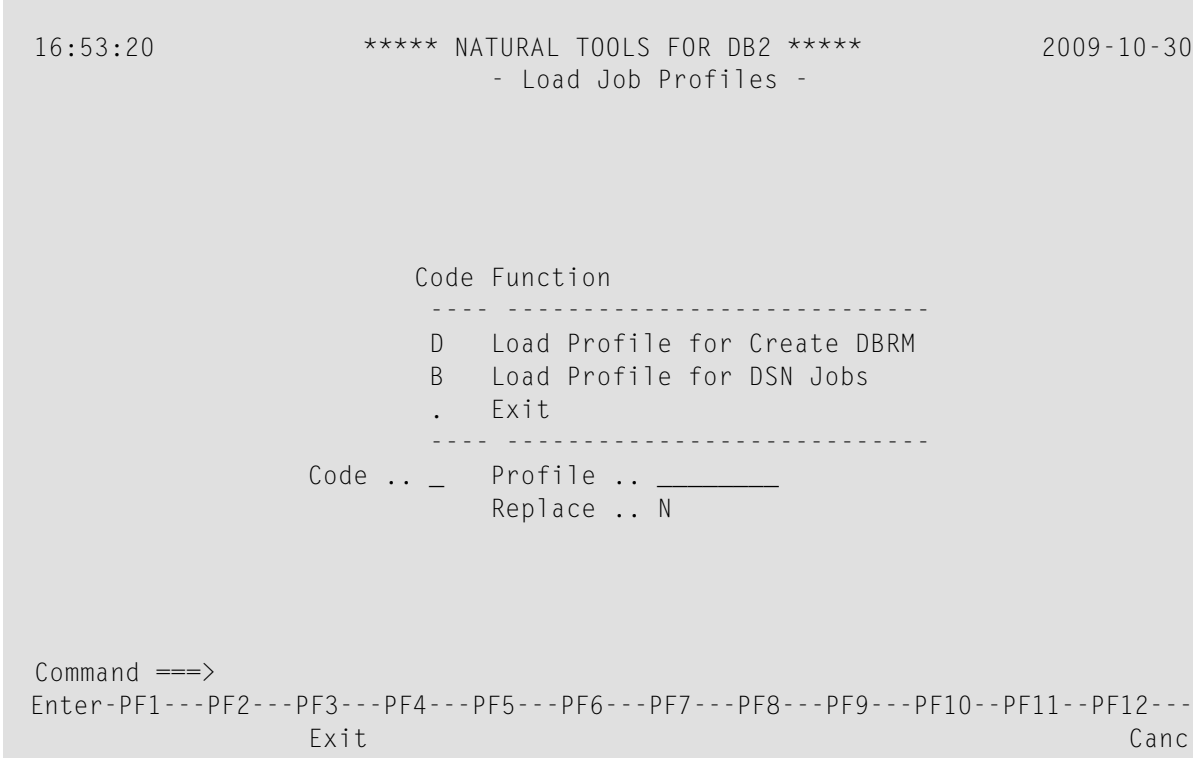
Jobprofile laden

Jobprofile für die DBRM-Erstellung und die Plan-/Package-Verwaltung werden im Batch-Modus aus dem Dataset `CMWKF01` geladen.

Um ein Jobprofil zu laden:

1. Melden Sie sich an der Library SYSDB2 an.
2. Setzen Sie in der Kommandozeile das Kommando `LOADPROF` ab.

   Das Menü **Load Job Profiles** wird angezeigt.



Die folgenden Funktionen sind verfügbar:

Code	Beschreibung
D	Dient zum Laden von Jobprofilen für die DBRM-Erstellung.
B	Dient zum Laden von Jobprofilen für die Plan- oder Package-Verwaltung.

Es gelten die folgenden Parameter:

Parameter	Beschreibung	
Profile	Gibt den Namen des zu ladenden Profils an. Dieser Parameter muss angegeben werden.	
Replace	Gibt an, ob eine Ersetzung erfolgen soll oder nicht, wenn bereits ein Profil mit dem angegebenen Namen existiert.	
	Y	Ein bereits vorhandenes Profil wird ersetzt.
	N	Ein bereits vorhandenes Profil wird *nicht* ersetzt. Dieser Parameter ist optional. Die Standardeinstellung ist N.

Jobprofile entladen

Jobprofile für die DBRM-Erstellung und die Plan-/Package-Verwaltung werden entladen und im Batch-Modus in den Dataset `CMWKF01` geschrieben.

Um ein Jobprofil zu entladen:

1 Melden Sie sich bei der Library SYSDB2 an.

2 Geben Sie in der Kommandozeile das Kommando `UNLDPROF` ein.

Das Menü **Load Job Profiles** wird angezeigt.

```
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
```

Die folgenden Funktionen sind verfügbar:

Code	Beschreibung
D	Entlädt Job-Profile für die DBRM-Erstellung.
B	Entlädt Job-Profile für die Plan- oder Package-Verwaltung.

Es gilt der folgende Parameter:

Parameter	Beschreibung
Profile	Gibt den Namen des zu entladenden Profils an. Dieser Parameter muss angegeben werden.

DBRM erstellen

Um ein DBRM zu erstellen, müssen Sie eine JCL für die DBRM-Erstellung generieren.

Um einen DBRM zu erstellen:

1 Geben Sie im Menü **Application Plan Maintenance** den Funktionscode `CD` ein, und drücken Sie Enter.

Der Bildschirm **Create DBRM** wird angezeigt. Hier können Sie neben einem Job-Namen, Ihren benutzerspezifischen Standard-Jobkarten und dem gewünschten Jobprofil alle notwendigen Informationen für die Kommandos `CREATE DBRM` und `MODIFY` angeben. Weitere Informationen siehe *Generierungsprozedur: Kommando CMD CREATE* und *Änderungsprozedur: Kommando CMD MODIFY* im Abschnitt *Programme für die statische Ausführung vorbereiten*.

```
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 Im Feld **Job Name** muss ein gültiger Job-Name angegeben werden.

Wenn Sie nur den Namen des Jobs ändern möchten, können Sie dies ebenfalls über das Feld **Job Name** tun.

3 Über das Feld **Job Cards** können Sie Ihre Standard-Jobkarten außer Funktion setzen. Geben Sie dazu ein `S` in das Feld **Job Cards** ein.

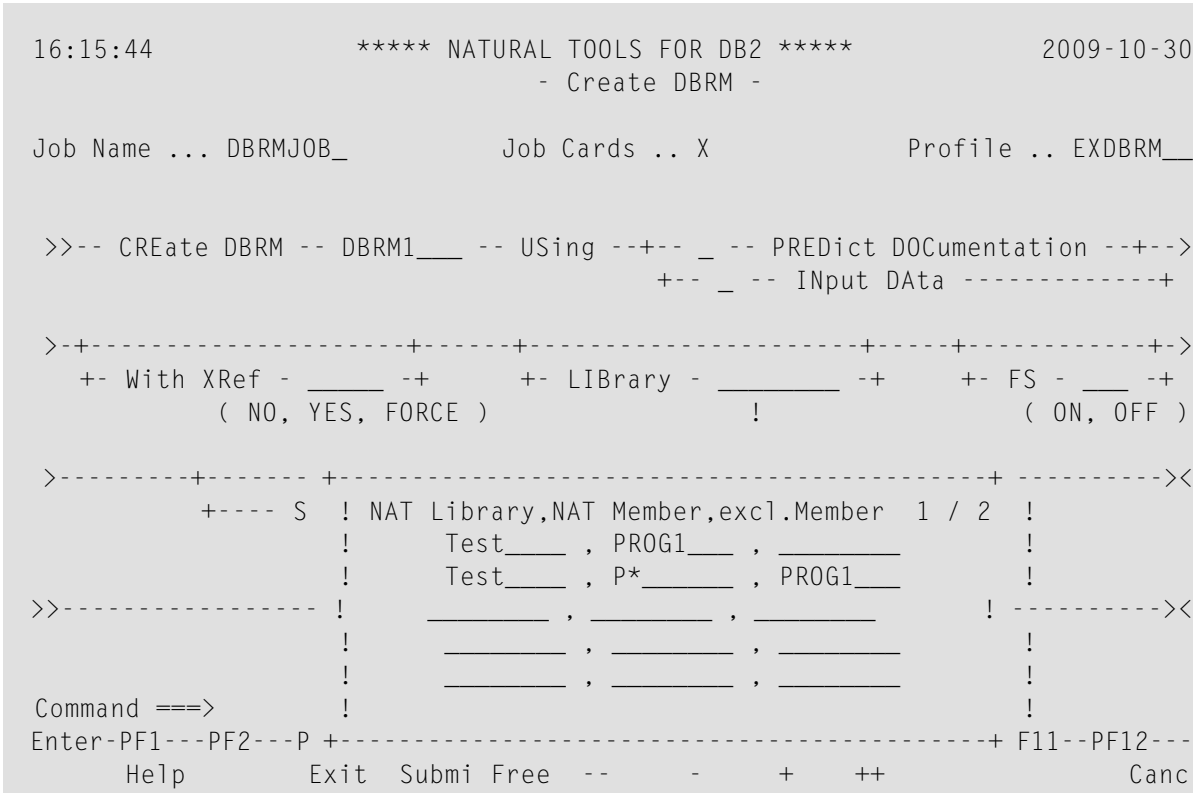
Es wird ein Fenster mit Ihren Jobkarten angezeigt.

Ein `X` im Feld **Job Cards** bedeutet, dass Jobkarten für die DBRM-Erstellung definiert sind. Ein leeres Feld **Job Cards** zeigt an, dass keine Jobkarten definiert sind.

4 Im Feld Profile können Sie den Namen eines gültigen Jobprofils für die DBRM-Erstellung angeben.

Wenn ein Wert gefolgt von einem Stern (*) angegeben wird, werden alle vorhandenen Jobprofile aufgelistet, deren Namen mit diesem Wert beginnen. Wird nur ein Stern angegeben, wird eine Auswahlliste mit allen verfügbaren Jobprofilen angezeigt.

5 Wenn Sie die Option **INput DAta** verwenden, wird ein Fenster angezeigt, in dem Sie die Natural-Libraries und -Programme (Member) angeben müssen, die in der DBRM enthalten sein sollen.



In der dritten Spalte des obigen Fensters können Sie ein Programm angeben, das aus dem DBRM ausgeschlossen werden soll. Dies ist nur möglich, wenn Sie einen Stern (*) mit dem Programmnamen in der zweiten Spalte angeben.

Innerhalb des Fensters können Sie mit PF6 (--), PF7 (-), PF8 (+) oder PF9 (++) blättern.

Der generierte JCL-Code kann entweder im Free Mode nach Drücken von PF5 (Free) bearbeitet und/oder gespeichert werden, oder durch Drücken von PF4 (Submi) sofort übergeben werden.

Plan binden

Um die JCL für das Binden eines Plans zu erzeugen, müssen Sie die Funktion **Bind** aufrufen. Alle Parameter, die für das Binden eines Plans erforderlich sind, werden auf vier Bildschirmen eingegeben, die die Syntax des DB2-Kommandos `BIND PLAN` zeigen.

Um die JCL zum Binden eines Plans zu generieren:

1 Geben Sie im Menü **Application Plan Maintenance** den Funktionscode `BI` ein.

Geben Sie im Feld **Object** den Wert `PLAN` oder `PL` ein, und drücken Sie Enter.

Der erste Bildschirm der Funktion **Bind Plan** wird angezeigt. Hier müssen Sie alle erforderlichen Informationen angeben.

```
 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 Neben den **Pflichtangaben** in den Feldern **Job Name**, **Job Cards** und **Profile** müssen Sie zum Binden eines Plans den Namen des Plans und alle DBRMs und/oder Packages angeben, die in den angegebenen Plan gebunden werden sollen.

3 Sie rufen das Fenster zur Angabe der DBRM Members und/oder Package Lists auf, indem Sie ein `S` in das Feld **MEMBER** und/oder **PKLIST** eingeben. Es müssen entweder eines oder beide Fenster aufgerufen werden, andernfalls werden Sie vom System aufgefordert, dies zu tun.

In den Fenstern für die DBRM- und Package-Angaben können Sie mit PF6 (--), PF7 (-), PF8 (+) oder PF9 (++) blättern.

4 Wenn Predict installiert ist und ein Plan in Predict dokumentiert ist, können die DBRM Members und/oder Package Lists, die einem Plan in Predict zugeordnet sind, durch Eingabe von `Y` bei dieser Option gelesen werden (Standard ist `N`). Es können maximal 50 DBRM Members und/oder 20 Package Lists gelesen werden.

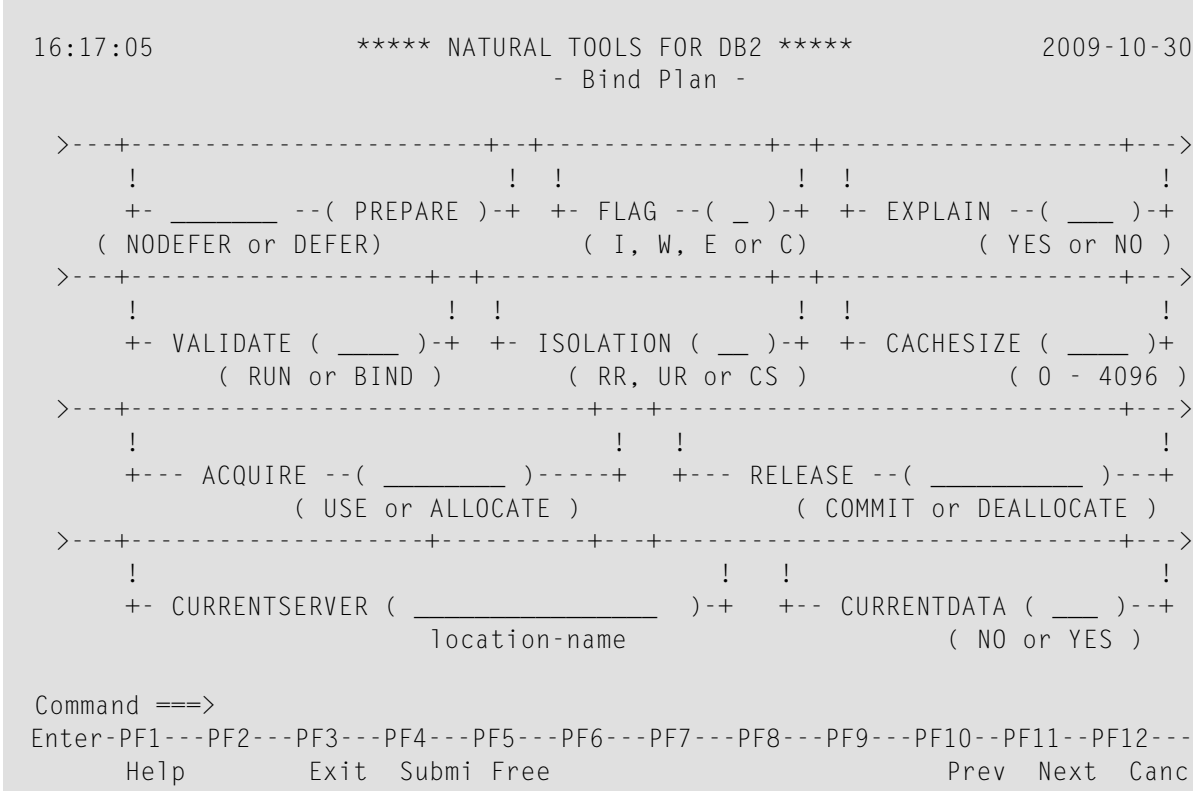
Wenn Sie diese Option verwenden und DBRM Members und/oder Package Lists erfolgreich gelesen wurden, werden die Auswahlfelder **MEMBER** und **PKLIST** mit `X` markiert und `DONE` wird neben dem Eingabefeld **(Y/N)** angezeigt; `FAILED` wird angezeigt, wenn:

- Inkonsistenzen in der Definition der Members/Package Lists festgestellt wurden,
- mehr als 50 DBRM Members oder mehr als 20 Package Lists für den angegebenen Plan definiert wurden,
- keine Members oder Package Lists für den angegebenen Plan definiert wurden,
- der Plan in Predict überhaupt nicht dokumentiert wurde.

Anmerkung: Wenn Predict nicht installiert ist, erscheint das Feld **Read member name / package list from Predict?** nicht auf dem obigen Bildschirm.

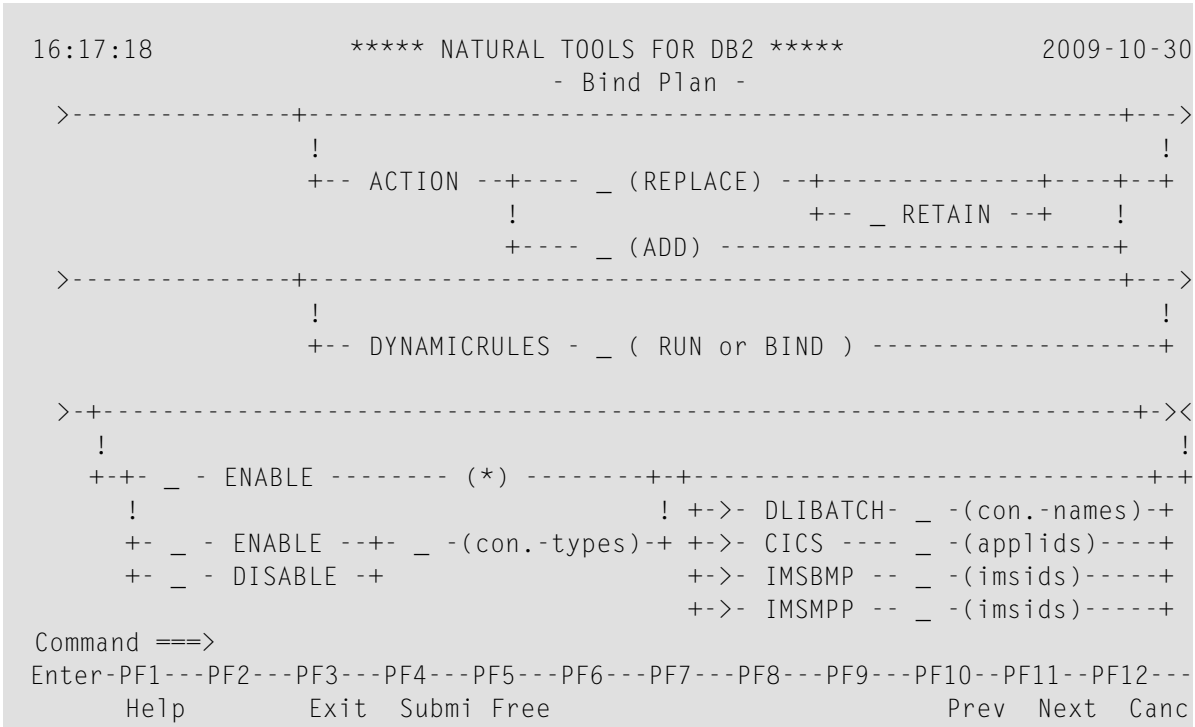
5 Durch Drücken von PF11 (Next) gelangen Sie auf einen zweiten **Bind Plan**-Bildschirm, auf dem Sie weitere Optionen des Db2-Kommandos `BIND` angeben können.

Ein Schlüsselwort wird durch Eingabe des ersten Buchstabens in das entsprechende Eingabefeld generiert. Die Standardwerte sind hervorgehoben.



Durch Drücken von PF10 (Prev) gelangen Sie zum vorherigen Bildschirm zurück.

6 Durch Drücken von PF11 (Next) gelangen Sie zu einem dritten **Bind Plan**-Bildschirm, in dem Sie weitere Optionen des Db2-Kommandos `BIND` angeben können.



7 Durch Drücken von PF11 (Next) gelangen Sie zu einem vierten **Bind Plan**-Bildschirm, in dem Sie weitere Optionen des Db2-Kommandos `BIND` auswählen können.

```
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 Der generierte JCL-Code kann entweder durch Drücken von PF5 (Free) im Free Mode bearbeitet und/oder gespeichert werden, oder durch Drücken von PF4 (Submi) sofort übergeben werden.

Plan neu binden

Um die JCL für das Neubinden eines Plans zu generieren, müssen Sie die Funktion **Rebind** aufrufen. Alle Parameter, die für das Neubinden eines Plans erforderlich sind, werden über drei Bildschirme eingegeben, die die Syntax des Db2-Kommandos `REBIND PLAN` zeigen.

Um die JCL für das Neubinden eines Plans zu generieren:

1 Geben Sie im Menü **Application Plan Maintenance** den Funktionscode `RB` ein.

Geben Sie im Feld **Object** `PLAN` oder `PL` ein, und drücken Sie Enter.

Es wird der erste Bildschirm **Rebind Plan** angezeigt, auf dem alle erforderlichen Informationen angegeben werden müssen.

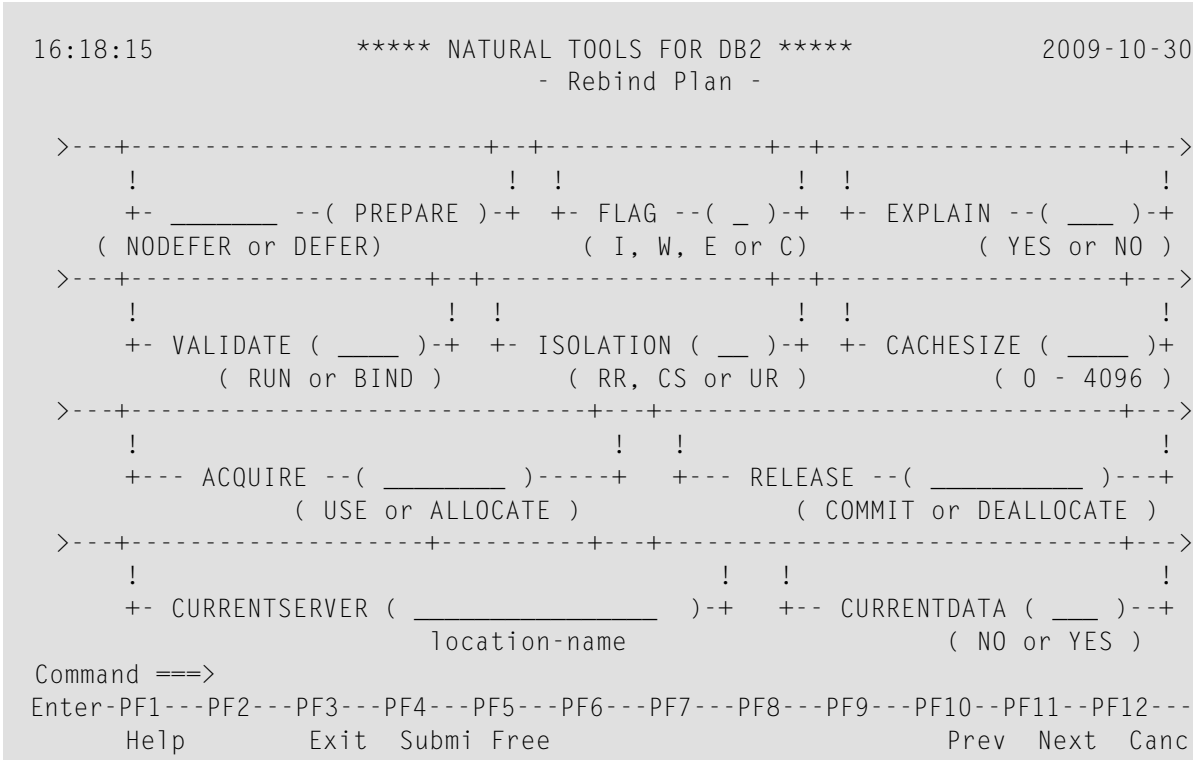
```
19:17:55             ***** NATURAL TOOLS FOR DB2 *****            2009-10-30
                                - Rebind Plan -

 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 Neben den **Angaben** in den Feldern **Job Name**, **Job Cards** und **Profile** müssen Sie in einem Fenster die Namen der Pläne angeben, die neu gebunden werden sollen.

Wenn Sie die Stern-Notation (*) angeben, werden alle vorhandenen Pläne neu gebunden.

3 Durch Drücken von PF11 (Next) gelangen Sie zu einem zweiten **Rebind Plan**-Bildschirm, in dem Sie weitere Optionen des Db2-Kommandos REBIND angeben können.

Ein Schlüsselwort wird durch Eingabe seines Anfangsbuchstabens in das entsprechende Eingabefeld erzeugt. Die Standardwerte sind hervorgehoben.



Wenn Sie PF10 (Prev) drücken, gelangen Sie zum vorherigen Bildschirm zurück.

4 Durch Drücken von PF11 (Next) gelangen Sie zu einem dritten **Rebind Plan**-Bildschirm, auf dem Sie weitere Optionen des Db2-Kommandos `REBIND` angeben können.

```
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 Der generierte JCL-Code kann entweder durch Drücken von PF5 (Free) im Free Mode bearbeitet und/oder gespeichert werden, oder durch Drücken von PF4 (Submi) sofort übergeben werden.

Plan freigeben

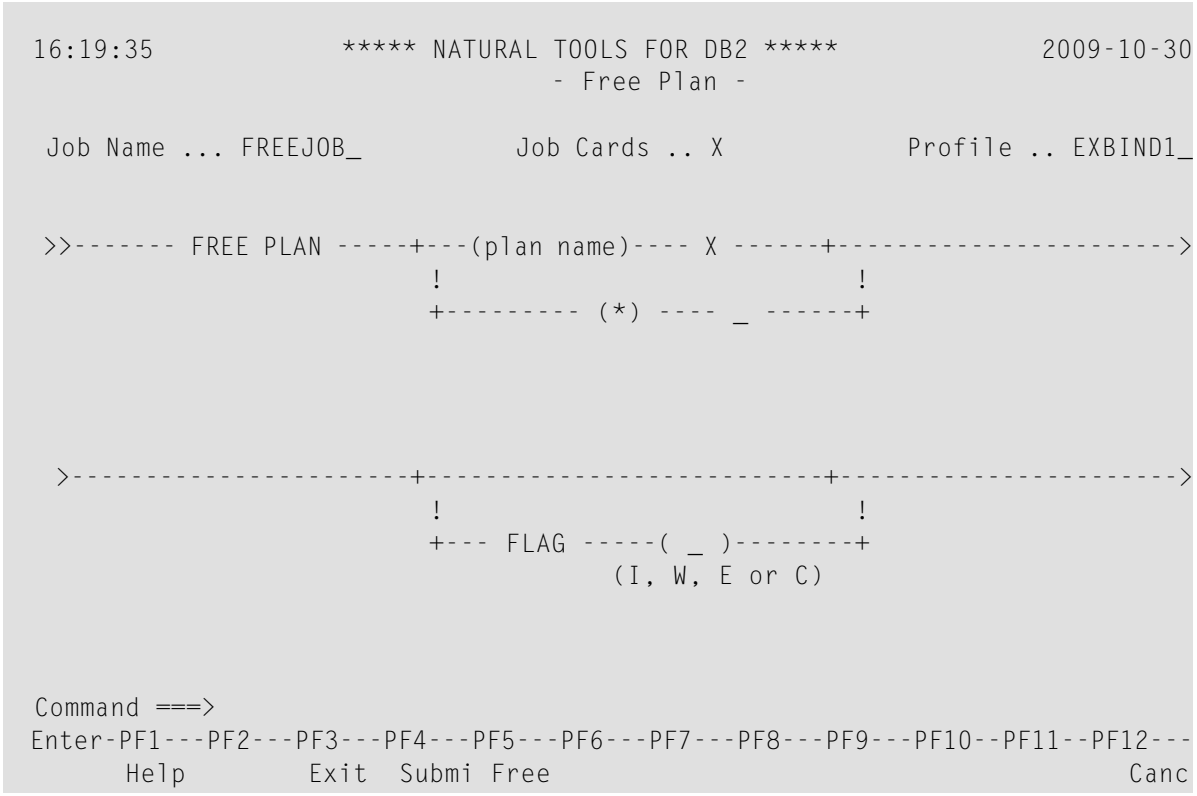
Ein freier Plan kann mit der Funktion **Free** des Menüs **Application Plan Maintenance** erzeugt werden.

Um die JCL zum Freigeben eines Plans zu generieren:

1 Geben Sie im Menü **Application Plan Maintenance** den Funktionscode `FR` ein.

Geben Sie im Feld **Object** `PLAN` oder `PL` ein, und drücken Sie **ENTER**.

Es erscheint der Bildschirm **Free Plan**, auf dem alle erforderlichen Informationen angegeben werden müssen.



2 Neben den Angaben in den Feldern **Job Name**, **Job Cards** und **Profile** werden in einem Fenster, das die Syntax des Db2-Kommandos `FREE PLAN` zeigt, alle zur Freigabe eines Plans notwendigen Parameter eingegeben.

Die Namen der freizugebenden Pläne werden in ein Fenster eingetragen. Wenn Sie Stern-Notation (*) verwenden, werden alle Pläne freigegeben.

3 Der generierte JCL-Code kann entweder durch Drücken von PF5 (Free) im Free Mode bearbeitet und/oder gespeichert oder durch Drücken von PF4 (Submi) sofort übergeben werden.

Package binden

Packages können mit der Funktion **Bind** des Menüs **Application Plan Maintenance** gebunden werden.

Alle für das Binden eines Package erforderlichen Parameter werden auf drei Bildschirmen eingegeben, die die Syntax des Db2-Kommandos `BIND PACKAGE` zeigen.

Um die JCL zum Binden eines Package zu generieren:

1 Geben Sie im Menü **Application Plan Maintenance** den Funktionscode „BI“ ein.

Geben Sie im Feld **Object** `PACKAGE` oder `PK` ein, und drücken Sie Enter.

Es erscheint der erste Bildschirm **Bind Package**, auf dem alle erforderlichen Informationen angegeben werden müssen.

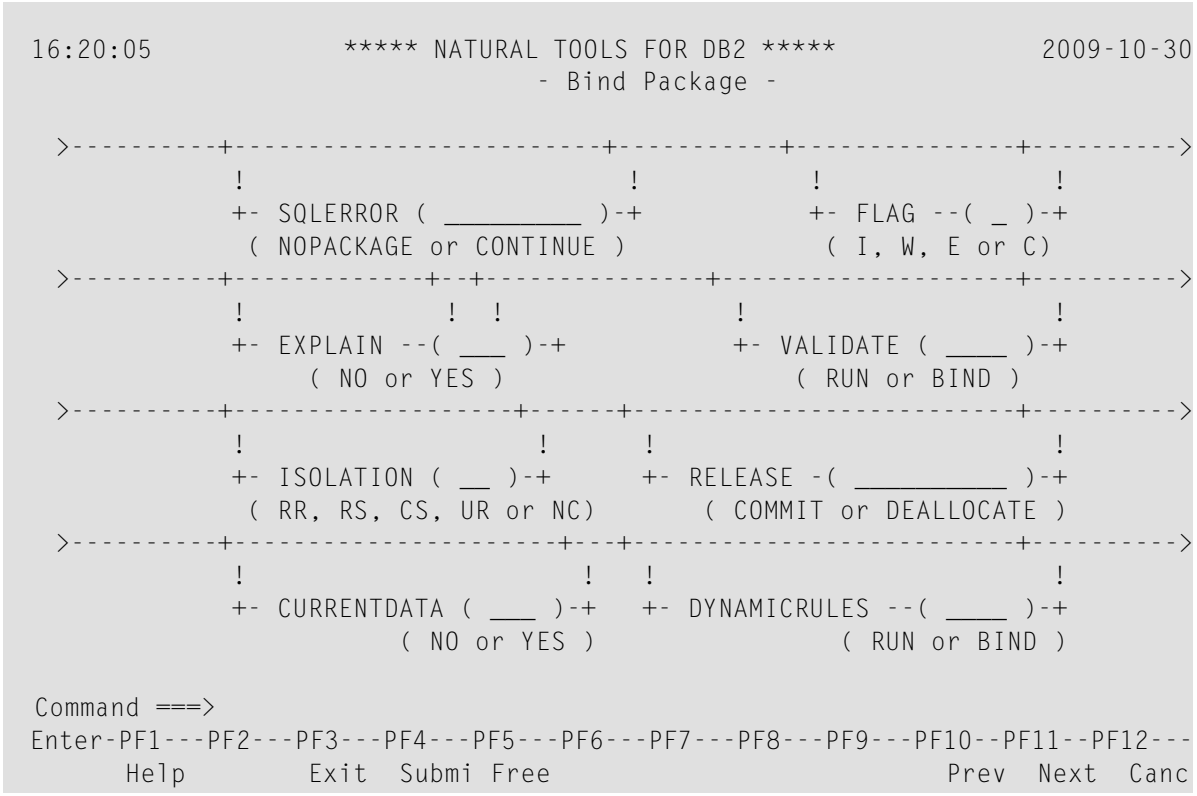
```
16:19:58             ***** NATURAL TOOLS FOR DB2 *****            2009-10-30
                                - Bind Package -

 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 Neben den **Angaben** in den Feldern **Job Name**, **Job Cards** und **Profile** müssen Sie zum Binden eines Package die **Collection ID** des Package und einen DBRM oder ein weiteres Package angeben, das in das angegebene Package eingebunden werden soll.

Das DBRM oder das zweite Package geben Sie im Feld **MEMBER** bzw. **COPY** an. Eines der beiden Felder muss ausgewählt werden und die Package ID ist entweder der DBRM-Name oder die Package ID des kopierten Package.

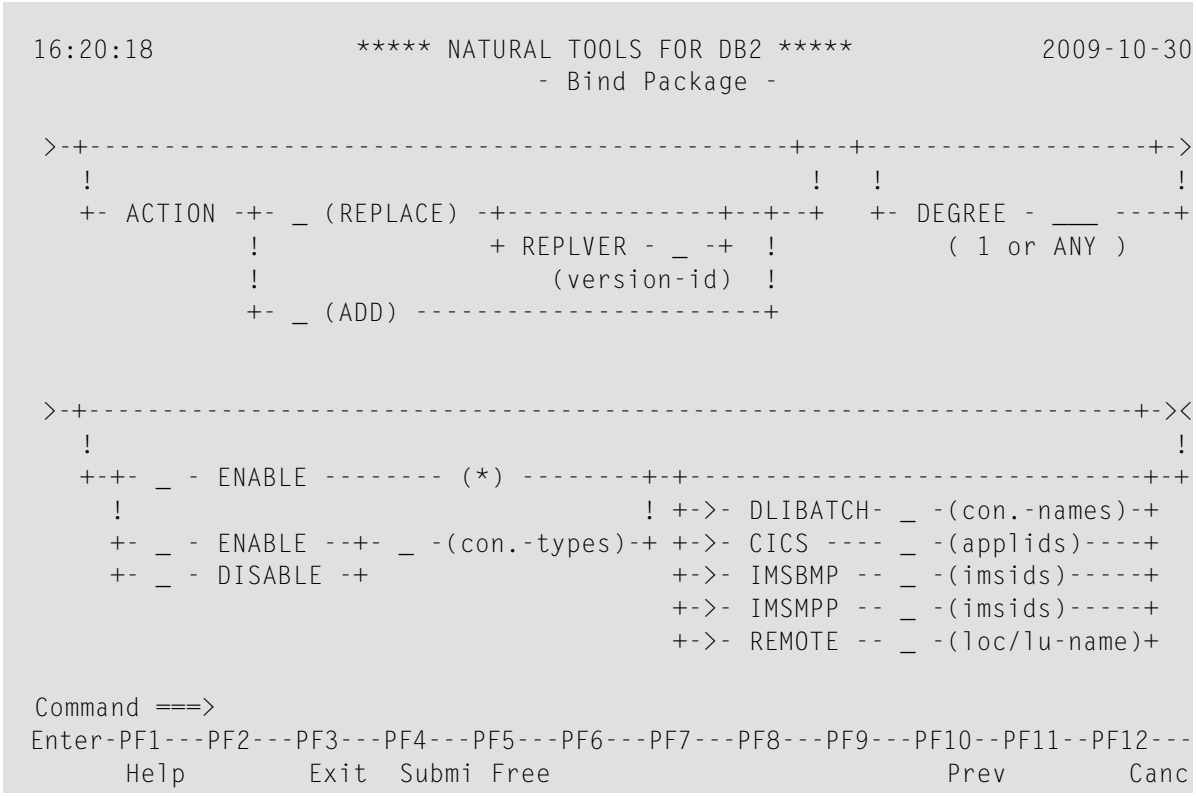
3 Durch Drücken von PF11 (Next) gelangen Sie zu einem zweiten **Bind Package**-Bildschirm, auf dem Sie weitere Optionen des Db2-Kommandos `BIND` angeben können.

Ein Schlüsselwort wird durch Eingabe des ersten Buchstabens in das entsprechende Eingabefeld erzeugt. Die Standardwerte sind hervorgehoben.



Durch Drücken von PF10 (Prev) kehren Sie zum vorherigen Bildschirm zurück.

4 Durch Drücken von PF11 (Next) gelangen Sie zu einem dritten **Bind Package**-Bildschirm, auf dem Sie weitere Optionen des Db2-Kommandos `BIND` angeben können.



5 Der generierte JCL-Code kann entweder durch Drücken von PF5 (Free) im Free Mode bearbeitet und/oder gespeichert oder durch Drücken von PF4 (Submi) sofort übergeben werden.

Package neu binden

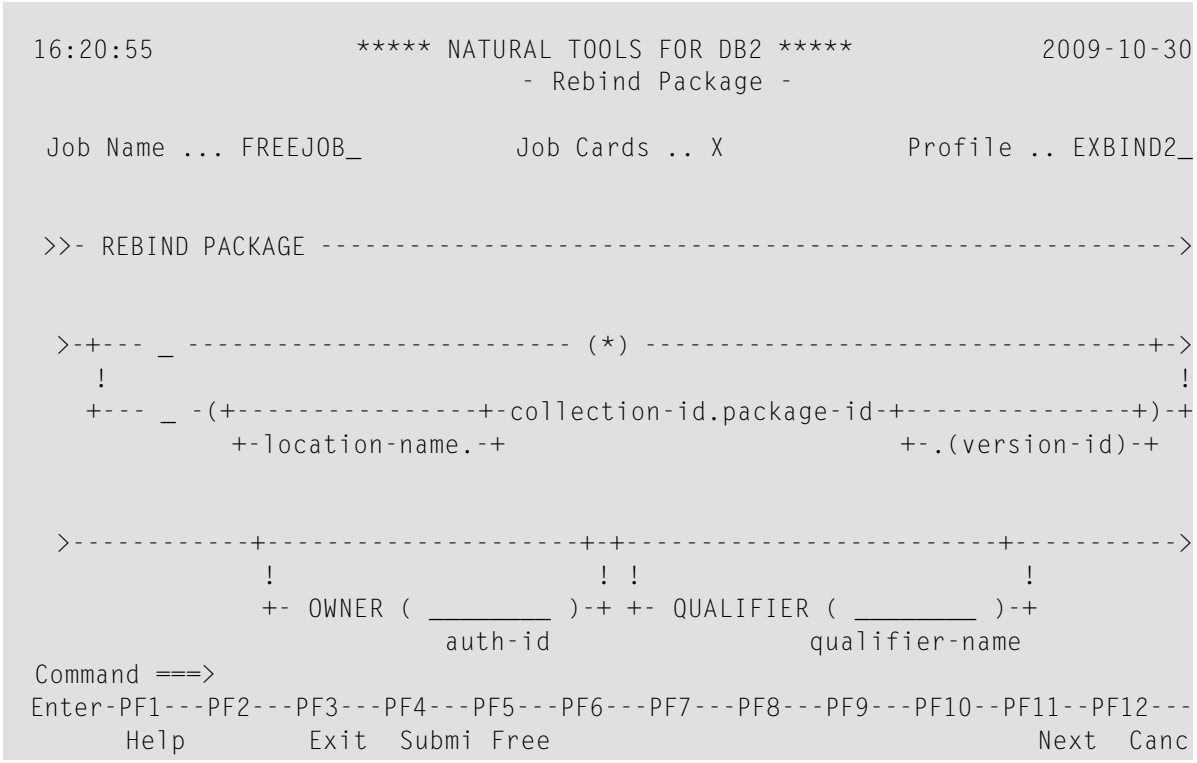
Ein Package kann mit der Funktion **Rebind** des Menüs **Application Plan Maintenance** neu gebunden werden. Alle für das Neubinden eines Package erforderlichen Parameter werden in zwei Bildschirmen eingegeben, die die Syntax des Db2-Kommandos `REBIND PACKAGE` zeigen

Um die JCL für das Neubinden eines Package zu generieren:

1 Geben Sie im Menü **Application Plan Maintenance** den Funktionscode `RB` ein.

Geben Sie im Feld **Object** `PACKAGE` oder `PK` ein, und drücken Sie Enter.

Es erscheint der erste Bildschirm **Rebind Package**, auf dem alle erforderlichen Informationen angegeben werden müssen.

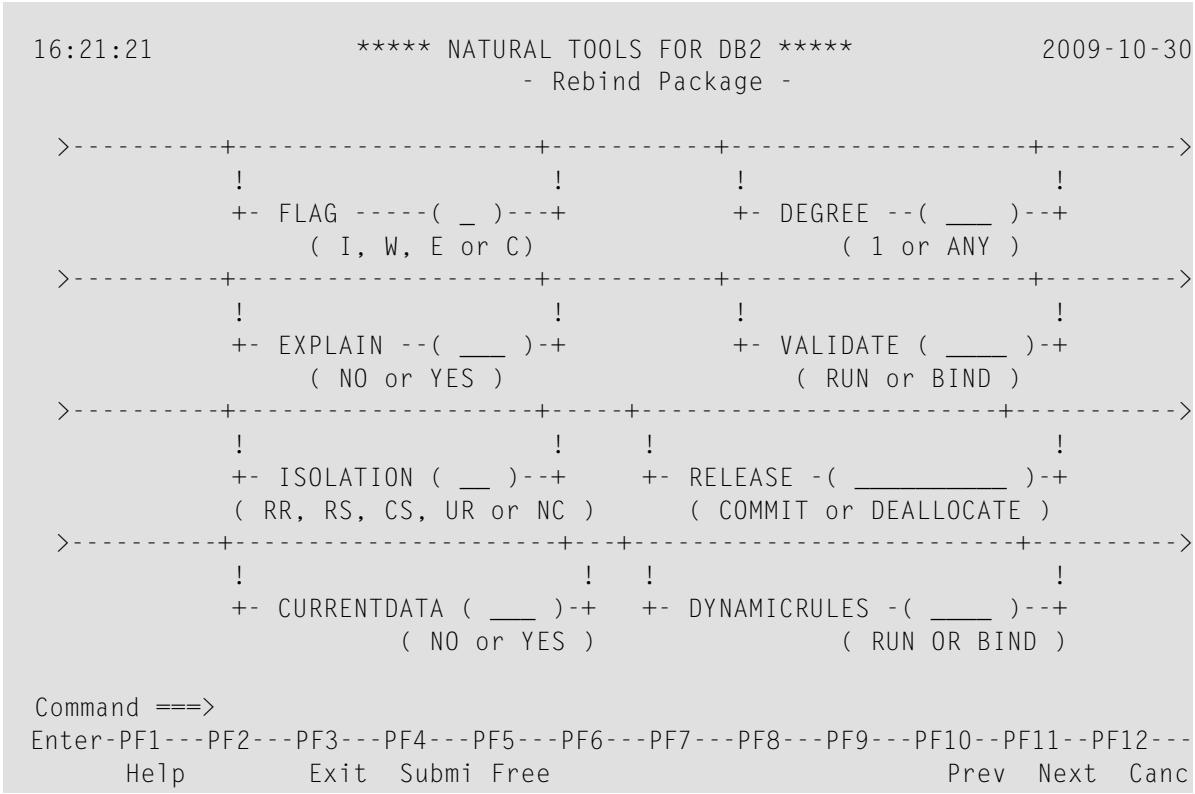


2 Neben den **Angaben**, die in den Feldern **Job Name**, **Job Cards** und **Profile** erforderlich sind, müssen Sie in einem Fenster die Namen der Packages angeben, die neu gebunden werden sollen.

Wenn Sie Stern-Notation (*) verwenden, werden alle lokal vorhandenen Packages neu gebunden.

3 Durch Drücken von PF11 (Next) gelangen Sie zu einem zweiten **Rebind Package**-Bildschirm, in dem Sie weitere Optionen des Db2-Kommandos `REBIND` angeben können.

Ein Schlüsselwort wird durch Eingabe seines Anfangsbuchstabens in das entsprechende Eingabefeld erzeugt. Die Standardwerte sind hervorgehoben.



Wenn Sie PF10 (Prev) drücken, gelangen Sie zum vorherigen Bildschirm zurück.

4 Durch Drücken von PF11 (Next) gelangen Sie zu einem dritten **Rebind Package**-Bildschirm, auf dem Sie weitere Optionen für das Db2-Kommando `REBIND` angeben können.

```
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 Der generierte JCL-Code kann entweder durch Drücken von PF5 (Free) im Free Mode bearbeitet und/oder gespeichert werden oder durch Drücken von PF4 (Submi) sofort übergeben werden.

Package freigeben

Ein Package kann mit der Funktion **Free Package** aus dem Menü **Application Plan Maintenance** freigegeben werden.

Um die JCL zum Freigeben eines Package zu generieren:

1 Geben Sie im Menü **Application Plan Maintenance** den Funktionscode `FR` ein.

Geben Sie im Feld **Object** `PACKAGE` oder `PK` ein, und drücken Sie Enter.

Es erscheint der Bildschirm **Free Package**, auf dem alle erforderlichen Informationen angegeben werden müssen.

```
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 Neben den **Angaben**, die in den Feldern **Job Name**, **Job Cards**, and **Profile** erforderlich sind, werden alle Parameter, die zur Freigabe eines Package notwendig sind, in einem Fenster eingegeben, das die Syntax des Db2-Kommandos `FREE PACKAGE` zeigt.

Die Namen der freizugebenden Packages werden in einem Fenster eingegeben. Wenn Sie Stern-Notation (*) verwenden, werden alle lokalen Packages freigegeben.

3 Der generierte JCL-Code kann entweder durch Drücken von PF5 (Free) im Free Mode bearbeitet und/oder gespeichert werden oder durch Drücken von PF4 (Submi) sofort übergeben werden.

JCL auflisten

Die Funktion **List JCL** dient zum Aufrufen des Free-Mode-Editors über das Menü **Application Plan Maintenance**.

Um die Funktion List JCL aufzurufen:

1 Geben Sie im Menü **Application Plan Maintenance** den Funktionscode `LJ` ein.

- Wenn Sie das Feld **JCL Member** leer lassen und Enter drücken, erscheint der leere Editor-Bildschirm für den Free Mode.

- Wenn Sie einen Wert gefolgt von einem Stern (*) oder nur einen Stern angeben und Enter drücken, wird eine Liste mit JCL Members zur Auswahl angezeigt.
- Wenn Sie einen gültigen Member-Namen angeben und Enter drücken, enthält der aufgerufene Free-Mode-Editor die entsprechende 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
```

Im Free-Mode-Editor können JCL-Member kopiert, aufgelistet, gelöscht, abgerufen oder in einer Natural Library gespeichert werden. Dazu stehen entsprechende Verwaltungskommandos zur Verfügung, siehe *Globale Verwaltungskommandos*.

2 Drücken Sie PF4 (Submi), um den im Editor aufgelisteten JCL-Code zu übergeben, drücken Sie PF5 (Fix), um in den Fixed Mode zu wechseln.

Job-Ausgabe anzeigen

Mit der Funktion **Display Job Output** können Sie sich die Ausgabe eines JCL-Members anzeigen lassen.

Anmerkung: Die Funktion **Display Job Output** ist nur verfügbar, wenn der Entire System Server installiert ist.

Um die Ausgabe eines JCL-Members anzuzeigen:

1 Geben Sie im Menü **Application Plan Maintenance** den Funktionscode `JO` ein.

Im Feld Node kann die Standardknotennummer (`148`) für den Entire System Server geändert werden.

Es wird ein Bildschirm angezeigt, auf dem Sie den gewünschten Job-Namen und die Job-Nummer sowie die Nummern der `SYSOUT`-Typen angeben können.

```
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 Im Feld **Job Name** kann ein gültiger Job-Name angegeben werden.

- Wenn Sie einen Wert gefolgt von einem Stern (*) oder nur Stern-Notation (*) verwenden, wird eine Liste der Job-Ausgabe-Members zur Auswahl angezeigt.

  In einer Auswahlliste der Job-Ausgabe-Members können Sie ein Ausgabe-Member entweder mit `B` markieren, um nur das Member anzuzeigen, oder mit `L`, um eine Liste aller `SYSOUT`-Datasets der Job-Ausgabe anzuzeigen, die ihrerseits mit `B` zur Anzeige markiert werden können.
- Wenn Sie das Feld **Job Name** leer lassen, müssen Sie eine Jobnummer angeben.

3 Im Feld **Job Number** können Sie eine eindeutige Jobnummer angeben. Nur wenn eine eindeutige Jobnummer angegeben wurde, können auch in den Feldern **Sysout Type** und **Sysout Number** Angaben gemacht werden.

4 Im Feld **Sysout Type** können Sie den Typ des SYSOUT-Datasets des Jobs mit der angegebenen Jobnummer angeben, der angezeigt werden soll. Es gelten die folgenden Codes:

Code	SYSOUT Type	SYSOUT-Typ
CC	Condition Code	Bedingungscode
JL	Job Listing	Job-Auflistung
SI	System Input	System-Eingabe
SM	System Message	System-Meldung
SO	System Output	System-Ausgabe

5 Im Feld **Sysout Number** können Sie eine Dateinummer angeben, um ein bestimmtes SYSOUT-Dataset des im Feld **Sysout Type** angegebenen Typs anzuzeigen.

Wenn Sie das Feld **Sysout Number** leer lassen, werden alle SYSOUT-Datasets des angegebenen Typs angezeigt.

6 Katalog verwalten

Mit der Funktion **Catalog Maintenance** der **Natural Tools for Db2** können Sie SQL-Statements zur Verwaltung des Db2-Katalogs (d. h. von Db2-Tabellen und anderen Db2-Objekten) generieren, ohne Ihre Entwicklungsumgebung zu verlassen.

Die Funktion **Catalog Maintenance** enthält einen SQL-Generator, der aus Ihren Eingaben automatisch den SQLCODE generiert, der für die Pflege des gewünschten Db2-Objekts erforderlich ist. Sie können den generierten SQLCODE anzeigen, ändern, speichern und abrufen.

Die DDL/TML-Definitionen werden in der aktuellen Natural Library gespeichert.

Fixed Mode und Free Mode

Die Katalogverwaltungsfunktion bietet zwei Betriebsarten: Fixed Mode und Free Mode.

Um vom Fixed Mode in den Free Mode zu wechseln:

- Drücken Sie PF5 (Free).

Um vom Free Mode in den Fixed Mode zurückzukehren:

- Drücken Sie PF3 (Exit) im Free Mode.

Fixed Mode

Im Fixed Mode stehen Ihnen Eingabebildschirme mit Syntaxgraphen zur Verfügung, die Sie bei der Eingabe des korrekten SQLCODE unterstützen. Sie geben einfach die erforderlichen Daten in die Eingabemasken ein, und die Daten werden automatisch auf Übereinstimmung mit der Db2-SQL-Syntax überprüft. Ist die Eingabe unvollständig, werden Sie aufgefordert, die fehlenden Daten einzugeben. Anschließend werden aus den eingegebenen Daten SQL-Members generiert. Die Members können durch Drücken von PF4 (Submi) direkt ausgeführt werden. Sie können aber auch PF5 (Free) drücken, um in den Free Mode zu wechseln, in dem der generierte SQLCODE verändert werden kann.

Nach der Ausführung eines SQL-Statements wird eine Meldung zurückgegeben, die anzeigt, dass das Statement erfolgreich ausgeführt wurde. Wenn ein Fehler aufgetreten ist, kann die entsprechende Db2-Fehlermeldung durch Drücken von PF2 (Error) angezeigt werden, wodurch das Kommando `SQLERR` ausgeführt wird.

Die Eingabebildschirme bestehen aus verschiedenen Arten von Eingabefeldern. Dazu gehören:

- Felder zur Eingabe von Db2-Objektnamen,
- Felder zum Aufrufen von Fenstern,
- Felder, die zur Auswahl markiert werden,

- Felder für die Eingabe von Schlüsselwörtern,
- Felder für die Angabe numerischer Werte,
- Felder für die Eingabe von String-Konstanten.

In jedem Feld, in dem ein Fenster aufgerufen werden kann, können Sie ein S angeben. Wenn Sie Enter drücken, erscheint das Fenster und Sie können die erforderlichen Informationen auswählen oder eingeben. Wenn eine solche Auswahl erforderlich ist, ist beim Aufruf des entsprechenden Bildes bereits ein S voreingestellt.

Wenn Sie erneut Enter drücken, wird das Fenster geschlossen, und wenn Daten eingegeben wurden, wird das Feld mit X statt mit S gekennzeichnet. Wenn nicht, bleibt das Feld leer oder wird erneut mit S gekennzeichnet.

Dies wird jedes Mal fortgesetzt, wenn Sie Enter drücken, bis kein S mehr vorhanden ist. Um ein Fenster, in das Daten eingegeben wurden, wieder anzuzeigen, ändern Sie die Markierung X wieder in S.

Wenn ein anderer Buchstabe oder ein anderes Zeichen verwendet wird, erscheint eine Fehlermeldung auf dem Bildschirm.

Markieren Sie das Feld mit S, um das Fenster anzuzeigen.

Das falsche Zeichen wird automatisch durch ein S ersetzt, und wenn Sie erneut Enter drücken, wird das entsprechende Fenster angezeigt.

In Feldern, in denen Schlüsselwörter eingegeben werden sollen, müssen Sie eines der unter dem Feld angezeigten Schlüsselwörter eingeben. Standardschlüsselwörter sind hervorgehoben.

Free Mode

Wenn der Free Mode vom Fixed Mode aus aufgerufen wird (durch Drücken von PF5 (Free), werden die Daten, die im Fixed Mode eingegeben wurden, als generierter SQLCODE angezeigt, der zur späteren Verwendung oder Änderung gespeichert werden kann.

Wenn Sie ein SQL-Member im Free Mode ändern, hat dies keine Auswirkungen auf die Version des Member im Fixed Mode. Sie können Ihren geänderten Code im Free Mode speichern, aber wenn Sie in den Fixed Mode zurückkehren, erscheinen wieder die Originaldaten. Es sind also sowohl die Originaldaten als auch die geänderten Daten verfügbar.

Im Free Mode können Sie das Member, das sich gerade im Quellcodebereich befindet, durch Drücken von PF4 (Submi) ausführen, wie im Fixed Mode.

Bei der Ausführung von SQL-Statements wird automatisch auf den Ausgabebildschirm umgeschaltet, auf dem der Rückgabecode der ausgeführten Kommandos angezeigt wird.

Eine Liste der im Free Mode verfügbaren SQLCODE-Pflegekommandos finden Sie im Abschnitt *Globale Verwaltungskommandos*.

Funktion Catalog Maintenance aufrufen

Um die Funktion Catalog Maintenance aufzurufen:

- Geben Sie im Hauptmenü **Natural Tools for DB2 Main Menu** den Funktionscode `C` ein, und drücken Sie Enter.

  Das Menü **Catalog Maintenance** wird angezeigt:

```
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
```

  Im Feld **Code** kann der Funktionscode, der der gewünschten Funktion zugeordnet ist, zusammen mit dem gewünschten Objekt-, Library- und/oder Member-Namen angegeben werden.

  Wenn Sie in den Free Mode wechseln und einen gültigen Member-Namen eingeben, können Sie dieses Member aus der mit dem Parameter **Library** angegebenen Natural Library lesen. Der **Library**-Parameter ist mit Ihrer Natural-Benutzerkennung vorbelegt.

  Bei den Funktionen `CREATE VIEW` und `EXPLAIN` muss ein Subselect bzw. ein explainable SQL-Statement eingegeben werden. Beides kann in einer separaten Editor-Sitzung erfolgen, in der zuvor gespeicherte Members verwendet werden können. Der Editor wird durch Eingabe eines `S` in das entsprechende Feld aufgerufen.

Bei den Funktionen CREATE, ALTER, GRANT und REVOKE muss ein Objektcode angegeben werden, zum Beispiel TB für TABLE. Wenn Sie das Objektfeld leer lassen, wird ein Fenster angezeigt, das Ihnen eine Liste aller verfügbaren Objekte mit ihren Objektcodes anzeigt.

Wenn Sie z.B. die Funktion CREATE eingeben, ohne ein Objekt anzugeben, wird ein Fenster aufgerufen, in dem Sie nach der Art des zu erstellenden Objekts gefragt werden:

```
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
```

Im folgenden Abschnitt wird anhand von Beispielen gezeigt, wie Sie die **Catalog Maintenance**-Funktion im Fixed Mode verwenden können.

Tabelle erstellen - Funktion: Create Table

Um die Funktion Create Table aufzurufen:

1 Geben Sie in der Funktion CREATE den Objektcode TB ein, und drücken Sie Enter.

Der erste **Create Table**-Syntaxeingabebildschirm wird angezeigt.

Auf diesem Bildschirm können Sie den Namen des Erstellers (`creator`) und der Tabelle (`table-name`) sowie die einzelnen Spaltennamen (`column-name`), Spaltenformate (`format`) und Spaltenlängen (`length`) eingeben, wie im folgenden Beispiel dargestellt:

```
  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
```

Anmerkung: Da die Angabe von Sonderzeichen als Teil eines Natural-Feld- oder DDM-Namens den Natural-Namenskonventionen widerspricht, sollten alle in Db2 zulässigen Sonderzeichen vermieden werden. Das Gleiche gilt für Db2-Bezeichner mit Begrenzungszeichen (Delimited Identifiers), die von Natural nicht unterstützt werden.

In der oberen rechten Ecke des Bildschirms wird der Index der obersten Spalte (1) und die Gesamtzahl der angegebenen Spalten (9) angezeigt. Wenn Sie mehr Spalten angeben wollen, als auf einen Terminal-Bildschirm passen, drücken Sie PF8 (+), um eine Seite vorwärtszublättern.

Ein `S` im Feld **S/M/B** der Spalte `COL4` bedeutet, dass die Option `FOR SBCS DATA` für diese Spalte ausgewählt ist. Andere mögliche Werte für dieses Feld sind `M` (`FOR MIXED DATA`) und `B` (`FOR BIT DATA`).

Die Spalten `COL3`, `COL2` und `COL5` bilden den Primärschlüssel in der angegebenen Reihenfolge. Primärschlüsselspalten müssen mit einem `S` ausgewählt oder durch Angabe entsprechender Zahlen zwischen 1 und 16 geordnet werden. Im vorliegenden Beispiel sind alle Primärschlüsselspalten als `NOT NULL` definiert. Darüber hinaus ist Spalte 7 als `NOT NULL` angegeben.

Für Spalte 6 wurde eine Feldprozedur in einem Fenster eingetragen, das durch `S` aufgerufen wurde. Das Fenster wurde wieder geschlossen, und das Feld **fld proc** ist nun mit `X` markiert.

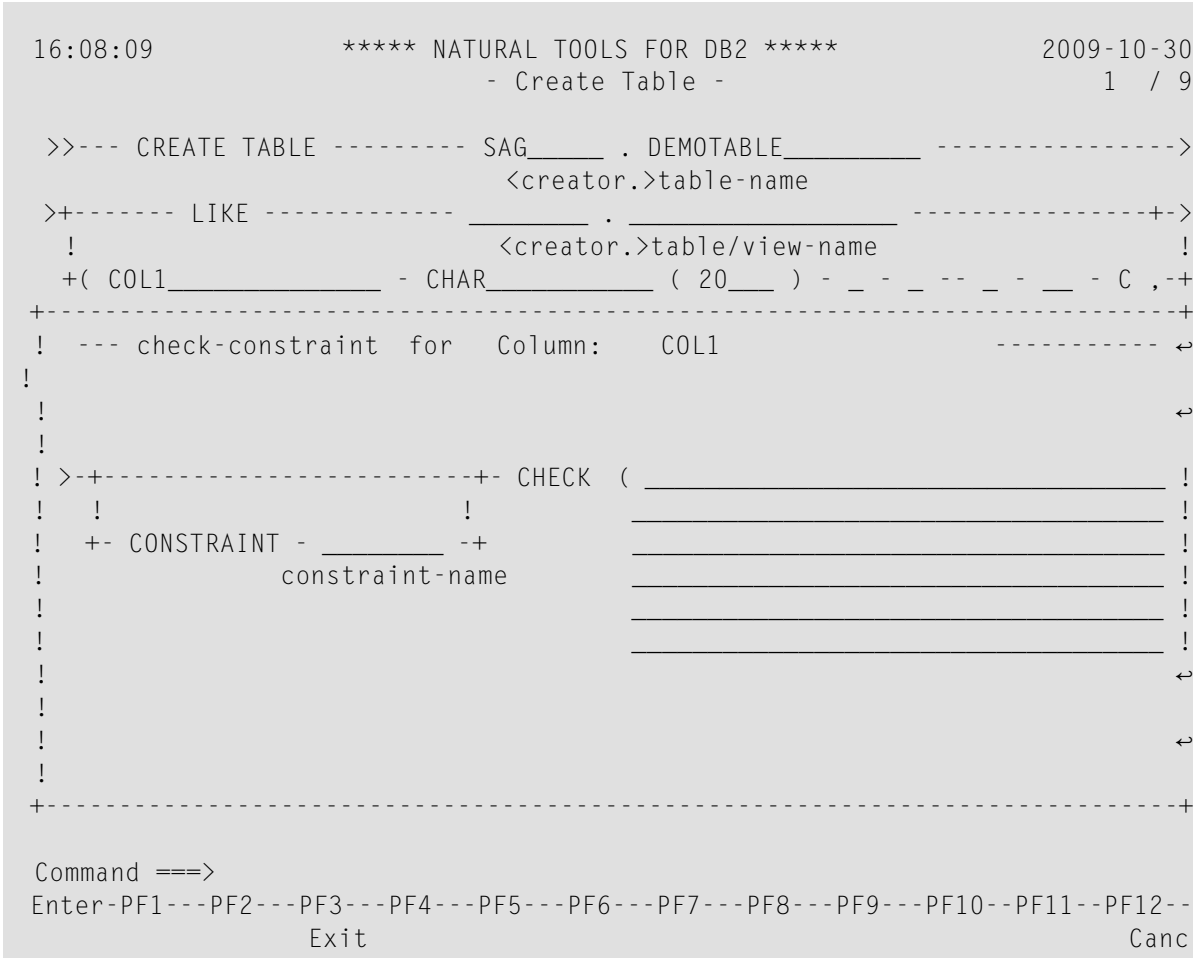
2 Wenn Sie für eine bestimmte Spalte ein `R` in das Feld **R/C/D/G** eingeben und Enter drücken, wird ein Fenster angezeigt, in dem Sie eine References-Klausel angeben können, die diese Spalte als Fremdschlüssel (Foreign Key) einer referentiellen Einschränkung (Constraint) kennzeichnet.

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

Sie müssen den Namen (`table-name`) der übergeordneten Tabelle mit einem optionalen Erstellernamen (`creator`) angeben, auf die verwiesen werden soll. Darüber hinaus müssen Sie die Aktion angeben, die ausgeführt werden soll, wenn eine Zeile in der referenzierten Tabelle gelöscht wird. Die folgenden Optionen sind verfügbar:

- `RESTRICT` oder `NO ACTION` verhindert die Löschung der übergeordneten Zeile, bis alle abhängigen Zeilen gelöscht sind.
- `CASCADE` löscht auch alle abhängigen Zeilen.
- `SET NULL` setzt alle Spalten des Fremdschlüssels in jeder abhängigen Zeile, die Nullwerte enthalten können, auf Null.
- Ein Schlüssel, der aus mehr als einer Spalte besteht, muss durch eine `FOREIGN KEY`-Klausel definiert werden.

3 Wenn Sie in das Feld **R/C/D/G** für eine bestimmte Spalte ein `C` eingeben und Enter drücken, wird ein Fenster angezeigt, in dem Sie eine Prüfeinschränkung (`check-constraint`) für diese Spalte angeben können.



Sie müssen eine Spaltenprüfbedingung angeben. Eine Prüfbedingung ist eine Suchbedingung mit verschiedenen Einschränkungen, die in der entsprechenden Db2-Literatur von IBM ausführlich beschrieben sind. Darüber hinaus können Sie einen Namen für die Prüfbedingung angeben.

4 Wenn Sie in das Feld **R/C/D/G** für eine bestimmte Spalte ein `D` eingeben und Enter drücken, wird ein Fenster angezeigt, in dem Sie einen anderen Standardwert als den Systemstandardwert für diese Spalte angeben können.

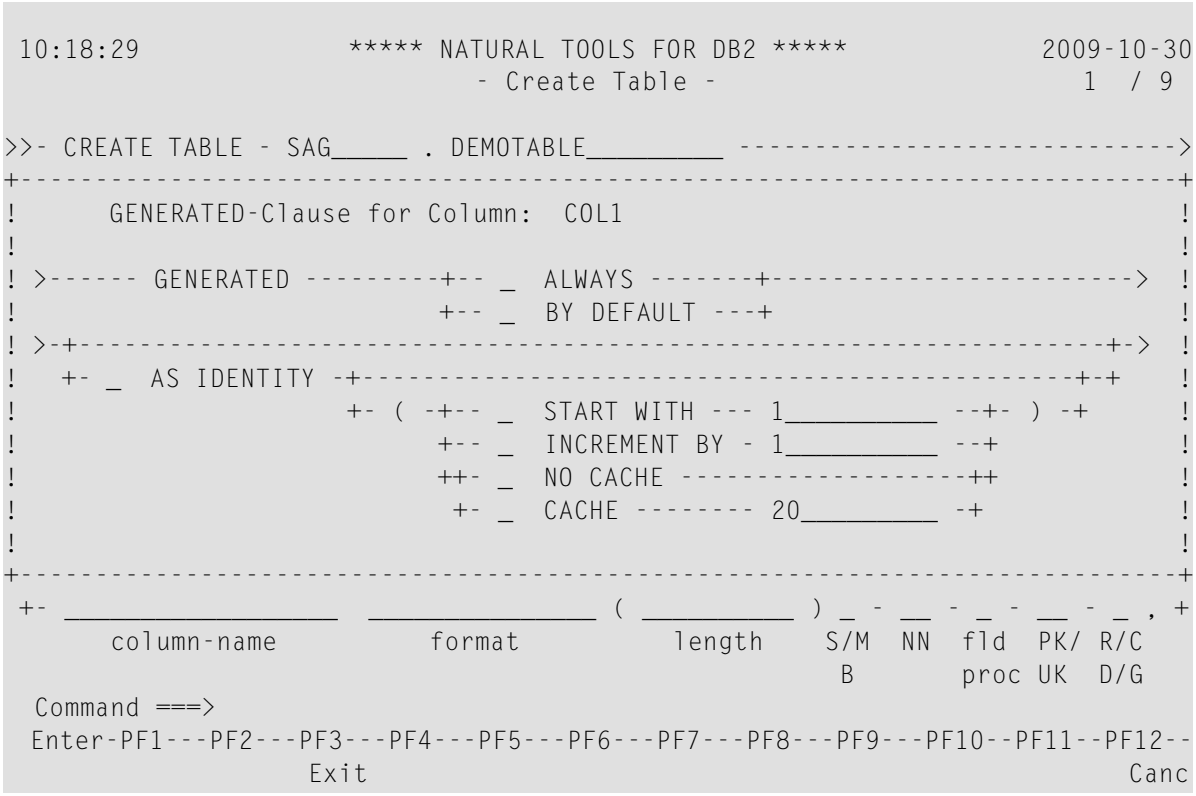
```
 10:14:04             ***** NATURAL TOOLS FOR DB2 *****            2009-10-30
                               - Create Table -                      1  / 9

+-------------------------------------------------------------------------------+
!      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
```

Es kann einer der folgenden Typen von Standardwerten angegeben werden:

- `USER`: ein Ausführungszeitwert des speziellen Registers `USER`.
- `CURRENT SQLID`: die SQL-Berechtigungskennung.
- `NULL`: der Nullwert.
- `constant`: eine Konstante, die den Standardwert für die Spalte benennt.

Weitere Informationen zu Standardwerten finden Sie in der entsprechenden Db2-Literatur von IBM.

5 Wenn Sie in das Feld **R/C/D/G** für eine bestimmte Spalte ein `G` eingeben und Enter drücken, wird ein Fenster angezeigt, in dem Sie die `GENERATED`-Klausel für diese Spalte definieren können.



`GENERATED` kann nur definiert werden, wenn die Spalte einen `ROWID`-Datentyp hat (oder einen `distinct type`, der auf einem `ROWID`-Datentyp basiert), oder wenn die Spalte eine Identity-Spalte sein soll.

Weitere Informationen zur *GENERATED-Clause* finden Sie in der entsprechenden Db2-Literatur von IBM.

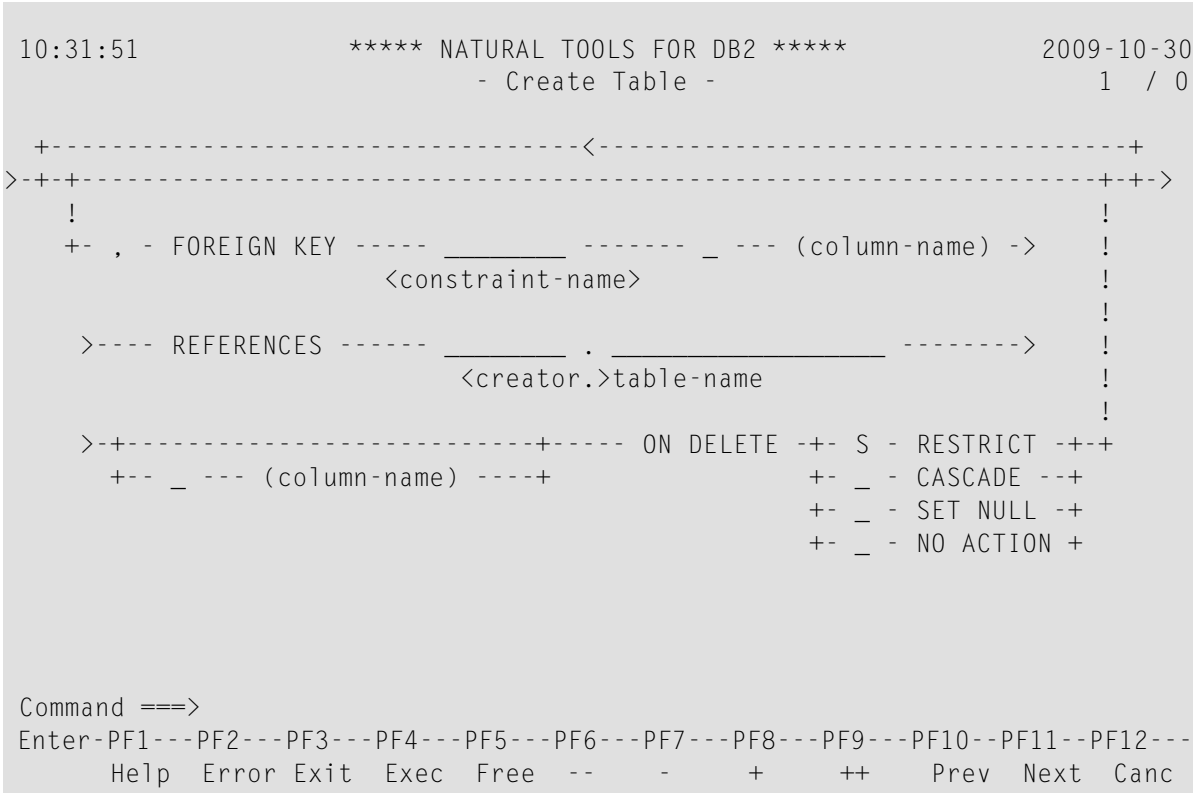
6 Fenster wie das folgende unterstützen Sie bei der Auswahl der richtigen Spalte. Sie werden durch Eingabe des Hilfezeichens (?) in das entsprechende Feld auf dem Bildschirm aufgerufen:

```
 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
```

Bei komplexen SQL-Statements kann mehr als ein Eingabebildschirm erforderlich sein. In diesem Fall können Sie durch Drücken von PF11 (Next) zum nächsten Bildschirm wechseln oder durch Drücken von PF10 (Prev) zum vorherigen Bildschirm zurückkehren.

Wie Sie auf dem obigen Bildschirm sehen können, ist der Beginn der Syntaxangabe für ein SQL-Statement immer mit >> gekennzeichnet.

7 Da die Syntax der `CREATE TABLE`-Anweisung ziemlich komplex ist, werden drei weitere Bildschirme benötigt. Nachdem alle erforderlichen Informationen auf dem ersten Bildschirm eingegeben wurden, drücken Sie PF11 (Next), um den nächsten **Create Table**-Eingabebildschirm anzuzeigen, auf dem Sie weitere optionale Parameter angeben können.



Auf diesem Bildschirm können Sie eine referenzielle Einschränkung (Constraint) auf eine andere Tabelle angeben. Geben Sie dazu ein `S` in das Feld **column-name** (Spaltenname) ein. Es wird eine Liste aller in der aktuellen Tabelle (abhängige Tabelle/Dependent Table) verfügbaren Spalten angezeigt, aus der Sie die Spalte(n) auswählen können, die den Fremdschlüssel in Bezug auf eine andere Tabelle (übergeordnete Tabelle/Parent Table) bilden sollen. Sie können auch einen Namen für die Einschränkung (`constraint-name`) angeben. Wenn nicht, wird der Name der Einschränkung von der ersten Spalte des Fremdschlüssels abgeleitet.

Ein Fremdschlüssel besteht aus einer oder mehreren Spalten in einer abhängigen Tabelle, die zusammen einen Wert annehmen müssen, der im Primärschlüssel der zugehörigen übergeordneten Tabelle existiert.

Im **REFERENCES**-Teil müssen Sie bei den Tabellennamen (mit optionalem Erstellernamen) der übergeordneten Tabelle angeben, die von der angegebenen Einschränkung betroffen sein soll. Darüber hinaus müssen Sie die Aktion angeben, die ausgeführt werden soll, wenn eine Zeile in der referenzierten übergeordneten Tabelle gelöscht wird.

Die folgenden Optionen sind verfügbar:

- `RESTRICT` oder `NO ACTION` verhindert die Löschung der übergeordneten Zeile, bis alle abhängigen Zeilen gelöscht sind.
- `CASCADE` bewirkt, dass alle abhängigen Zeilen ebenfalls gelöscht werden.

- SET NULL setzt alle Spalten des Fremdschlüssels in jeder abhängigen Zeile, die Nullwerte enthalten können, auf Null.

In der rechten oberen Ecke des Bildschirms wird der Index des aktuell angezeigten referentiellen Constraint-Blocks (1) und die Gesamtzahl der definierten referentiellen Constraint-Blöcke (0) angezeigt.

Wenn Sie alle Informationen eingegeben haben, können Sie entweder PF10 (Prev) drücken, um zum vorherigen Bildschirm zurückzukehren, oder PF11 (Next), um zum nächsten Bildschirm zu gelangen.

8 Auf dem nächsten Bildschirm haben Sie wieder die Möglichkeit, Spalten als eindeutig zu kennzeichnen. Diesmal können jedoch bis zu sechs Gruppen von eindeutigen Spalten definiert werden, mit bis zu 16 Spalten pro Gruppe. Die einzelnen Spalten werden in einem Fenster angegeben, das für jede Gruppe aufgerufen werden kann.

```
 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
```

Da eindeutige Spalten keine Nullwerte enthalten dürfen, wird automatisch ein weiteres Fenster aufgerufen, in dem Sie die als eindeutig angegebenen Spalten auch als NOT NULL definieren können (es sei denn, Sie haben sie bereits auf dem ersten **Create Table**-Bildschirm als solche definiert).

Wenn Sie alle Informationen eingegeben haben, können Sie entweder PF10 (Prev) drücken, um zum vorherigen Bildschirm zurückzukehren, oder PF11 (Next), um zum letzten Syntaxeingabe-Bildschirm zu gelangen.

9 Auf dem letzten Syntaxeingabe-Bildschirm können Sie nun

- das Auslassen der aktuellen Tabelle (und auch der Datenbank und des Tabellenraums (Tablespace)), die diese Tabelle enthalten) einschränken.
- Prüfeinschränkung für die aktuelle Tabelle definieren.

  Um eine Prüfbeschränkung zu definieren, müssen Sie eine Tabellenprüfbedingung angeben. Eine Prüfbedingung ist eine Suchbedingung mit verschiedenen Einschränkungen, die in der entsprechenden Db2-Literatur von IBM beschrieben sind. Darüber hinaus können Sie einen Namen für die Prüfbeschränkung angeben.

```
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
```

Wenn Sie auf diesem Bildschirm PF10 (Prev) drücken, kehren Sie zum vorherigen Bildschirm zurück.

Wie Sie auf dem obigen Bildschirm sehen können, wird das Ende der Syntaxangabe für ein SQL-Statement immer durch >< angezeigt.

Eine aktive Hilfe, die aus Auswahllisten in Fenstern besteht, steht für alle Felder zur Verfügung, die auf bestehende Datenbankobjekte verweisen. Die Auswahllisten werden aufgerufen, indem entweder ein Stern (*) oder ein Teil eines Objektnamens, gefolgt von einem Stern, in das entsprechende Eingabefeld eingegeben wird.

Wenn Sie z.B. `D*` in das Feld `database-name` des obigen Bildschirms eingeben, erscheint ein Fenster, in dem Sie Ihre Auswahlkriterien überprüfen können. Wenn Sie Enter drücken, wird eine Liste aller Datenbanken angezeigt, deren Name mit `D` beginnt.

```
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  CDBPR06  !
  >------- 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
```

Innerhalb der Auswahlliste können Sie nach oben (PF6 / -- oder PF7 / -) oder unten (PF8 / + oder PF9 / ++) blättern und die gewünschte Datenbank auswählen. Der Name der ausgewählten Datenbank wird in das entsprechende Feld auf Ihrem Eingabebildschirm übernommen.

10 Wenn Sie alle Informationen eingegeben haben, können Sie entweder in den Free Mode wechseln (PF5) oder das erstellte Member direkt an Db2 zur Ausführung übergeben (PF4). Wenn die Ausführung erfolgreich ist, erscheint die Meldung

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

Wenn nicht, wird ein Fehlercode zurückgegeben.

Im Free Mode zeigt der folgende Bildschirm des Editors den generierten SQLCODE an:

```
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
```

Der Free Mode Editor ist eine angepasste Version des Software AG Editors. Er ist nahezu identisch mit dem interaktiven **ISQL - Input**-Bildschirm. Jedoch können im Free Mode keine `SELECT`-Statements abgesetzt werden.

Weitere Einzelheiten entnehmen siehe *Software AG Editor*-Dokumentation.

Tabellenraum erstellen - Funktion: Create Tablespace

Um die Funktion Create Tablespace aufzurufen:

1 Geben Sie im Bildschirm **Catalog Maintenance** den Code `CR` ein.

Geben Sie im Feld **Object** den Code `TS` ein und drücken Sie Enter.

Der erste Bildschirm zur Eingabe der Syntax von **Create Tablespace** wird angezeigt:

```
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 Wenn Sie alle erforderlichen Informationen eingegeben haben, drücken Sie PF11 (Next), um zum nächsten Bildschirm zu gelangen:

```
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
```

Auf dem zweiten Bildschirm zur Syntaxeingabe für **Create Tablespace** können Sie nun zusätzliche Buffer Pool-Namen sowie die Option `LOCKSIZE` mit der Klausel `LOCKMAX` angeben.

Wenn Sie ein `S` in das Feld `bufferpool-name` eingeben und Enter drücken, wird ein Fenster angezeigt, in dem Sie zusätzliche Buffer Pool-Namen angeben können.

Weitere Einzelheiten zu den Klauseln `COMPRESS`, `LOCKSIZE` und `LOCKMAX` finden Sie in der entsprechenden Db2-Literatur von IBM.

Tabelle ändern - Funktion: Alter Table

Das folgende Beispiel veranschaulicht die Verwendung des Bildschirms zur Syntaxeingabe für die Funktion **Alter Table**.

Um die Funktion Alter Table aufzurufen:

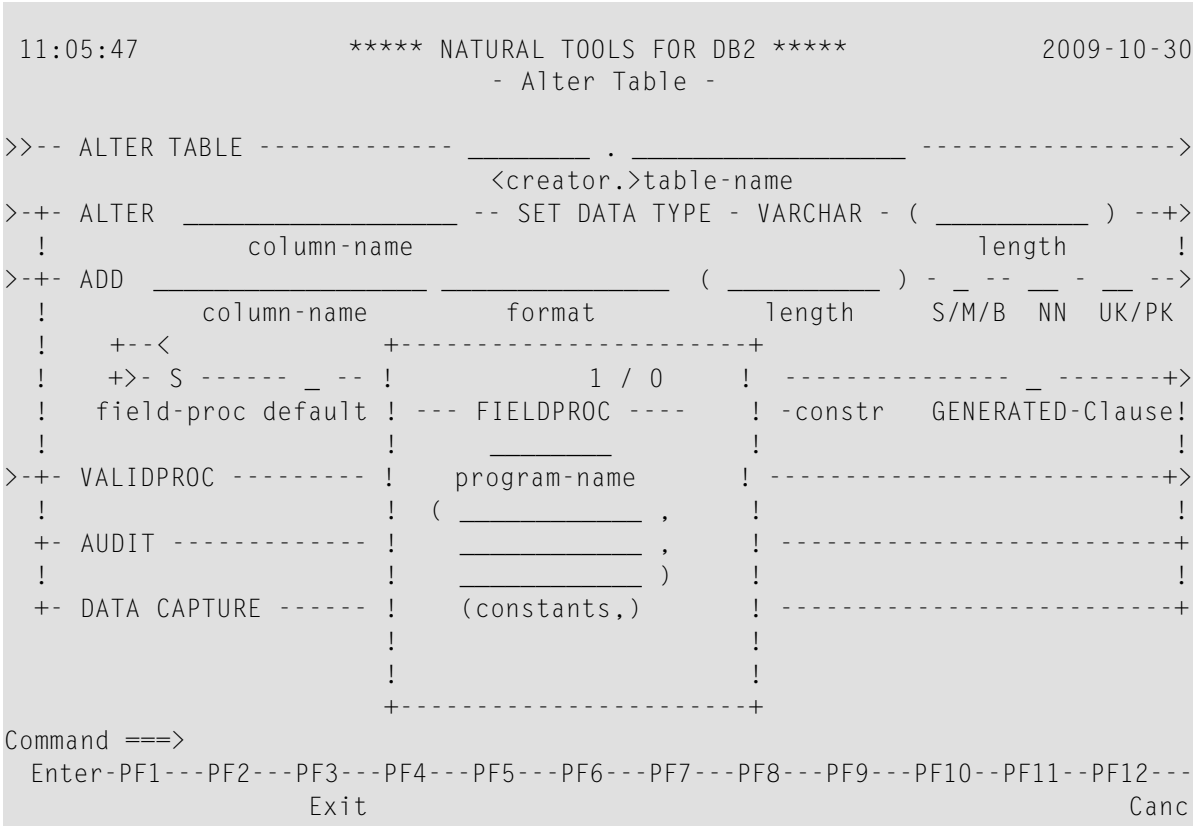
1 Geben Sie im Bildschirm **Catalog Maintenance** den Code `AL` ein.

Geben Sie im Feld **Object** den Code `TB` ein und drücken Sie Enter.

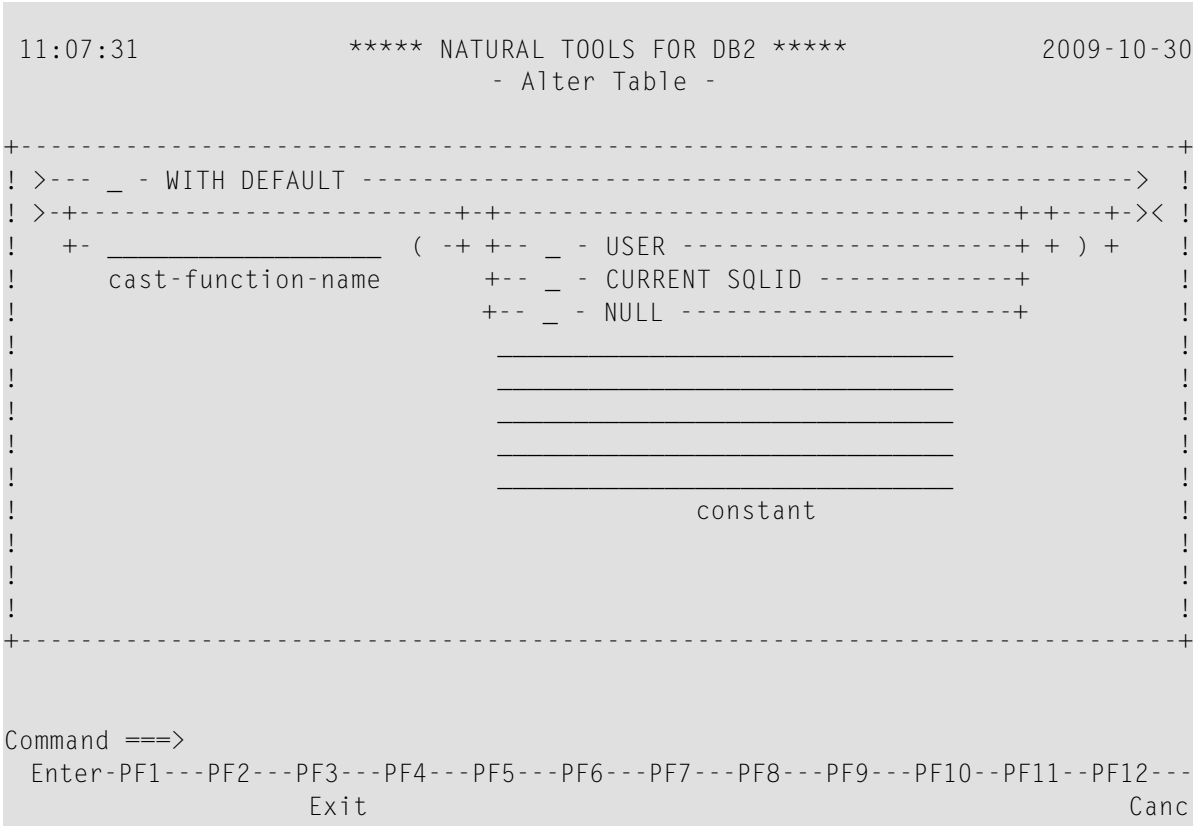
Es erscheint der Bildschirm **Alter Table**, auf dem sie Folgendes angeben können:

```
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 Wenn Sie im Eingabefeld `field-proc` ein `S` eingeben und Enter drücken, wird ein Fenster angezeigt, in dem Sie eine Feldprozedur angeben können, die für diese Spalte ausgeführt werden soll:



3 Wenn Sie ein `S` in das Feld `default` eingeben und Enter drücken, wird ein Fenster angezeigt, in dem Sie einen anderen Standardwert als den Systemstandardwert für diese Spalte angeben können:

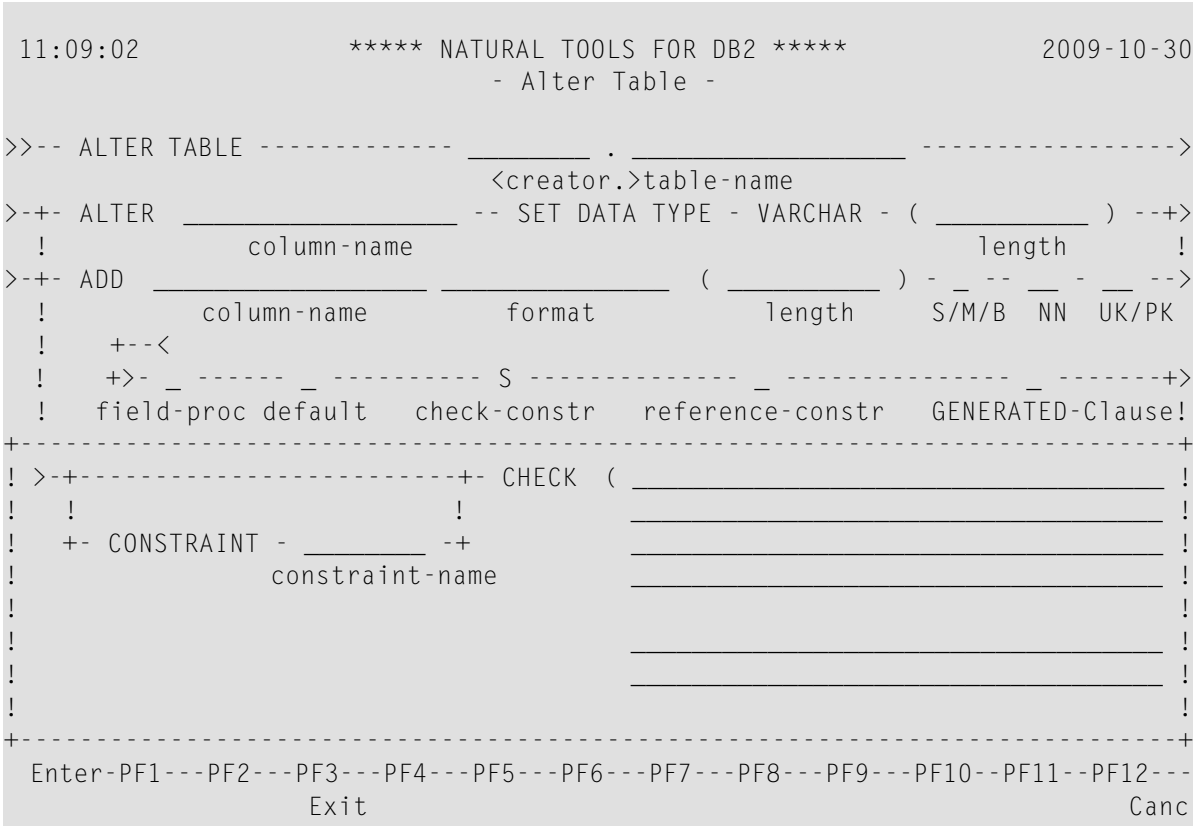


Es kann einer der folgenden Typen von Standardwerten angegeben werden:

- `USER`: ein Ausführungszeitwert des Spezialregisters `USER`.
- `CURRENT SQLID`: die SQL-Berechtigungskennung.
- `NULL`: der Nullwert.
- `constant`: eine Konstante, die den Standardwert für die Spalte benennt.

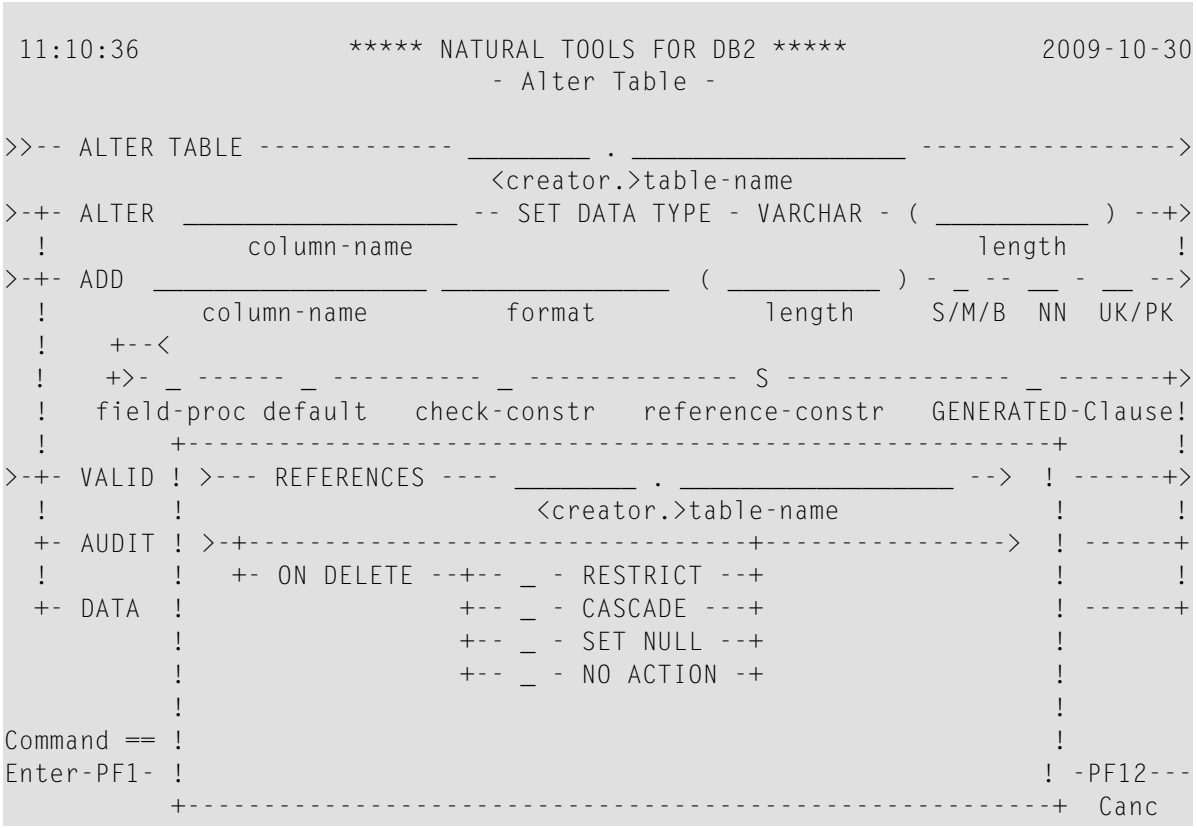
Weitere Informationen zu Standardwerten finden Sie in der entsprechenden Db2-Literatur von IBM.

4 Wenn Sie im Feld `check-constraint` ein `S` eingeben und Enter drücken, wird ein Fenster angezeigt, in dem Sie einen Prüfeinschränkung für diese Spalte angeben können:



Sie müssen eine Spaltenprüfbedingung angeben. Eine Prüfbedingung ist eine Suchbedingung mit verschiedenen Einschränkungen, die in der entsprechenden Db2-Literatur von IBM ausführlich beschrieben sind. Zusätzlich können Sie einen Namen für die Prüfeinschränkung angeben.

5 Wenn Sie im Feld `reference-constraint` ein `S` eingeben und Enter drücken, wird ein Fenster angezeigt, in dem Sie eine `references`-Klausel angeben können, die diese Spalte als Fremdschlüssel einer referentiellen Einschränkung (`reference-constr`) identifiziert:



Sie müssen den Namen (mit einem optionalen Erstellernamen) der übergeordneten Tabelle angeben, auf die verwiesen werden soll. Darüber hinaus müssen Sie die Aktion angeben, die ausgeführt werden soll, wenn eine Zeile in der referenzierten Tabelle gelöscht wird. Die folgenden Optionen sind verfügbar:

- `RESTRICT` oder `NO ACTION` verhindert die Löschung der übergeordneten Zeile, bis alle abhängigen Zeilen gelöscht sind.
- `CASCADE` löscht auch alle abhängigen Zeilen.
- `SET NULL` setzt alle Spalten des Fremdschlüssels in jeder abhängigen Zeile, die Nullwerte enthalten können, auf Null.

6 Wenn Sie Ihre Spaltendefinitionen eingegeben haben, drücken Sie PF11 (Next).

Es wird ein Bildschirm aufgerufen, auf dem Sie Primär- und/oder Fremdschlüssel hinzufügen oder weglassen können:

```
 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 Nachdem Sie die erforderlichen Informationen zum Hinzufügen und/oder Löschen von Primär- und/oder Fremdschlüsseln eingegeben haben, drücken Sie PF11 (Next). Es wird ein Bildschirm aufgerufen, auf dem Sie eine RESTRICT ON DROP-Klausel angeben, eine CHECK-Einschränkung (Constraint) hinzufügen oder löschen und/oder eine beliebige Einschränkung löschen können:

```
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
```

Tabellenraum ändern - Funktion: Alter Tablespace

Das folgende Beispiel veranschaulicht die Benutzung des Bildschirms zur Eingabe der Syntax für die Funktion **Alter Tablespace**.

Um die Funktion Alter Tablespace aufzurufen:

1 Geben Sie im Bildschirm **Catalog Maintenance** den Code `AL` ein.

Geben Sie im Feld **Object** den Code `TS` ein und drücken Sie Enter.

Sie gelangen auf den Bildschirm **Alter Tablespace**, auf dem Sie die folgenden Angaben machen können:

```
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 Wenn Sie ein `S` in das Feld `bufferpool-name` eingeben und Enter drücken, wird ein Fenster angezeigt, in dem Sie weitere Buffer Pool-Namen angeben können:

```
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 Wenn Sie sich wieder auf dem ersten Bildschirm für die Eingabe der **Alter Tablespace**-Syntax befinden, drücken Sie PF11 (Next), um zum nächsten Bildschirm zu gelangen:

```
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 Auf dem zweiten Bildschirm zur Eingabe der **Alter Tablespace**-Syntax können Sie nun die Optionen `LOCKMAX` und `LOCKSIZE` angeben.

Weitere Einzelheiten zu den Klauseln `COMPRESS`, LOCKSIZE und `LOCKMAX` finden Sie in der entsprechenden Db2-Literatur von IBM.

SQL Skeleton Members

SQL Skeleton Members werden für die Verarbeitung der folgenden SQL-Statements bereitgestellt, die von der **Catalog Maintenance**-Funktion nicht unterstützt werden:

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

Ein SQL Skeleton Member ist ein Natural-Text-Objekt, das ein SQL-Skelett enthält, das den Db2-SQL-Syntaxregeln entspricht, wie sie in der entsprechenden IBM-Literatur beschrieben sind. Die

in Kleinbuchstaben dargestellten ersetzbaren Elemente des SQL-Skeletts müssen mit Benutzereingaben gefüllt werden, damit das Skelett zu einem gültigen SQL-Statement wird, das im Free Mode (siehe Free Mode) oder ISQL (siehe **Interaktives SQL**) ausgeführt werden kann. Die Textobjekte des Skeletts werden zusammen mit Beispiel-SQL-Textobjekten in der Natural System Library SYSDB2 ausgeliefert.

7 Interaktives SQL

Mit der Funktion **Interactive SQL** der **Natural Tools for Db2** können Sie SQL-Statements dynamisch ausführen.

Funktion Interactive SQL aufrufen

Um die Funktion Interaktives SQL aufzurufen:

- Geben Sie im Menü **Natural Tools for DB2 Main Menu** den Funktionscode `I` ein.

  Der Bildschirm **Interactive SQL** wird angezeigt:

```
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
```

  Die folgenden Funktionen sind verfügbar:

Code	Beschreibung
I	Zeigt SQL-Member (Textobjekte) auf dem **Interactive SQL**-Eingabebildschirm an.
O	Zeigt Ausgabe-Member (Textobjekte) auf dem **Interactive SQL**-Ausgabebildschirm an.

  Die folgenden Parameter können angegeben werden:

Parameter	Beschreibung
Library	Gibt den Namen der aktuellen Natural Library an, die die angegebenen Eingabe-/Ausgabe-Members (Textobjekte) enthält. Die Angabe von Libraries, deren Namen mit SYS beginnen, ist nicht zulässig. Der Library-Name ist mit Ihrer Natural-Benutzerkennung voreingestellt.
Member	Wenn ein gültiger Member-Name angegeben ist, wird das entsprechende Member angezeigt. Wird ein Wert gefolgt von einem Stern (*) angegeben, werden alle Ein-/Ausgabe-Member in der aktuellen Bibliothek aufgelistet, deren Namen mit diesem Wert beginnen. Wird nur die Stern-Notation angegeben, wird eine Auswahlliste aller Ein-/Ausgabe-Member angezeigt. Wird das Feld **Member** leer gelassen, wird der leere SQL-Ein-/Ausgabebildschirm angezeigt.

SQL Input Members

Um die Funktion SQL Input Member aufzurufen:

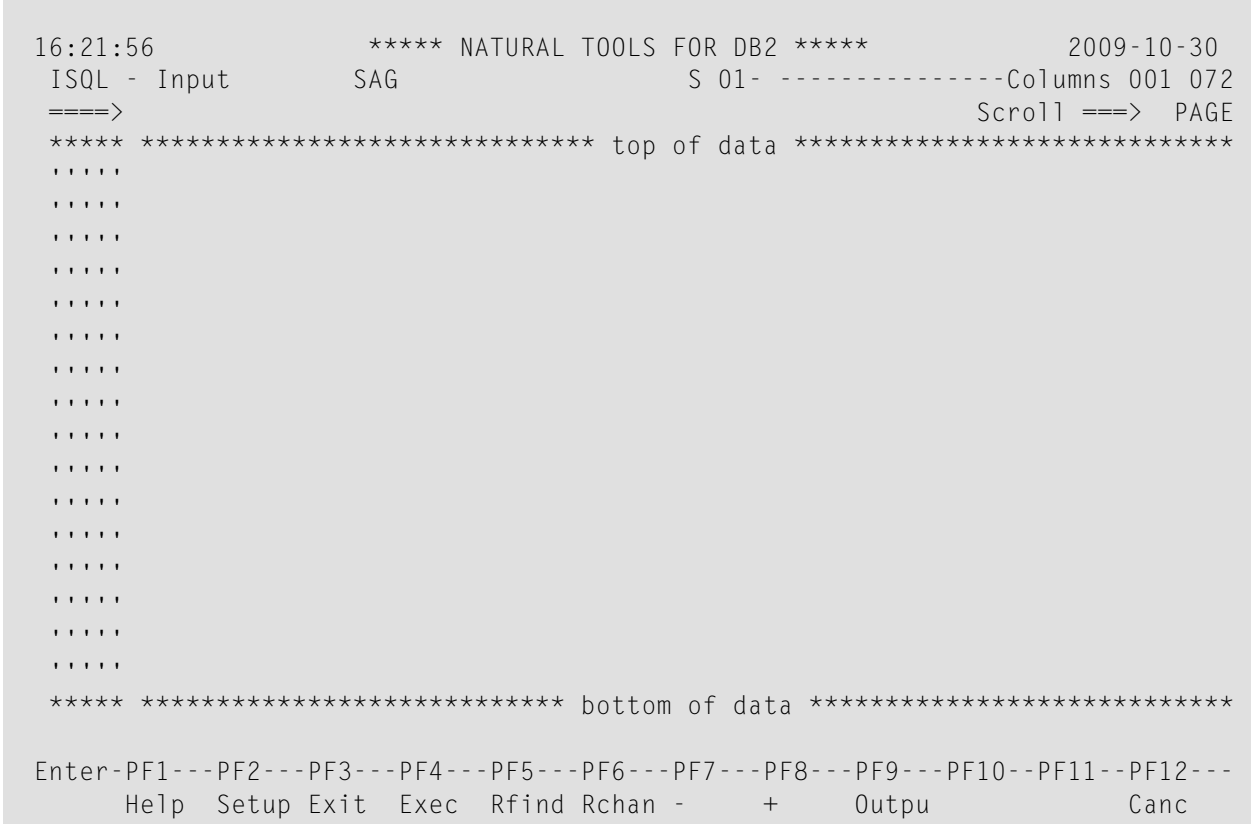
- Geben Sie auf dem Bildschirm **Interactive SQL** den Funktionscode `I` ein und drücken Sie Enter.

  Je nachdem, welchen Member-Namen (Textobjekt-Namen) Sie angegeben haben, werden unterschiedliche Bildschirme angezeigt.

  Diese Bildschirme werden in den folgenden Abschnitten erläutert.

ISQL Input-Bildschirm

Wenn Sie das Feld **Member** leer lassen, wird der leere Bildschirm **ISQL - Input** aufgerufen:



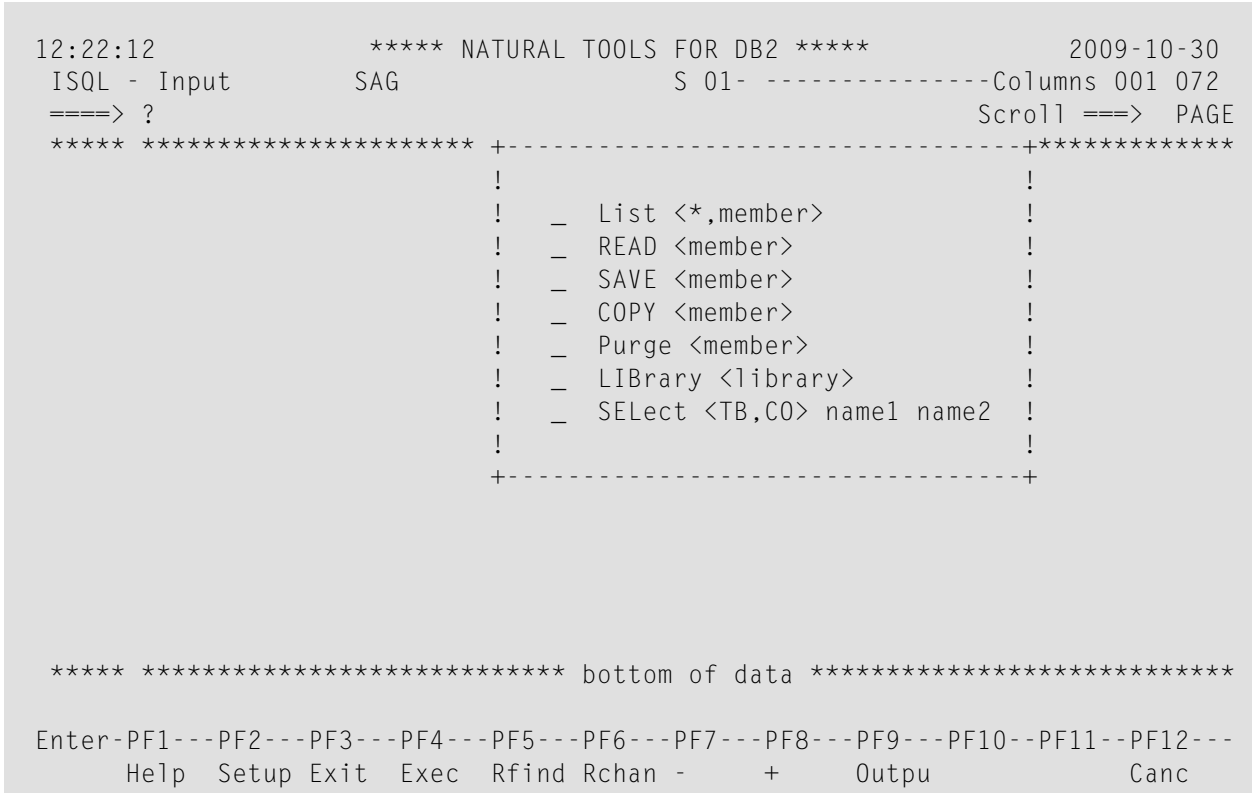
Der **ISQL - Input**-Bildschirm ist ein Free Mode-Editor (siehe *Mit den Natural Tools for Db2 editieren*), der eine ähnliche Funktionalität wie der Software AG Editor bietet. Mit dem Editor können Sie SQL-Statements über Editor-Hauptkommandos und -Zeilenkommandos eingeben oder bearbeiten. Sie können die SQL-Statements sofort aus dem Editor heraus ausführen, indem Sie PF4 (Exec) drücken, oder Sie können sie als SQL-Member (Textobjekt) in einer Natural Library für eine spätere Ausführung speichern.

Informationen zu den verfügbaren PF-Tasten finden Sie unter *PF-Tastenbelegungen*.

Anmerkung: Das `PRINT`-Kommando ist auf dem **ISQL - Input**-Bildschirm nicht verfügbar.

Neben den Haupt- und Zeilenkommandos des Editors stehen auch SQLCODE- Verwaltungskommandos zur Verfügung, um SQL-Member in einer Natural Library zu pflegen; siehe *Globale Verwaltungskommandos*. Mit diesen Verwaltungskommandos können Input-Member aufgelistet, abgerufen, in einer Natural Library gespeichert, kopiert und gelöscht werden. Sie werden in der Kommandozeile des Bildschirms eingegeben.

Eine Liste der verfügbaren Verwaltungskommandos erhalten Sie auch, wenn Sie das Hilfezeichen, d.h. ein Fragezeichen (`?`), in die Kommandozeile des Bildschirms eingeben. Es wird ein Fenster angezeigt, in dem das gewünschte Kommando ausgewählt werden kann. Der Inhalt des Fensters kann durch Drücken von PF8 vorwärts oder durch Drücken von PF7 rückwärts durchblättert werden.



Um Sie bei der Programmierung Ihres SQL-Members zu unterstützen, können vorhandene Db2-Tabellen und Spalten mit dem `SELECT`-Kommando aufgelistet werden. Von dieser Liste aus können Sie Tabellen- und Spaltennamen in den Editor einfügen.

Das `SELECT`-Kommando ist für die Auswahl von Tabellen und Spalten verfügbar:

Kommando	Beschreibung
`SELECT TABLE [creator.]name`	Wählt alle Tabellen mit dem angegebenen Ersteller (optional) und Namen aus. Sowohl für *creator* (Ersteller) als auch für *name* können Sie einen Wert, gefolgt von einem Stern (*), angeben. Dann werden alle Tabellen, deren Namen mit diesem Wert beginnen, ausgewählt. . Wenn Sie nur Stern-Notation angeben, werden alle vorhandenen Tabellen ausgewählt. Wenn Sie einen Tabellennamen ohne Ersteller angeben, werden alle Tabellen mit dem angegebenen Namen ausgewählt, unabhängig von ihrem Ersteller.
`SELECT COLUMN creator.name`	Selektiert alle Spalten der Tabelle *creator.name*. Da die Tabelle eindeutig identifiziert werden muss, kann keine Stern-Notation verwendet werden.

Beispiel eines Eingabebildschirms mit Fenster für die Tabellenauflistung

```
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 !                                                                       !
     +-------------------------------------------------------------------------+
```

In der Tabellenübersicht können Sie eine Tabelle zur Anzeige ihrer Spalten auswählen, indem Sie sie mit `C` vor dem Tabellennamen markieren. Die Spalten einer Tabelle werden zusammen mit ihrem Typ und ihrer Länge aufgelistet. Ein Ersteller- oder Tabellenname, der länger als 32 Zeichen ist, wird abgeschnitten. Dies wird durch ein > Symbol am Ende des Ersteller- oder Tabellennamens angezeigt.

Beispiel eines Eingabebildschirms mit einem Fenster zur Spaltenauflistung

```
 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 +-----------------------------------------------------+
```

Wenn Sie Tabellen- oder Spaltennamen aus einer Auswahlliste in den Editor übernehmen wollen, markieren Sie die entsprechende Tabelle oder Spalte mit `M`, wie auf dem vorherigen Bildschirm gezeigt. Die Tabellen- bzw. Spaltennamen werden entweder nach oder vor die mit `A` bzw. `B` markierte Zeile oder an den Anfang der angezeigten Daten kopiert.

Beispiel eines Eingabebildschirms mit kopierten Spaltennamen

```
 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 +-------------------------------------------------+
```

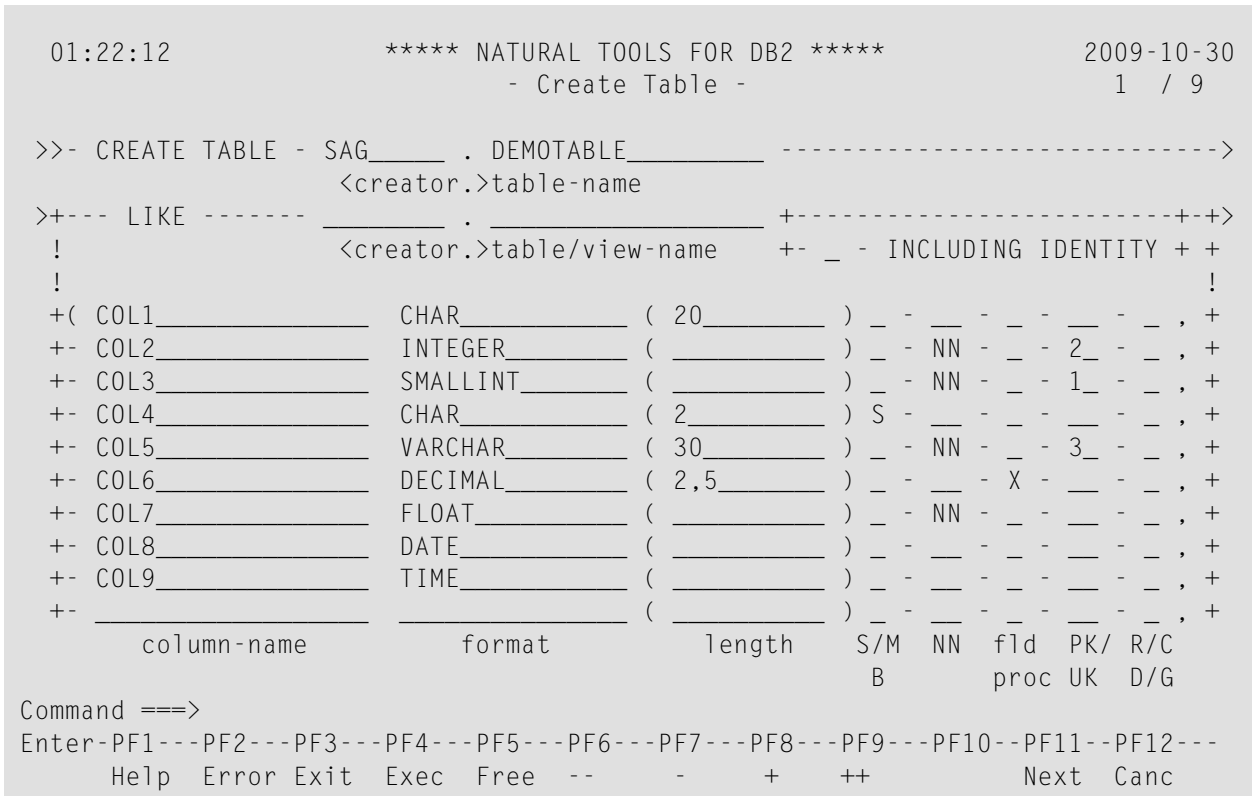
Fixed Mode mit Interaktivem SQL

Alle Fixed Mode-Eingabebildschirme aus dem **Catalog Maintenance**-Teil der **Natural Tools for DB2** sind als Hilfemasken im Interactive SQL-Teil verfügbar.

Um diese Hilfe aufzurufen, geben Sie in der Kommandozeile Ihres **ISQL - Input**-Bildschirms den Namen des gewünschten SQL-Statements ein, z.B. `CREATE TABLE` oder `CR TB` für das Kommando `CREATE TABLE`.

Es gelten die gleichen Kommandoabkürzungen wie bei der Funktion **Catalog Maintenance**.

Wenn Sie `CREATE TABLE` oder `CR TB` eingeben, wird der Bildschirm **Create Table** aufgerufen:



Wenn Sie Daten für ein vollständiges SQL-Statement eingegeben haben, können Sie aus den eingegebenen Daten ein SQL-Statement generieren und in den **ISQL - Input**-Bildschirm einfügen.

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

Mit PF4 (Incl) binden Sie den generierten SQLCODE ein. Sie bleiben auf dem Bildschirm **Create Table**.

Mit PF5 (IBack) binden Sie den generierten SQLCODE ein und kehren zum Bildschirm **ISQL - Input** zurück.

Ein SQL-Member abrufen

Wenn Sie im Feld **Member** des Bildschirms Interactive SQL einen eindeutigen Member-Namen (Textobjekt) angeben, wird das entsprechende SQL-Member auf dem Eingabebildschirm gelistet (angezeigt). Wenn kein Member mit dem angegebenen Namen existiert, wird eine entsprechende Meldung ausgegeben.

Beispiel für die Anzeige eines SQL-Member auf dem Eingabebildschirm

```
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
```

Gelistete SQL-Member können gelöscht, geändert, ausgeführt oder gespeichert werden.

Ein Stern (*) vor einer Statement-Zeile macht diese Zeile zu einer Kommentarzeile, was bedeutet, dass der entsprechende SQLCODE bei der Ausführung nicht berücksichtigt wird.

Liste der SQL-Member

Wenn Sie im Feld **Member** des Bildschirms **Interactive SQL** einen Wert gefolgt von einem Stern (*) angeben, wird eine Liste aller SQL-Eingabe-Member (Textobjekte) in der aktuellen Library angezeigt, deren Name mit diesem Wert beginnt.

Wenn Sie einen Stern (*) angeben, wird eine Liste aller SQL-Eingabe-Member in der aktuellen Library angezeigt.

Beispiel einer Auswahlliste für SQL-Eingabe-Member

```
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
```

Aus der Auswahlliste des Eingabebildschirms können Sie SQL-Member zur Anzeige auswählen, indem Sie sie mit einem `S` markieren.

Wenn die Liste durch ein `PURGE`-Kommando aufgerufen wurde, können die Member gelöscht werden, indem sie mit einem `P` markiert werden.

Durch Drücken von PF11 (>) können Sie von der Standardansicht des Bildschirms **Select Member**, wie oben gezeigt, zur erweiterten Ansicht wechseln, in der die erste Zeile jedes Member in der Spalte **Description** angezeigt wird:

```
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
```

Die erste Zeile eines Member kann die erste Zeile einer SQL-Statements oder eine Kommentarzeile sein, die weitere Informationen über das Member enthält.

Data Output Members

Um die Funktion Data Output Member aufzurufen:

- Geben Sie auf dem Bildschirm **Interactive SQL** den Funktionscode `0` ein und drücken Sie Enter.

  Je nachdem, welchen Member-Namen (Textobjekt) Sie angegeben haben, werden unterschiedliche Bildschirme angezeigt

  Diese Bildschirme werden in den folgenden Abschnitten erläutert.

Bildschirm für die Datenausgabe

Wenn Sie das Feld **Member** auf dem Bildschirm **Interactive SQL** leer lassen, wird der leere Bildschirm **ISQL - Output** aufgerufen.

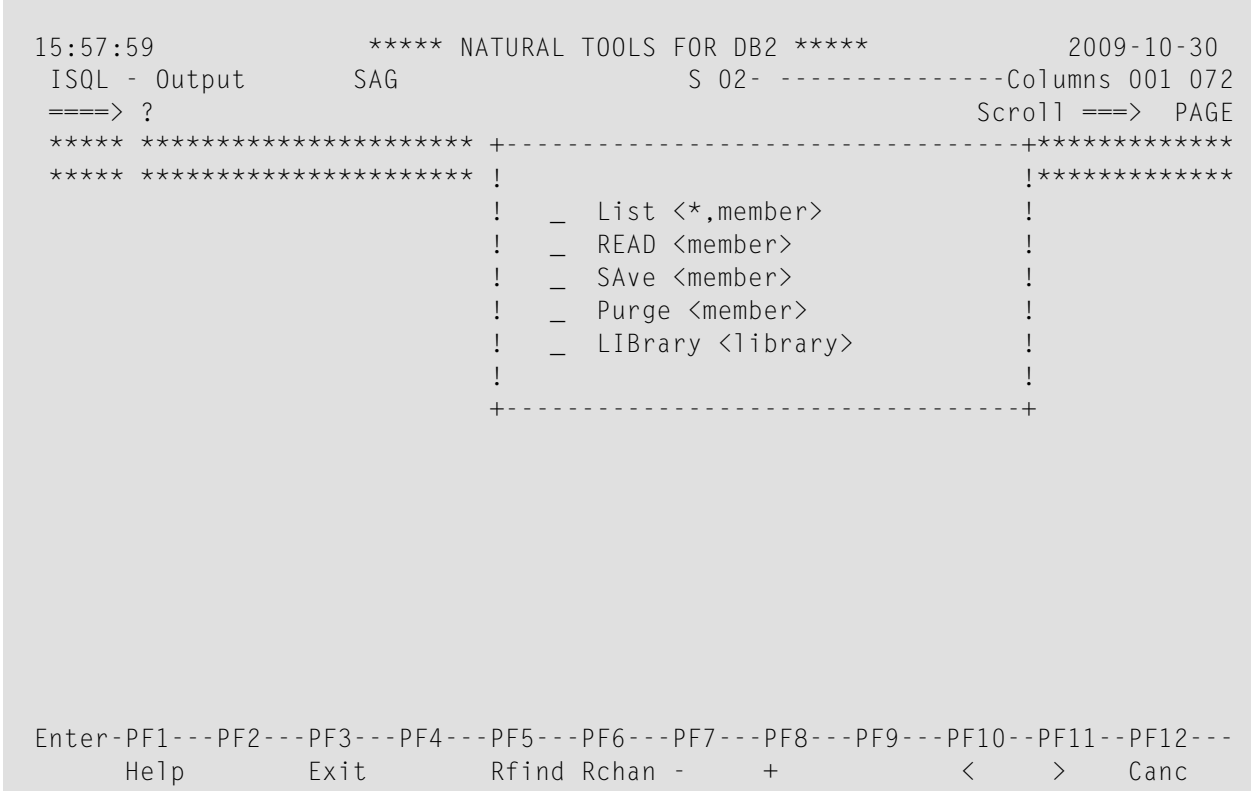
```
15:19:15              ***** NATURAL TOOLS FOR DB2 *****             2009-10-30
 ISQL - Output       SAG                  S 02- ---------------Columns 001 072
 ====>                                                        Scroll ===>  PAGE
 ***** ***************************** top of data ****************************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









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

Vom Bildschirm für die Datenausgabe aus haben Sie nur Zugriff auf die Ausgabedaten-Member. Bei den Ausgabedaten handelt es sich um Daten, die als Ergebnis von ausgeführten SQL-Statements aus der Datenbank abgerufen werden. Diese Daten können durchsucht und zur späteren Verwendung als Ausgabe-Member in der Natural-Systemdatei FUSER gespeichert werden. Zusätzlich zu den aus der Datenbank abgerufenen Daten enthalten die Ausgabe-Member auch Db2-Statusinformationen und das ausgeführte SQL-Member.

Wenn Sie ein SQL-Statement ausführen, werden die Ergebnisse automatisch auf dem Ausgabebildschirm angezeigt. Sie können den interaktiven SQL-Ausgabebildschirm also auch durch Ausführen eines SQL-Statements vom Eingabebildschirm aus aufrufen. Vom Ausgabebildschirm können Sie durch Drücken von PF3 (Beenden) zum Eingabebildschirm zurückkehren.

Informationen zu den anderen verfügbaren PF-Tasten siehe *PF-Tastenbelegungen*.

Die für Ausgabe-Member verfügbaren Verwaltungskommandos können auch in einem Fenster angezeigt und ausgewählt werden, siehe *Globale Verwaltungskommandos*. Das Fenster wird durch Eingabe des Hilfezeichens, d.h. eines Fragezeichens (?), in der Kommandozeile des Ausgabebildschirms aufgerufen.



Abgesehen von den Verwaltungskommandos stehen nur Blätter-Kommandos zur Verfügung (siehe *Mit den Natural Tools for Db2 editieren*), da die Ausgabe-Member nicht geändert werden können. Sowohl Blätter- als auch Verwaltungskommandos werden in der Kommandozeile des Ausgabebildschirms eingegeben.

Wenn ein Ausgabe-Member zu groß ist, um auf den Bildschirm zu passen, können Sie das Kommando `FIX ON n` verwenden, um die ersten *n* Zeichen beim Blättern nach links oder rechts auf dem Bildschirm zu behalten.

Ein Ausgabe-Member abrufen

Wenn Sie einen eindeutigen Member-Namen im Feld **Member** des **Interactive SQL**-Bildschirms angeben, wird das entsprechende Output Member auf dem Bildschirm aufgelistet. Wenn kein Member mit dem angegebenen Namen existiert, wird eine entsprechende Meldung zurückgegeben.

Beispiel für ein auf dem Output-Bildschirm aufgelistetes Ausgabe-Member

```
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
```

Ausgabe-Member auflisten

Wenn Sie im Feld **Member** des Bildschirms **Interactive SQL** einen Wert gefolgt von einem Stern (*) angeben, wird eine Liste aller Datenausgabe-Member in der aktuellen Library angezeigt, deren Namen mit diesem Wert beginnen.

Wenn Sie nur die Stern-Notation angeben, wird eine Liste aller Datenausgabe-Member in der aktuellen Library angezeigt.

Beispiel einer Auswahlliste für Datenausgabe-Member

```
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
```

Aus der Auswahlliste der Ausgabe-Member können Sie Ausgabe-Member zur Anzeige auswählen, indem Sie diese mit einem `S` markieren.

Wurde die Liste durch ein `PURGE`-Kommando aufgerufen, können die Member durch Markierung mit einem `P` gelöscht werden.

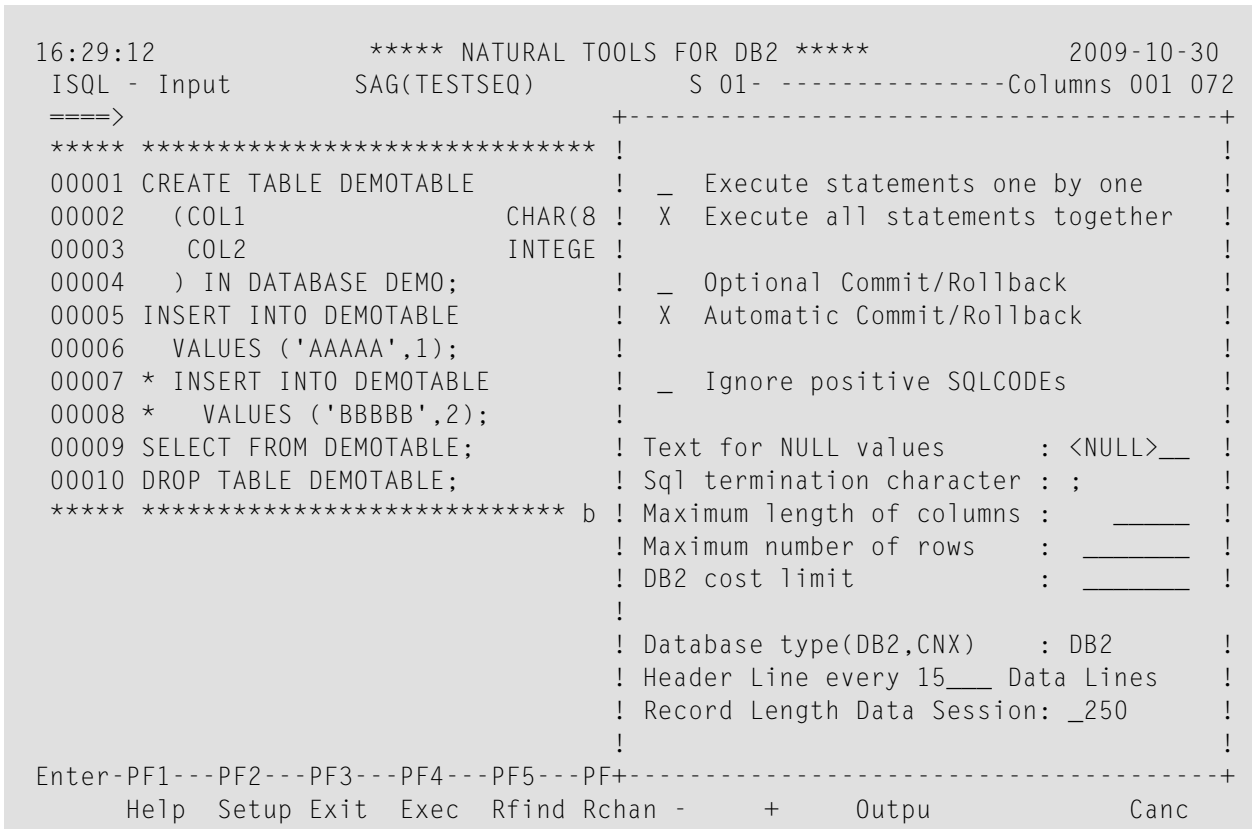
SQL-Statements verarbeiten

Auf SQL-Eingabe-Member (Textobjekte) kann nur über den Bildschirm **ISQL - Input** zugegriffen werden. Sie werden durch Drücken von PF4 (Exec) vom Eingabebildschirm aus gegen Db2 ausgeführt.

Nach der Ausführung erscheint der Bildschirm zur Datenausgabe, der die Ergebnisse des ausgeführten SQL-Members enthält.

Besteht ein SQL-Member aus mehreren SQL-Statements, müssen die einzelnen Statements durch ein Semikolon getrennt werden. Sie können einzeln oder alle zusammen zur selben Zeit ausgeführt werden.

Zur Auswahl der Ausführungsform steht ein Fenster zur Verfügung, das durch Drücken von PF2 (Setup) aufgerufen werden kann.



Nachstehend finden Sie Informationen zu den angebotenen Optionen:

- Statements der Reihe nach ausführen
- Alle Statements zusammen ausführen
- Automatisches Festschreiben/Zurücknehmen
- Optionales Festschreiben/Zurücknehmen
- Text für NULL-Werte
- SQL-Beendigungszeichen
- Maximale Länge der Spalten
- Maximale Anzahl der Zeilen
- Db2-Kostengrenze
- Kopfzeile alle n Datenzeilen

- Datensatzlänge pro Datensitzung

Statements der Reihe nach ausführen

(Execute Statements One By One)

Nach jedem SQL-Statement wird der Bildschirm für die Ausgabe angezeigt. Vom Ausgabebildschirm aus können Sie entweder das nächste SQL-Statement vom Eingabebildschirm aus ausführen, indem Sie PF4 (Next) drücken, oder die restlichen SQL-Statements überspringen und sofort zum Eingabebildschirm zurückkehren, indem Sie PF3 (Exit) drücken.

Alle Statements zusammen ausführen

(Execute All Statements Together)

Alle Statements werden sofort nacheinander ausgeführt. Auf dem Ausgabebildschirm werden die Ergebnisse aller Statements gemeinsam angezeigt.

Statements, die Cursornamen, Hostvariablen oder Parametermarkierungen enthalten, können mit interaktivem SQL nicht ausgeführt werden. Ebenfalls nicht ausgeführt werden Statements, die nur als eingebettetes SQL verfügbar sind, d.h. Statements, deren Funktionen automatisch von Natural ausgeführt werden.

Diese Statements sind:

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

Automatisches Festschreiben/Zurücknehmen

(Automatic Commit/Rollback)

Wenn Sie **Automatic Commit/Rollback** wählen, wird jede Änderung der Datenbank automatisch entweder festgeschrieben oder zurückgenommen, je nachdem, ob alle beteiligten SQL-Statements erfolgreich ausgeführt wurden. Wenn ja, wird ein `SQL COMMIT WORK`-Kommando ausgeführt. Wenn nicht, werden mit einem `SQL ROLLBACK`-Kommando alle Datenbankänderungen seit dem letzten Commit-Punkt rückgängig gemacht.

Optionales Festschreiben/Zurücknehmen

(Optional Commit/Rollback)

Wenn Sie **Optional Commit/Rollback** wählen, wird nach jedem SQL-Statement ein Fenster eingeblendet, in dem Sie die Möglichkeit haben, die auf dem Bildschirm angezeigten Datenbankänderungen entweder festzuschreiben oder zurückzunehmen.

Anmerkung: Da unter CICS und IMS TM jede Terminal-Ein-/Ausgabe zu einem `SYNCPOINT` führt, kommt die optionale Commit/Rollback-Funktion nur in einer TSO-Umgebung zur Anwendung.

In allen Umgebungen können Sie auch `SQL COMMIT`- und `ROLLBACK`-Kommandos in Ihr Eingabe-Member aufnehmen. Unter CICS und IMS TM werden diese Kommandos jedoch in die entsprechenden TP-Monitor-Aufrufe übersetzt.

Text für NULL-Werte

(Text For NULL Values)

Der Text, der bei `NULL`-Werten angezeigt werden soll, kann hier angegeben werden. Die Standardzeichenfolge ist `---`.

SQL-Beendigungszeichen

(SQL Termination Character)

Wenn Sie mehrere SQL-Statements eingeben, müssen diese voneinander getrennt werden. Das Standardzeichen für den Abschluss eines Statements ist das Semikolon (;).

Maximale Länge der Spalten

(Maximum Length of Columns)

Begrenzt die Länge für eine einzelne Spalte auf *n* Zeichen. Diese Begrenzung gilt nur für Zeichendaten. `DATE`-, `TIME`- oder `NUMERIC`-Spalten werden nicht abgeschnitten. Der Wert `0` zeigt an, dass es keine Begrenzung gibt.

Maximale Anzahl der Zeilen

(Maximum Number of Rows)

Begrenzt die Anzahl der Zeilen, die von einem `SELECT`-Statement zurückgegeben werden. Der Wert `0` zeigt an, dass es keine Begrenzung gibt.

Db2-Kostengrenze

(DB2 Cost Limit)

Legt ein Limit für die Db2-Kalkulation fest. `SELECT`-Statements, die dieses Limit überschreiten, werden nicht ausgeführt. Der Wert `0` zeigt an, daß kein Limit existiert.

Kopfzeile alle n Datenzeilen

(Header Line Every *n* Data Lines)

Für `SELECT`-Statements können Sie festlegen, dass alle *n* Datenzeilen eine Kopfzeile mit den Namen der ausgewählten Spalten eingefügt wird. Wenn *n* auf `0` gesetzt wird, wird nur eine Kopfzeile am Anfang der Daten angezeigt.

Datensatzlänge pro Datensitzung

(Record Length Data Session)

Die Satzlänge (*n*) für die Ausgabesitzung kann angegeben werden. Wenn die angegebene Satzlänge kleiner ist als die Satzlänge der Ausgabedaten, werden die Ausgabesätze entsprechend abgeschnitten. Das Abschneiden von Datensätzen wird durch ein Größer-als-Zeichen (>) als ganz linkes Zeichen in der ersten Zeile unter jeder Kopfzeile angezeigt. Der Standardwert für *n* ist 250 Bytes.

PF-Tastenbelegungen

Die folgenden PF-Tasten-Einstellungen gelten für den Bildschirm **ISQL - Input**:

Taste	Belegung	Funktion
PF2	Setup	Ruft ein Fenster mit weiteren Verarbeitungsoptionen auf.
PF4	Exec	Führt das SQL-Member (Textobjekt) aus, das sich gerade auf dem Bildschirm befindet.
PF5	Rfind	Wiederholt das zuletzt ausgeführte `FIND`-Kommando.
PF6	Rchan	Wiederholt das zuletzt ausgeführte `CHANGE`-Kommando.
PF7	-	Blättert die Anzeige eine Seite zurück.
PF8	+	Blättert die Anzeige um eine Seite vorwärts.
PF9	Outpu	Ruft die Auswahlliste der Ausgabe-Member (Textobjekte) direkt aus dem Eingabebildschirm heraus auf.

Abgesehen von PF2 (Setup), PF4 (Exec) und PF9 (Outpu) gelten die gleichen PF-Tasten-Einstellungen auch für den Bildschirm **ISQL - Output**. Darüber hinaus sind die folgenden PF-Tastenbelegungen verfügbar:

Taste	Belegung	Funktion
PF4	Next	Führt das nächste SQL-Statement aus, wenn ein SQL-Member aus mehr als einem Statement besteht und Sie die Ausführung der einzelnen Statements nacheinander gewählt haben. Ist dies nicht der Fall, bleibt die Taste PF4 unbelegt.
PF10	<	Blättert die Anzeige des Bildschirms nach links.
PF11	>	Blättert die Anzeige des Bildschirms nach rechts.

Interaktive SQL-Ergebnisse entladen

Die Ergebnisse von interaktivem SQL werden entladen und in ein Dataset mit dem DD-Namen `CMWKF01` im Batch-Modus mit dem Befehl `UNLDDATA` geschrieben.

`CMWKF01` sollte ein variables Satzformat haben. Die Satzlänge hängt von der Größe des SQL-Ausgabe-Members (Textobjekt) ab und kann zwischen 250 und 4000 Byte betragen.

Um die Ergebnisse von interaktivem SQL zu entladen:

1 Melden Sie sich an der Natural Library `SYSDB2` an.

2 Geben Sie in der Kommandozeile das Kommando `UNLDDATA` ein und drücken Sie Enter.

Das Menü **Unload SQL Results** wird angezeigt:

```
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
```

Die folgende Funktion ist verfügbar:

Code	Beschreibung
U	Entlädt die Ergebnisse der interaktiven SQL-Ausführung.

Es gelten die folgenden Parameter:

Parameter	Beschreibung
Library	Gibt den Namen der Natural Library an, aus der die angegebenen Ausgabe-Member entladen werden sollen. Sie können keine Libraries angeben, deren Namen mit SYS beginnen. Dieser Parameter muss angegeben werden.
Member	Gibt den/die Namen des/der zu entladenden Ausgabe-Member an. Dieser Parameter muss angegeben werden.

8 Systemtabellen abrufen

⚠ **Wichtig:** Bevor Sie die Funktion **Retrieval of System Tables** verwenden: Lesen Sie den Abschnitt *LISTSQL and Explain Functions* unter *Special Requirements for Natural Tools for DB2* in der *Installing Natural for DB2 on z/OS*-Dokumentation.

Die Db2-Systemtabellen liefern Informationen über den Inhalt Ihres Db2-Systems. Mit der Funktion **Retrieval of System Tables** können Sie:

- Informationen über Db2-Objekte anzeigen, ohne SQL-Abfragen zu programmieren,
- auf einfache Weise auf verwandte Objekte, wie z.B. die Indexe einer Tabelle, zugreifen.

Die von der Funktion **Retrieval of System Tables** unterstützten Db2-Objekte sind Database, Tablespace, Tables, Index, Column, Plan, Check Constraints, Statistic Tables, Package und DBRM (Database Request Module) sowie die Zugriffsrechte auf und die Beziehungen zwischen diesen Objekten.

Db2-Objekte werden auf eine der beiden folgenden Arten dargestellt:

- Als Auswahllisten, in denen alle Objekte vom gleichen Typ sind und in denen Kommandos abgesetzt werden können, um verwandte Objekte anzuzeigen.
- Sie können Datenbanken, Tabellen, Pläne und Packages nach Namen auflisten:
  - Von den Datenbanklisten aus können Sie Listen der Tablespaces oder Tabellen einer Datenbank aufrufen.
  - Von der Tabellenliste aus können Sie eine Liste der Spalten und Indexe einer Tabelle aufrufen.
  - Über die Planliste können Sie die DBRMs eines Plans, die Package-Liste eines Plans, die von einem Plan verwendeten Tabellen und Indexe sowie die für einen Plan aktivierten oder deaktivierten Systeme auflisten.
  - Von der Package-Liste aus können Sie Listen der in einem Paket verwendeten Tabellen und Indexe und der Systeme aufrufen, die für ein Package aktiviert oder deaktiviert sind.
  - Von den Datenbank-, Tabellen-, Plan- oder Paketlisten aus können Sie auch ermitteln, wer zum Zugriff auf ein Db2-Objekt berechtigt ist.

  Außerdem können Sie über das Menü **User Authorization** alle bestehenden Zugriffsrechte nach Benutzerkennungen auflisten.
- Als Reports, die lediglich Informationen über verschiedene Typen von Db2-Objekten enthalten und in denen nur Browse-Kommandos ausgegeben werden können.

Die wichtigsten Browse-Kommandos können auch über PF-Tasten abgesetzt werden, siehe ***Mit den Natural Tools for Db2 editieren***.

In diesem Kapitel werden die folgenden Themen behandelt:

Funktion Retrieval of System Tables aufrufen

Um die Funktion Retrieval of System Tables aufzurufen:

- Geben Sie im **Natural Tools for DB2 Main Menu** den Funktionscode `R` ein.

  Der Bildschirm **Retrieval of System Tables** wird angezeigt:

```
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
```

  Mit PF2 (Setup) kann die maximale Länge einer Spalte und die Anzahl der Festzeichen beim Blättern nach links festgelegt werden. Die Standardwerte für beide Parameter können im Subprogramm `CONFIG` in der Library SYSDB2 geändert werden.

  Wenn ein Spaltenwert länger als die maximale Länge ist, wird er abgeschnitten und wie folgt gekennzeichnet:

  - mit einem Größer-als-Zeichen (>) im Falle von Zeichenketten, die am rechten Ende abgeschnitten werden, oder
  - einem Kleiner-als-Zeichen (<) im Falle von Zahlen, die am linken Ende abgeschnitten werden.

Beachten Sie, dass für weitere Kommandos in einer Zeile, z. B. das Zeilenkommando `I`, nur der sichtbare Wert als Eingabe genommen werden kann. Dies bedeutet, dass Kommandos in Zeilen fehlschlagen, wenn die Werte für die weitere Verarbeitung abgeschnitten werden.

```
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
```

Die folgenden Funktionen sind verfügbar:

Code	Beschreibung
D	Listet die im Db2-Katalog definierten Datenbanken auf.
K	Listet die im Db2-Katalog definierten Pakete auf.
P	Listet die im Db2-Katalog definierten Pläne auf.
S	Statistiktabellen.
T	Listet die im Db2-Katalog definierten Tabellen auf.
U	Informiert, welche(r) Benutzer auf welche Db2-Objekte zugreifen darf.

Die folgenden Parameter müssen als Auswahlkriterien angegeben werden:

Parameter	Beschreibung
`Database Name`	Der Name der Datenbank, die aufgelistet werden soll. Stern-Notation (*) zur Angabe eines Bereichs ist möglich. Der Parameter `Database Name` ist nur für die Funktion **List Databases** relevant.
`Package Collection`	Die Sammlung des aufzulistenden Package. Stern-Notation (*) für die Bereichsangabe ist möglich. Der Parameter `Package Collection` ist nur für die Funktion **List Packages** relevant.
`Package Name`	Der Name des aufzulistenden Pakets. Stern-Notation (*) für die Bereichsangabe ist möglich. Der Parameter `Package Name` ist nur für die Funktion **List Packages** relevant.
`Plan Name`	Der Name des aufzulistenden Plans. Stern-Notation (*) für die Bereichsangabe ist möglich. Der Parameter `Plan Name` ist nur für die Funktion **List Plans** relevant.
`Table Creator`	Der Name des Erstellers der aufzulistenden Tabelle(n). Stern-Notation (*) für die Bereichsangabe ist möglich. Der Parameter `Table Creator` ist nur für die Funktion **List Tables** relevant.
`Table Name`	Der Name der aufzulistenden Tabelle. Stern-Notation (*) für die Bereichsangabe ist möglich. Der Parameter `Table Name` ist nur für die Funktion **List Tables** relevant.

Datenbanken auflisten - Funktion: List Databases

Um die Funktion List Databases aufzurufen:

1 Geben Sie im **Retrieval of System Tables** den Funktionscode `D` ein.

2 Geben Sie den Namen der Datenbank(en) an, die aufgelistet werden sollen.

- Wenn ein Wert gefolgt von einem Stern angegeben wird, werden alle im Db2-Katalog definierten Datenbanken aufgelistet, deren Namen mit diesem Wert beginnen.
- Wenn nur ein Stern angegeben wird, werden alle im Db2-Katalog definierten Datenbanken aufgelistet.

```
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
```

Die folgenden Zeilenkommandos sind auf dem Bildschirm mit der Datenbankauflistung verfügbar. Die Zeilenkommandos werden in den Felder vor der/den gewünschten Datenbank(en) eingegeben:

Kommando	Beschreibung
I	Zeigt Informationen zu einer Datenbank an.
S	Wählt eine Datenbank aus, die mit den Hauptkommandos (siehe unten) verwendet werden soll.
U	Macht die Auswahl einer Datenbank rückgängig.
AU	Zeigt Informationen über die Zugriffsrechte auf eine Datenbank an.
TB	Zeigt alle in einer Datenbank definierten Tabellen an.
TS	Zeigt alle in einer Datenbank definierten Tablespaces an.

Die als Ergebnis des Kommandos TB oder TS angezeigten Auflistungen von Tabellen oder Tablespaces können für die weitere Bearbeitung benutzt werden, während der Inhalt der als Ergebnis des Kommandos AU oder I angezeigten Bildschirme nur zu Informationszwecken dient.

Eine Liste aller Zeilenkommandos, die bei der Funktion **List Database** zur Verfügung stehen, kann als Fenster aufgerufen werden, indem das Hilfezeichen, d.h. ein Fragezeichen (?), vor einer der aufgelisteten Datenbanken eingegeben wird.

Die Kommandos `AU`, `TB` und `TS` können auch als Hauptkommandos verwendet werden. Hauptkommandos werden in der Kommandozeile des Bildschirms mit der Datenbankliste eingegeben und gelten für alle zuvor mit dem Zeilenkommando `S` ausgewählten Datenbanken.

Ein weiteres Hauptkommando ist das `INFO`-Kommando, das dem Zeilenkommando `I` entspricht, aber Informationen über alle zuvor ausgewählten Datenbanken anzeigt. Alle Informationen, die sich aus den Kommandos `I` oder `INFO` ergeben, können nicht nur angezeigt, sondern auch zum Drucken markiert werden. Auch wenn die Informationen bereits angezeigt werden, können sie mit dem Kommando `PRINT` gedruckt werden.

```
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
```

Eine Liste aller Hauptkommandos, die in der Funktion **List Database** zur Verfügung stehen, kann als Fenster aufgerufen werden, indem das Hilfezeichen, d.h. ein Fragezeichen (`?`), in die Kommandozeile des Bildschirms **List Database** eingegeben wird.

Tablespaces auflisten - Funktion: List Tablespaces

Die Funktion zum Auflisten von Tablespaces (Tabellenräume) ist nicht Teil des Hauptmenüs **Retrieval of System Tables**.

Um Tablespaces aufzulisten:

- Geben Sie das Kommando `TS` nur auf dem Bildschirm mit der Datenbankliste ein.

  Es wird z.B. ein Bildschirm mit einer Tablespace-Liste angezeigt, zum Beispiel:

```
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      BP0           4          0       1       0 A
 __ DB2DEMO  EMPLOYEE    SAG      BP0           4          0       1       0 A
 ** ***************************** bottom of data ******************************
```

  Die folgenden Zeilenkommandos sind auf dem Bildschirm mit der Tablespace-Liste verfügbar. Die Zeilenkommandos werden in den Feldern vor dem/den gewünschten Tablespace(s) eingegeben:

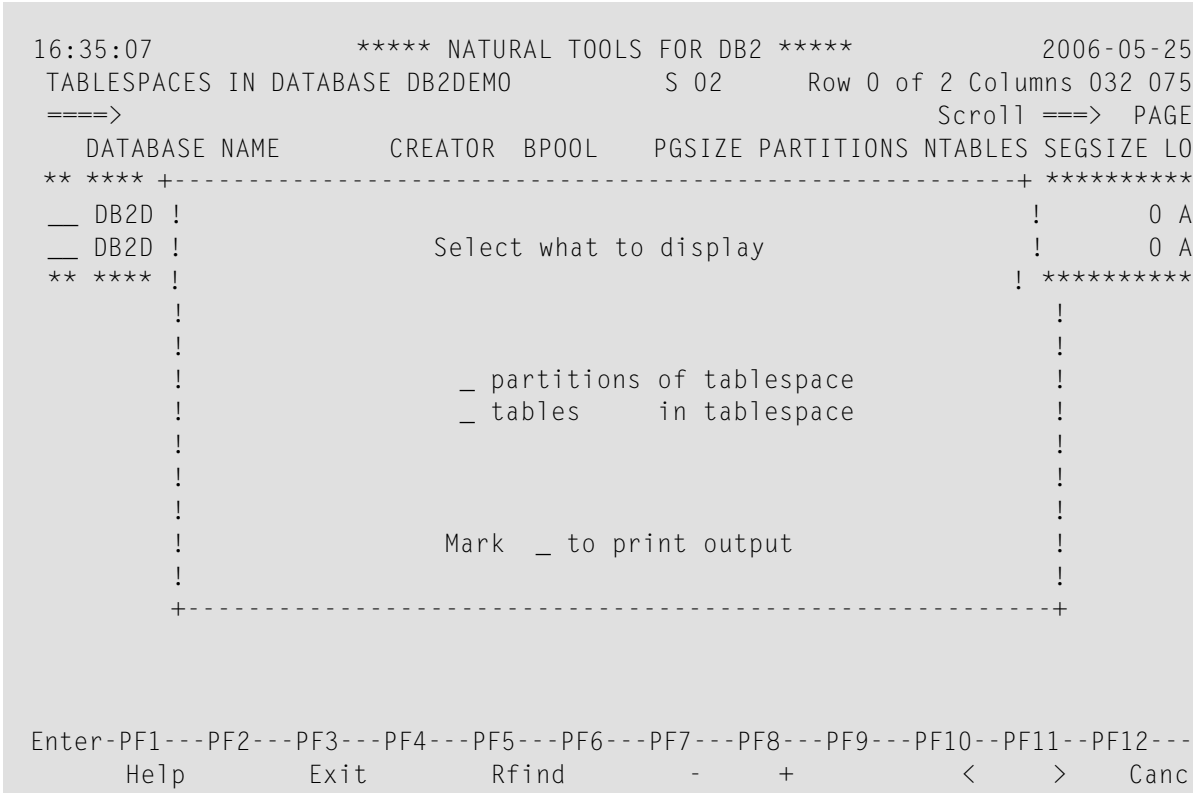
Kommando	Beschreibung
`I`	Zeigt Informationen zu einem Tablespace an.
`S`	Wählt einen Tablespace aus, der mit Hauptkommandos verwendet werden soll.
`U`	Macht die Auswahl eines Tablespace rückgängig.
`PT`	Zeigt alle Partitionen eines Tablespaces an.
`TB`	Zeigt alle in einem Tablespace definierten Tabellen an.

  Die mit dem Kommando `TB` angezeigten Listen mit Tabellen können für die weitere Bearbeitung verwendet werden, während die mit den Kommandos `I` und `PT` angezeigten Listen nur zu Informationszwecken dienen.

  Eine Liste aller auf dem Bildschirm zur Auflistung von Tablespaces verfügbaren Zeilenkommandos kann als Fenster aufgerufen werden, indem das Hilfezeichen, d. h. ein Fragezeichen (`?`), vor einem der aufgelisteten Tablespaces eingegeben wird.

  Die Kommandos `PT` und `TB` können auch als Hauptkommandos verwendet werden, die in der Kommandozeile des Bildschirms für die Tablespace-Liste eingegeben werden. Hauptkommandos gelten für alle Tablespaces, die zuvor mit dem Zeilenkommando `S` ausgewählt wurden.

Ein weiteres Hauptkommando ist das Kommando `INFO`, das dem Zeilenkommando `I` entspricht, aber Informationen über alle zuvor ausgewählten Tablespaces anzeigt. Alle Informationen, die sich aus dem Kommando `I` oder `INFO` ergeben, können nicht nur angezeigt, sondern auch zum Drucken markiert werden. Auch wenn die Informationen bereits angezeigt werden, können sie mit dem Kommando `PRINT` gedruckt werden.



Eine Liste aller Hauptkommandos, die auf dem Bildschirm für die Auflistung von Tablespaces verfügbar sind, kann als Fenster aufgerufen werden, indem das Hilfezeichen, d.h. ein Fragezeichen (`?`), in die Kommandozeile des Bildschirms eingegeben wird.

Pläne auflisten - Funktion: List Plans

Um die Funktion List Plans aufzurufen:

- Geben Sie im Bildschirm **Retrieval of System Tables** den Funktionscode `P` ein.

  Der Name des/der aufzulistenden Plans/Pläne muss angegeben werden.

  - Wird ein Wert gefolgt von einem Stern (*) angegeben, werden alle im Db2-Katalog definierten Pläne aufgelistet, deren Name mit diesem Wert beginnt.

- Wenn nur ein Stern (*) angegeben wird, werden alle im Db2-Katalog definierten Pläne aufgelistet.

Drücken Sie 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
```

Erlaubte Kommandos in Plänen

Die folgenden Zeilenkommandos sind auf dem Bildschirm mit der Planliste verfügbar. Die Zeilenkommandos werden in den Feldern vor dem/den gewünschten Plänen eingegeben:

Kommando	Beschreibung
I	Zeigt Informationen zu einem Plan an.
S	Wählt einen Plan aus, der mit Hauptkommandos verwendet werden soll.
U	Macht die Auswahl eines Plans rückgängig.
AU	Zeigt Informationen zu den Zugriffsrechten auf einen Plan an.
DR	Zeigt alle in einem Plan enthaltenen DBRMs an.
IX	Zeigt alle Indexe an, die von einem Plan verwendet werden.
PK	Zeigt die Package-Liste eines Plans an.

Kommando	Beschreibung
SY	Zeigt die Systeme an, die für einen Plan aktiviert oder deaktiviert sind.
TB	Zeigt die in einem Plan verwendeten Tabellen an.

Die als Ergebnis des Kommandos DR, IX, PK oder TB angezeigte Auflistung kann zur weiteren Bearbeitung verwendet werden, während der Inhalt der als Ergebnis des Kommandos I, AU oder SY angezeigten Bildschirme nur zu Informationszwecken dient.

Eine Liste aller mit der Funktion **List Plans** verfügbaren Zeilenkommandos kann als Fenster aufgerufen werden, indem das Hilfezeichen (?) vor einem der aufgelisteten Pläne eingegeben wird.

Die Kommandos AU, DR, IX, PK, SY und TB können auch als Hauptkommandos verwendet werden, die in der Kommandozeile des Bildschirms eingegeben werden und sich auf alle zuvor mit dem Zeilenkommando S ausgewählten Pläne beziehen.

Das Hauptkommando INFO, das dem Zeilenkommando I entspricht, zeigt Informationen über die DBRMs und ihre SQL-Statements an, die in den zuvor ausgewählten Plänen enthalten sind. Wie bei der Funktion **List Database** können die Informationen, die aus den Kommandos I oder INFO resultieren, auch gedruckt werden.

```
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
 __ 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
```

Eine Liste aller Hauptkommandos, die mit der Funktion **List Plans** zur Verfügung stehen, kann als Fenster aufgerufen werden, indem das Hilfezeichen, d.h. ein Fragezeichen (?), in die Kommandozeile des Bildschirms eingegeben wird.

Datenbankanforderungsmodule eines Plans auflisten - Funktion: DBRMs of Plan

Wenn Sie auf dem Bildschirm **List Plan** das Kommando `DR` eingeben, wird eine Liste aller Datenbankanforderungsmodule (DBRMs) angezeigt, die in den/die ausgewählten Pläne eingebunden sind.

```
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 ******************************
```

Zulässige Kommandos für DBRMs

Die folgenden Zeilenkommandos sind auf dem Bildschirm mit der DBRM-Liste verfügbar. Zeilenkommandos werden vor dem/den gewünschten DBRM(s) eingegeben:

Kommando	Beschreibung
`I`	Zeigt Informationen zu einem DBRM an.
`S`	Wählt einen DBRM aus, der mit Hauptkommandos benutzt werden soll.
`U`	Macht die Auswahl eines DBRM rückgängig.

Eine Liste aller auf dem DBRM-Auflistbildschirm verfügbaren Zeilenkommandos kann als Fenster aufgerufen werden, indem das Hilfezeichen, d. h. ein Fragezeichen (`?`), vor einem der aufgelisteten DBRMs eingegeben wird.

Das einzige Hauptkommando, das für DBRMs gültig ist, ist das `INFO`-Kommando, das dem Zeilenkommando `I` entspricht, aber Informationen über alle zuvor ausgewählten DBRMs anzeigt. Alle Informationen, die sich aus dem Kommando `I` oder `INFO` ergeben, können nicht nur angezeigt, sondern auch zum Drucken markiert werden. Auch wenn die Informationen bereits angezeigt werden, können sie mit dem `PRINT`-Kommando gedruckt werden.

```
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
```

Im Plan verwendete Indexe

Wenn Sie im **Plan-** oder im **Tabellen-Auflistbildschirm** das Kommando `IX` eingeben, wird eine Liste aller Indexe angezeigt, die in dem/den ausgewählten Plänen oder Tabellen verwendet werden.

```
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 *****************************
```

Zulässige Kommandos für Indexe

Die folgenden Zeilenkommandos sind auf dem Bildschirm mit der Indexliste verfügbar. Die Zeilenkommandos werden vor dem/den gewünschten Index/Indexe eingegeben:

Kommando	Beschreibung
I	Zeigt Informationen zu einem Index an.
S	Wählt einen Index aus, der mit Hauptkommandos verwendet werden soll.
U	Macht die Auswahl eines Indexes rückgängig.
CO	Zeigt alle Spalten eines Indexes an.
PT	Zeigt die Partitionen eines Indexes an.

Die als Antwort auf das Kommando CO oder PT angezeigten Listen mit Spalten können nicht für die weitere Bearbeitung verwendet werden. Sie dienen, wie die Anzeige nach dem Kommando I, nur zu Informationszwecken.

Eine Liste aller auf dem Bildschirm für die Indexauflistung verfügbaren Zeilenkommandos kann als Fenster aufgerufen werden, indem das Hilfezeichen (?) vor einem der aufgelisteten Indexe eingegeben wird.

Die Kommandos CO und PT können auch als Hauptkommandos verwendet und in die Kommandozeile des Index-Auflistbildschirms eingegeben werden. In diesem Fall werden alle Spalten aller zuvor mit dem Zeilenkommando S ausgewählten Indexe angezeigt.

Ein weiteres Hauptkommando ist das INFO-Kommando, das dem Zeilenkommando I entspricht, aber Informationen zu allen zuvor ausgewählten Indexe anzeigt. Alle Informationen, die sich aus den Kommandos I oder INFO ergeben, können statt zur Anzeige auch für den Druck markiert werden. Auch wenn die Informationen bereits angezeigt werden, können sie mit dem Kommando PRINT gedruckt werden.

```
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
```

Eine Liste aller Hauptkommandos, die auf dem Bildschirm mit der Indexliste verfügbar sind, kann durch Eingabe des Hilfezeichens (`?`) in der Kommandozeile des Bildschirms als Fenster aufgerufen werden.

Package-Liste des Plans auflisten – Funktion: Package List for Plan

Wenn Sie auf dem Plan-Auflistbildschirm das Kommando `PK` eingeben, wird eine Liste aller Einträge in der Package-Liste des/der ausgewählten Plans/Pläne angezeigt.

```
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 ***************
```

Zulässige Kommandos für Package List-Einträge

Die folgenden Zeilenkommandos sind auf dem Package List-Bildschirm verfügbar. Die Zeilenkommandos werden vor dem gewünschten Package List-Eintrag eingegeben:

Kommando	Beschreibung
I	Zeigt Informationen zu einem Package List-Eintrag an.
S	Wählt einen Package List-Eintrag aus, der mit Hauptkommandos benutzt werden soll.
U	Macht die Auswahl eines Package List-Eintrags rückgängig.
PK	Zeigt alle Packages eines Package List-Eintrags an.

Die **Auflistung der Packages** als Ergebnis des Kommandos PK kann für die weitere Bearbeitung verwendet werden, während die Anzeige als Ergebnis des Kommandos I nur zu Informationszwecken dient.

Eine Liste aller bei einer Package List verfügbaren Zeilenkommandos kann als Fenster aufgerufen werden, indem das Hilfezeichen, d.h. ein Fragezeichen (?), vor einem der aufgelisteten Einträge eingegeben wird.

Das Kommando PK kann auch als Hauptkommando verwendet werden, das in der Kommandozeile des obigen Bildschirms eingegeben wird und für alle zuvor mit dem Zeilenkommando S ausgewählten Package List-Einträge gilt.

Packages auflisten - Funktion: List Packages

Um die Funktion List Packages aufzurufen:

- Geben Sie im Bildschirm **Retrieval of System Tables** den Funktionscode K ein.

  Sie können die Collection und den Namen des/der aufzulistenden Packages angeben.

  Wird ein Wert gefolgt von einem Stern (*) angegeben, werden alle im Db2-Katalog definierten Packages aufgelistet, deren Collections/Namen mit diesem Wert beginnen. Wenn nur ein Stern (*) angegeben wird, werden alle im Db2-Katalog definierten Packages aufgelistet.

  Drücken Sie 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  ? ?? 14927068OF77EO>SAG      SAG      SAG
 __ SAGQCATV SAGQVRAS    ?   ??=? 148C409B09097E>SAG      SAG      SAG
 __ SAGQCATV SAGQVREL    ?   ??yO 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
```

Zulässige Kommandos für Packages

Die folgenden Zeilenkommandos sind auf dem Bildschirm mit der Liste der Packages verfügbar. Die Zeilenkommandos werden vor dem/den gewünschten Package(s) eingegeben:

Kommando	Beschreibung
I	Zeigt Informationen zu einem Package an.
S	Wählt ein Package aus, das mit den Hauptkommandos verwendet werden soll.
U	Macht die Auswahl eines Pakets rückgängig.
AU	Zeigt Informationen zu den Zugriffsrechten auf ein Package an.
IX	Zeigt alle Indexe an, die von einem Package verwendet werden.
SY	Zeigt alle Systeme an, die für ein Package aktiviert oder deaktiviert sind.
TB	Zeigt alle von einem Package verwendeten Tabellen an.

Die Auflistungen von Indexen oder Tabellen, die als Ergebnis des Kommandos `IX` oder `TB` angezeigt werden, können für die weitere Bearbeitung verwendet werden, während die Anzeigen, die aus dem Kommando `AU`, `SY` oder `I` resultieren, nur zu Informationszwecken dienen.

Eine Liste aller Zeilenkommandos, die mit der Funktion **List Packages** zur Verfügung stehen, kann als Fenster aufgerufen werden, indem das Hilfezeichen, d.h. ein Fragezeichen (`?`), vor einem der aufgeführten Packages eingegeben wird.

Die Kommandos AU, IX, SY und TB können auch als Hauptkommandos verwendet werden, die in der Kommandozeile des Bildschirms für die Tabellenauflistung eingegeben werden und sich auf alle zuvor mit dem Zeilenkommando S ausgewählten Tabellen beziehen.

Das Hauptkommando INFO, das dem Zeilenkommando I entspricht, zeigt Informationen über alle zuvor ausgewählten Tabellen an. Alle Informationen, die sich aus den Kommandos I oder INFO ergeben, können auch gedruckt werden.

```
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
```

Eine Liste aller Hauptkommandos, die mit der Funktion **List Packages** zur Verfügung stehen, kann als Fenster aufgerufen werden, indem das Hilfezeichen, d.h. ein Fragezeichen (?), in die Kommandozeile des Bildschirms **List Packages** eingegeben wird.

Tabellen auflisten – Funktion: List Tables

Um die Funktion List Tables aufzurufen:

- Geben Sie im Bildschirm **Retrieval of System Tables** den Funktionscode T ein.

  Der Ersteller (Creator) und der Name der aufzulistenden Tabelle(n) können angegeben werden.

Wenn ein Wert gefolgt von einem Stern (*) angegeben wird, werden alle im Db2-Katalog definierten Tabellen aufgelistet, deren Ersteller/Name mit diesem Wert beginnt. Wenn nur ein Stern (*) angegeben wird, werden alle im Db2-Katalog definierten Tabellen aufgelistet.

Drücken Sie 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
```

Zulässige Kommandos für Tabellen

Die folgenden Zeilenkommandos sind auf dem Bildschirm mit der Tabellenliste verfügbar. Die Zeilenkommandos werden vor der/den gewünschten Tabelle(n) eingegeben:

Kommando	Beschreibung
I	Zeigt Informationen über eine Tabelle an.
S	Wählt eine Tabelle aus, die mit Hauptkommandos verwendet werden soll.
U	Macht die Auswahl einer Tabelle rückgängig.
AU	Zeigt Informationen über die Zugriffsrechte auf eine Tabelle an.
CO	Zeigt alle Spalten einer Tabelle an.
IX	Zeigt alle Indexe einer Tabelle an.
CC	Prüft auf Einschränkungen (Constraints).

Die mit dem Kommando `IX` angezeigten Indexlisten können für die weitere Bearbeitung verwendet werden, während die mit dem Kommando `CO` angezeigten Spaltenlisten sowie die mit dem Kommando `AU` oder `I` angezeigten Tabellen nur zu Informationszwecken dienen.

Eine Liste aller mit der Funktion **Tabellen auflisten** verfügbaren Zeilenkommandos kann als Fenster aufgerufen werden, indem das Hilfezeichen, d.h. ein Fragezeichen (`?`), vor einer der aufgeführten Tabellen eingegeben wird.

Die Kommandos `AU`, `CO` und `IX` können auch als Hauptkommandos verwendet werden, die in der Kommandozeile des Bildschirms eingegeben werden und sich auf alle zuvor mit dem Zeilenkommando `S` ausgewählten Tabellen beziehen.

Das Hauptkommando `INFO`, das dem Zeilenkommando `I` entspricht, zeigt Informationen über alle zuvor ausgewählten Tabellen an. Alle Informationen, die sich aus den Kommandos `I` oder `INFO` ergeben, können auch gedruckt werden.

```
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
```

Eine Liste aller Hauptkommandos, die mit der Funktion **List Tables** zur Verfügung stehen, kann als Fenster aufgerufen werden, indem das Hilfezeichen, d.h. ein Fragezeichen (`?`), in die Kommandozeile des Bildschirms **List Tables** eingegeben wird.

Benutzerberechtigungen anzeigen – Funktion: User Authorizations

Um die Funktion User Authorizations aufzurufen:

- Geben Sie im Bildschirm **Retrieval of System Tables** den Funktionscode `U` ein und drücken Sie Enter.

  Das Menü **Retrieval of User Authorizations** wird angezeigt:

```
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
```

  Die folgenden Funktionen sind verfügbar:

Code	Beschreibung
C	Zeigt die Spalten an, auf die der angegebene Berechtigte zugreifen kann.
D	Zeigt die Datenbanken an, auf die der angegebene Berechtigte zugreifen kann.
K	Zeigt die Packages an, auf die der angegebene Berechtigte zugreifen kann.
P	Zeigt die Pläne an, auf die der angegebene Berechtigte zugreifen kann.
R	Zeigt die Ressourcen an, auf die der angegebene Berechtigte zugreifen kann.
T	Zeigt die Tabellen an, auf die der angegebene Berechtigte zugreifen kann.
U	Zeigt die Systemberechtigungen des angegebenen Berechtigten an.

Der folgende Parameter muss angegeben werden:

Parameter	Beschreibung
Grantee	Es wird eine Liste aller vorhandenen Db2-Objekte des angegebenen Objekttyps angezeigt, auf die der angegebene Berechtigte Zugriff hat.

Statistik-Tabellen auflisten – Funktion: List Statistic Tables

Um die Funktion List Statistic Tables aufzurufen:

- Geben Sie im Bildschirm **Retrieval of System Tables** den Funktionscode S ein und drücken Sie Enter.

Das Menü **Retrieval of Statistic Tables** wird angezeigt:

```
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
```

Die folgenden Funktionen sind verfügbar:

Code	Beschreibung
C	Zeigt die partitionierten Statistiken für Spalten in einem partitionierten Tablespace an.
D	Zeigt die Verteilung der Werte der ersten Spalte eines partitionierten Indexes an.
I	Zeigt die Statistiken für einen partitionierten Index an.
T	Zeigt die Statistiken für einen partitionierten Tablespace an.

Die folgenden Parameter müssen angegeben werden:

Parameter	Beschreibung
Table Creator	Der Name des Erstellers der Tabelle, für die die Statistiken angezeigt werden sollen.
Table Name	Der Name der Tabelle, für die die Statistiken angezeigt werden sollen.
Index Owner	Der Name des Eigentümers des Indexes, für den die Indexstatistiken angezeigt werden sollen.
Index Name	Der Name des Indexes, für den die Indexstatistiken angezeigt werden sollen.

9 Environment Setting-Funktionen benutzen

Mit der **Environment Setting**-Funktionalität der **Natural Tools for Db2** können Sie spezielle SQL-Statements interaktiv eingeben.

Einzelheiten zu den in diesem Kapitel beschriebenen SQL-Statements finden Sie in der entsprechenden Db2-Literatur von IBM.

Menü Environment Setting aufrufen

Um das Menü Environment Setting aufzurufen:

- Geben Sie im **Natural Tools for DB2 Main Menu** den Funktionscode `S` ein und drücken Sie Enter.

  Der Bildschirm **Environment Setting** wird angezeigt.

```
 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 ===>
```

  Folgende Codes sind im Menü **Environment Setting** zur Angabe von SQL-Statements und Ausführung der entsprechenden Funktionen vorhanden:

Code	SQL-Statement	Siehe Funktion:
CO	CONNECT	Connect
RE	RELEASE	Release
SC	SET CONNECTION	Set Connection
SS	SET CURRENT SQLID	Set Current SQLID
SP	SET CURRENT PACKAGESET	Set Current Packageset
SD	SET CURRENT DEGREE	Set Current Degree
SU	SET CURRENT RULES	Set Current Rules
SO	SET CURRENT OPTIMIZATION HINT	Set Current Optimization Hint
SL	SET CURRENT LOCALE LC_CTYPE	Set Current Locale LC_CType
SA	SET CURRENT PATH	Set Current Path
SE	SET CURRENT PRECISION	Set Current Precision
SM	SET CURRENT MAINTAINED TABLE TYPE FOR OPTIMIZATION	Set Current Maintained Types for Optimization
SB	SET CURRENT PACKAGE PATH	Set Current Package Path
SF	SET CURRENT REFRESH AGE	Set Current Refresh Age
SH	SET CURRENT SCHEMA	Set Current Schema
SN	SET CURRENT APPLICATION ENCODING SCHEME	Set Current Application Encoding Scheme
SY	SET ENCRYPTION PASSWORD	Set Encryption Password
SR	Anzeige der aktuellen Werte der unterstützten speziellen Register.	Display Special Registers

Connect

Um die Funktion Connect aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode CO ein und drücken Sie Enter.

  Der Bildschirm **Connect** wird angezeigt:

```
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
```

Die Funktion **Connect** stellt eine Verbindung zwischen der aktuellen Anwendung und einem bestimmten Server her. Dieser Server ist der aktuelle Server, der im Feld **Current Server Version** angezeigt wird.

Auf dem Bildschirm **Connect** identifizieren Sie den aktuellen Server, indem Sie einen Standortnamen angeben. Der identifizierte Server muss dem lokalen Db2-Subsystem bekannt sein.

Release

Um die Funktion Release aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `RE` ein und drücken Sie Enter.

  Der Bildschirm **Release** wird angezeigt:

```
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
```

Die Funktion **Release** versetzt eine oder mehrere Verbindungen in den Status „Release Pending“ (Freigabe anstehend).

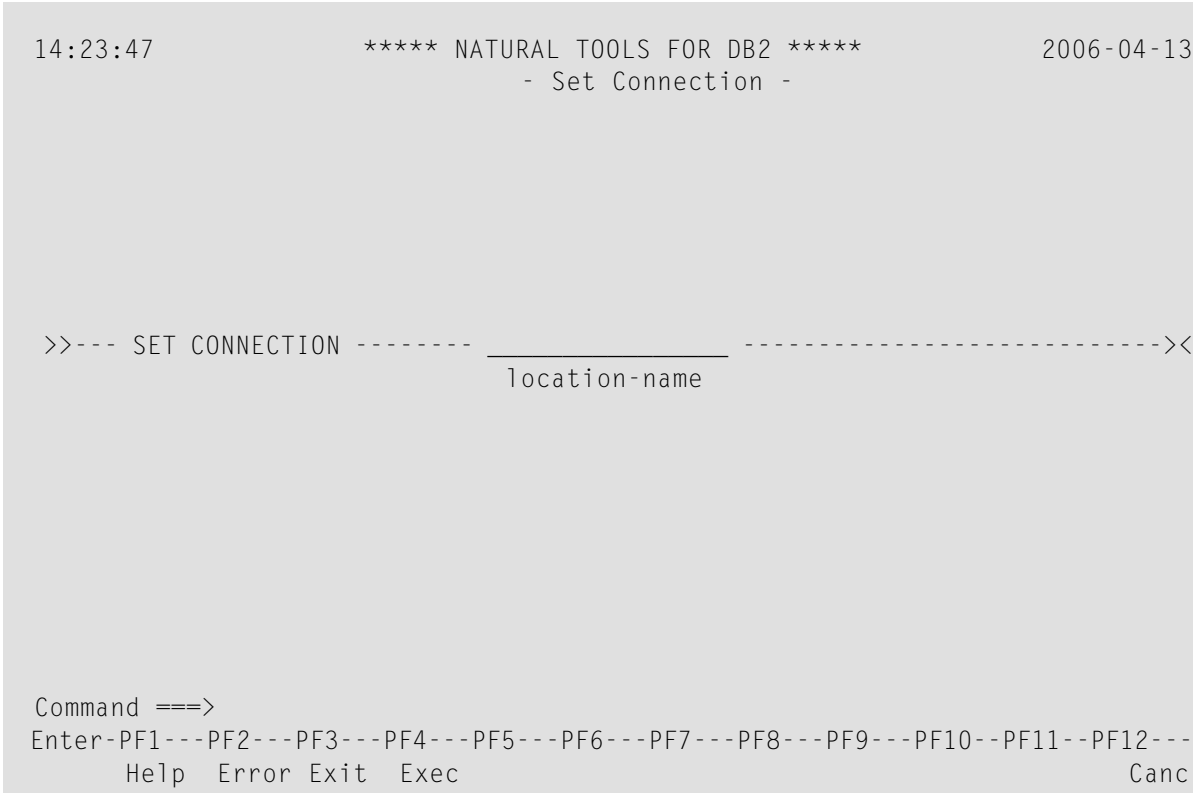
Set Connection

Um die Funktion Set Connection aufzurufen:

- Geben Sie im Bildschirm **Environment Setting** den Funktionscode `SC` und drücken Sie Enter.

  Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SC` ein und drücken Sie Enter.

  Der Bildschirm **Set Connection** wird angezeigt:



Auf dem Bildschirm **Set Connection** identifizieren Sie einen Server, indem Sie einen Standortnamen (`location-name`) angeben. Der identifizierte Server muss dem lokalen Db2-Subsystem bekannt sein.

Set Current SQLID

Um die Funktion Set Current SQLID aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SS` ein und drücken Sie Enter.

  Der Bildschirm **Set Current SQLID** wird angezeigt:

```
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
```

Die Funktion **Set Current SQLID** ändert den Wert des SQL-Autorisierungsbezeichners. Bei SQL-Statements, die unqualifizierte Tabellennamen verwenden, verwendet Db2 die SQLID als impliziten Tabellenqualifizierer. Dadurch können Sie auf identische Tabellen mit demselben Tabellennamen, aber mit unterschiedlichen Ersteller-Namen zugreifen.

Auf dem Bildschirm **Set Current SQLID** können Sie den Wert von `CURRENT SQLID` durch den Wert des speziellen Registers `USER` oder durch eine String-Konstante ersetzen. Die String-Konstante kann bis zu 8 Zeichen lang sein.

In allen unterstützten TP-Monitorumgebungen kann die SQLID dann über Terminal-Ein-/Ausgaben hinweg beibehalten werden, bis sie zurückgesetzt oder die Sitzung beendet wird.

Set Current Packageset

Um die Funktion Set Current Packageset aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SP` ein und drücken Sie Enter.

  Der Bildschirm **Set Current Packageset** wird angezeigt:

```
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
```

Das Statement `SET CURRENT PACKAGESET` weist dem speziellen Register `CURRENT PACKAGESET` einen Wert zu.

Auf dem Bildschirm **Set Current Packageset** können Sie den Wert von `CURRENT PACKAGESET` durch den Wert des speziellen Registers `USER` oder durch eine bis zu 18-stellige String-Konstante ersetzen.

Set Current Degree

Um die Funktion Set Current Degree aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SD` ein und drücken Sie Enter.

  Der Bildschirm **Set Current Degree** wird angezeigt:

```
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` legt den Grad der Parallelität für die Ausführung von Abfragen fest, die vom Anwendungsprozess dynamisch vorbereitet werden.

Set Current Rules

Um die Funktion Set Current Rules aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SU` ein und drücken Sie Enter.

  Der Bildschirm **Set Current Rules** wird angezeigt:

`CURRENT RULES` gibt an, ob bestimmte SQL-Statements gemäß den Db2-Regeln oder den Regeln des SQL-Standards ausgeführt werden.

Set Current Optimization Hint

Um die Funktion Set Current Optimization Hint aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SO` ein und drücken Sie Enter.

  Der Bildschirm **Set Current Optimization Hint** wird angezeigt:

```
 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` gibt den benutzerdefinierten Optimierungshinweis an, den Db2 verwenden soll, um den Zugriffspfad für dynamische Statements zu generieren.

Set Current Locale LC_CType

Um die Funktion Set Current Locale LC_CType aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SL` ein und drücken Sie Enter.

  Der Bildschirm **Set Current Locale LC_CType** wird angezeigt:

```
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
```

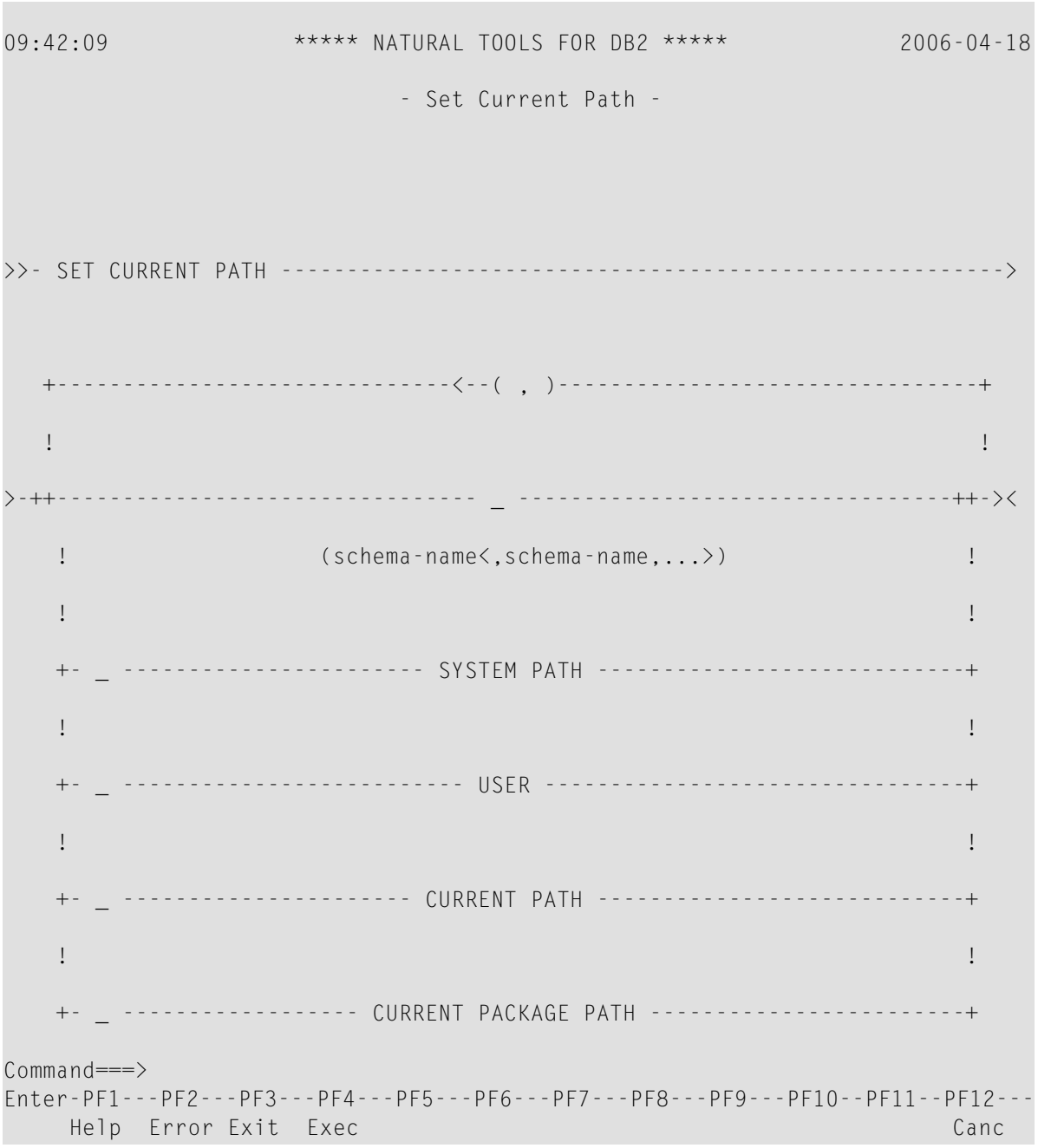
`CURRENT LOCALE LC_CTYPE` gibt das Gebietsschema `LC_CTYPE` (Locale) an, das für die Ausführung von SQL-Statements verwendet wird, die eine eingebaute Funktion verwenden, die auf ein Gebietsschema verweist.

Set Current Path

Um die Funktion Set Current Path aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SA` ein und drücken Sie Enter.

  Der Bildschirm **Set Current Path** wird angezeigt:



`CURRENT PATH` gibt den SQL-Pfad an, der zur Auflösung von nicht qualifizierten Datentypnamen und Funktionsnamen in dynamisch vorbereiteten SQL-Statements verwendet wird.

Set Current Precision

Um die Funktion Set Current Precision aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SE` ein und drücken Sie Enter.

  Der Bildschirm **Set Current Precision** wird angezeigt:

```
 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` legt die Regeln fest, die verwendet werden sollen, wenn beide Operanden in einer Dezimaloperation eine Genauigkeit von 15 oder weniger haben.

Set Current Maintained Types for Optimization

Um die Funktion Set Current Maintained Types aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SM` ein und drücken Sie Enter.

  Der Bildschirm **Set Current Maintained Types** (zur Optimierung) wird angezeigt:

```
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` gibt einen Wert an, der die Objekttypen identifiziert, die bei der Optimierung der Verarbeitung von dynamischen SQL-Abfragen berücksichtigt werden können. Dieses Register enthält ein Schlüsselwort, das für Tabellentypen steht.

Set Current Package Path

Um die Funktion Set Current Package Path aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SB` ein und drücken Sie Enter.

  Der Bildschirm **Set Current Package Path** wird angezeigt:

```
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` gibt einen Wert an, der den Pfad angibt, der verwendet wird, um Referenzen auf Packages aufzulösen, die zur Ausführung von SQL-Anweisungen verwendet werden.

Set Current Refresh Age

Um die Funktion Set Current Refresh Age aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SF` ein und drücken Sie Enter.

  Der Bildschirm **Set Current Refresh Age** wird angezeigt:

```
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
```

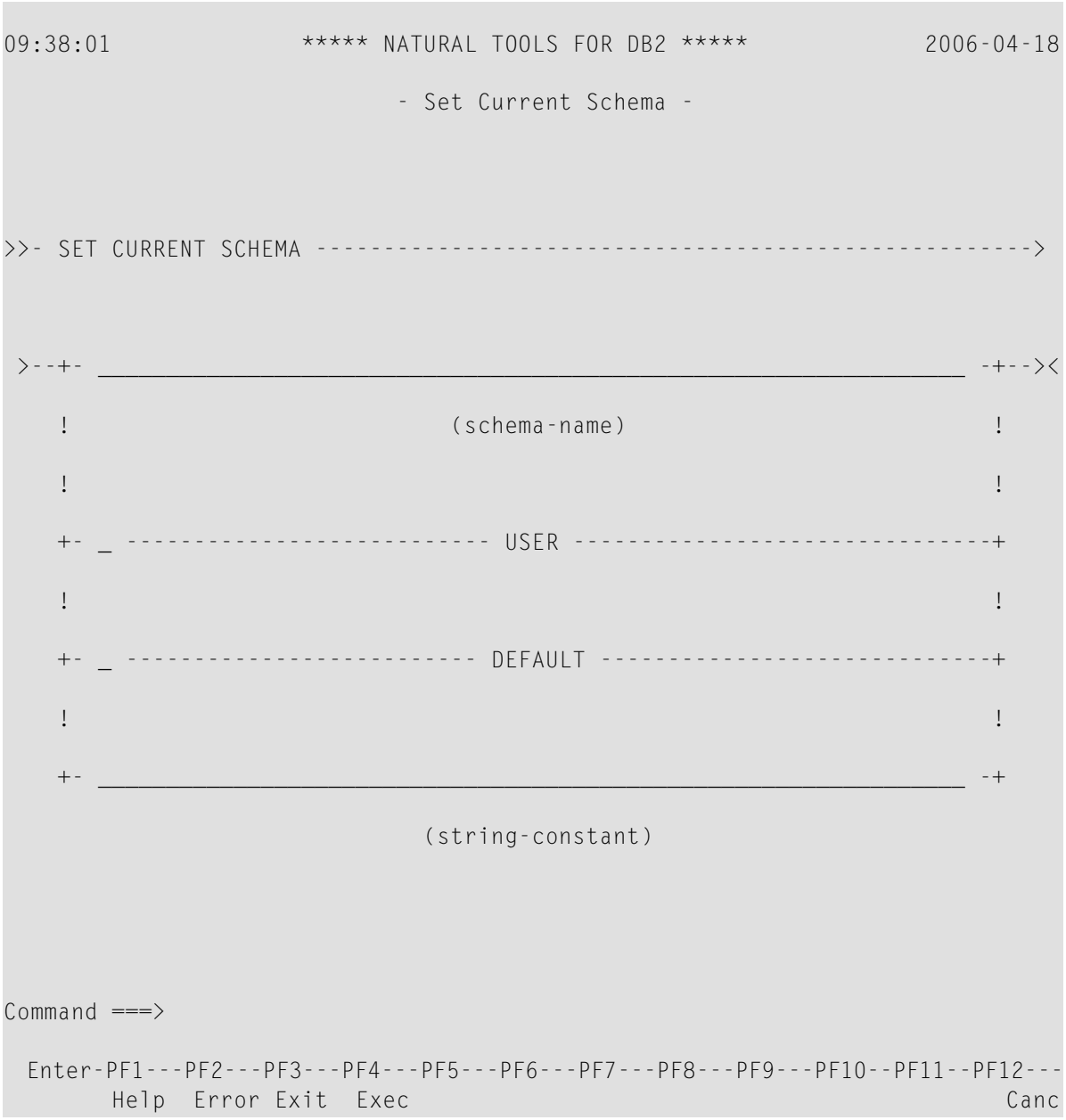
`CURRENT REFRESH AGE` gibt einen Zeitstempel-Dauerwert mit einem Datentyp DECIMAL an.

Set Current Schema

Um die Funktion Set Current Schema aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SH` ein und drücken Sie Enter.

  Der Bildschirm **Set Current Schema** wird angezeigt:



Das spezielle Register `CURRENT SCHEMA`, oder äquivalent `CURRENT_SCHEMA`, gibt den Schemanamen an, der verwendet wird, um nicht qualifizierte Datenbankobjektreferenzen in dynamisch vorbereiteten SQL-Statements zu qualifizieren.

Set Current Application Encoding Scheme

Um die Funktion Set Current Application Encoding Scheme aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SN` ein und drücken Sie Enter.

  Der Bildschirm **Set Current Application Encoding Scheme** wird angezeigt:

```
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
```

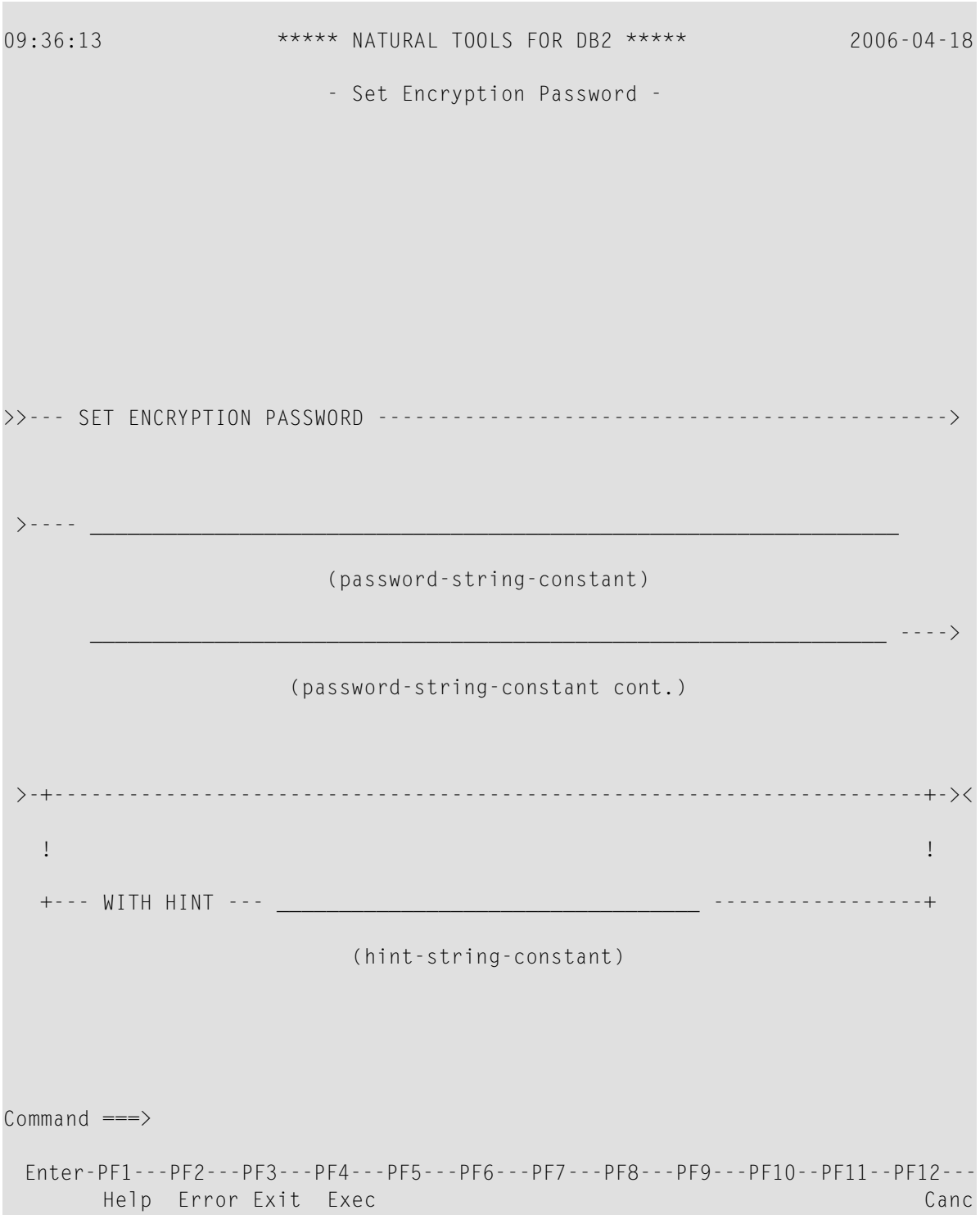
  `CURRENT APPLICATION ENCODING SCHEME` gibt an, welches Kodierungsschema für dynamische Statements verwendet werden soll. Mit dieser Funktion kann eine Anwendung das Kodierungsschema angeben, das für die Datenverarbeitung verwendet wird.

Set Encryption Password

Um die Funktion Set Encryption Password aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SY` ein und drücken Sie Enter.

  Der Bildschirm **Set Encryption Password** wird angezeigt:



Mit der Funktion **Set Encryption Password** werden der Wert des Verschlüsselungskennworts und optional der Kennworthinweis festgelegt.

Display Special Registers

Um die Funktion Display Special Registers aufzurufen:

- Geben Sie auf dem Bildschirm **Environment Setting** den Funktionscode `SR` und drücken Sie Enter.

  Der Bildschirm **Display Special Registers** wird angezeigt:

```
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
```

  Wenn Sie PF11 drücken, wird der nächste Bildschirm mit den Werten der speziellen Register angezeigt.

```
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
```

Wenn Sie PF10 drücken, wird der vorherige Bildschirm mit den Werten der speziellen Register angezeigt.

Die Bildschirme der Funktion **Display Special Registers** zeigen Ihnen die aktuellen Werte der von Natural for Db2 unterstützten speziellen Register von Db2 an.

Felder, denen ein Pluszeichen (+) vorangestellt ist, können mehr Daten enthalten als auf dem Bildschirm angezeigt werden. Sie können den vollständigen Inhalt anzeigen, indem Sie den Cursor auf das Feld (Beschreibung oder Daten) stellen und Enter drücken oder indem Sie die Abkürzung des Feldes (die Großbuchstaben der Beschreibung) mit dem vorangestellten Pluszeichen (+) in die Kommandozeile eingeben. So zeigt z.B. `+PS` ein Fenster mit dem vollständigen Wert des **Current Package Set** an.

10 Explain PLAN_TABLE-Funktionalität benutzen

⚠ **Wichtig:** Bevor Sie die Funktion Explain PLAN_TABLE verwenden: Lesen Sie den Abschnitt *LISTSQL and Explain Functions* unter *Special Requirements for Natural Tools for DB2* in der *Installing Natural for DB2 on z/OS*-Dokumentation.

Die Funktion **Explain PLAN_TABLE** der **Natural Tools for Db2** interpretiert die Ergebnisse von SQL EXPLAIN-Kommandos aus Ihrer `PLAN_TABLE`. Die in Ihrer `PLAN_TABLE` enthaltenen Informationen werden in so genannten **Explanations** (Erklärungen) dargestellt.

Die **Explanations** einer `PLAN_TABLE` beschreiben die von Db2 gewählten Zugriffspfade zur Ausführung von SQL-Statements.

Ein SQL-Statement wird von Db2 in einem oder mehreren Schritten ausgeführt. Für jeden Ausführungsschritt wird eine Zeile in die `PLAN_TABLE` eingefügt. Alle Zeilen zusammen, die den Zugriffspfad für ein SQL-Statement beschreiben, werden als „Explanation“ bezeichnet.

Die Explanations werden in der `PLAN_TABLE` durch eine Kombination aus **Plan Name** (Plan-Name), **DBRM Name** (Name des Datenbankanforderungsmoduls) und **Query Number** (Abfragenummer) oder **Collection Name** (Sammlungsname), **Package Name** und **Query Number** (Abfragenummer) identifiziert.

EXPLAIN-Modi

Db2 bietet drei Arten, SQL-Statements zu erklären:

- Dynamic EXPLAIN
- Bind Plan EXPLAIN
- Bind Package EXPLAIN

Je nach Art unterscheiden sich die Bezeichnungen der Erklärungen.

Dynamic EXPLAIN

Führt ein SQL `EXPLAIN`-Kommando dynamisch aus, wobei die Erklärung in die `PLAN_TABLE` Ihrer aktuellen SQLID eingefügt wird.

Das `EXPLAIN`-Kommando kann in der **Catalog Maintenance**-Funktion und der **Interactive SQL**-Funktion der **Natural Tools for Db2** abgesetzt werden. Außerdem können Sie mit dem Natural-Kommando `LISTSQL` SQL-Statements aus katalogisierten Natural-Programmen extrahieren und mit dem SQL `EXPLAIN`-Kommando für die extrahierten SQL-Statements absetzen.

Wenn Sie das SQL `EXPLAIN`-Kommando dynamisch absetzen, sollten Sie eine Query-Nummer angeben, damit Sie die Erklärung in der `PLAN_TABLE` identifizieren können. Die gleiche Query-Nummer sollte für zugehörige Statements verwendet werden.

Abhängig von der Methode, mit der das vom dynamischen SQL-Prozessor verwendete DBRM in den Plan eingebunden wird, verwendet Db2 zwei verschiedene Methoden zur Identifizierung von Zeilen in der `PLAN_TABLE`:

- Dynamic Mode
- Package Mode

Dynamic Mode

Das DBRM wird direkt in den Plan eingebunden.

Wenn eine Erklärung eingefügt wird, werden der **Plan Name**, der **DBRM Name** und die **Query Number** von Db2 wie folgt bestimmt:

Parameter	Beschreibung
`plan name`	Wird leer gelassen.
`DBRM name`	Ist der Name des vom dynamischen SQL-Prozessor verwendeten DBRM
`query number`	Ist gleich der Query-Nummer, die Sie mit dem `EXPLAIN`-Kommando angegeben haben (die Standard-Query-Nummer ist `1`).

Dieser Explain-Modus wird als dynamischer Modus bezeichnet.

Package Mode

Das DBRM wird als Package in den Plan eingebunden.

Wenn eine **Explanation** (Erklärung) eingefügt wird, werden der **Collection Name** (Sammlungsname), der **Package Name** (Paketname) und die **Query Number** (Abfragenummer) von Db2 wie folgt bestimmt:

Parameter	Beschreibung
`collection name`	Ist der Name der Collection, die das Package enthält.
`package name`	Ist der Name des Package, das vom dynamischen SQL-Prozessor verwendet wird.
`query number`	Ist gleich der Abfragenummer, die Sie mit dem `EXPLAIN`-Kommando angegeben haben (die Standardabfragenummer ist `1`).

This explanation mode is called package mode.

Bind Plan EXPLAIN

Bindet einen Anwendungsplan mit der Option **EXPLAIN YES**, wobei die Erklärung in die `PLAN_TABLE` des Eigentümers des Plans eingefügt wird. Wenn eine Erklärung eingefügt wird, werden der **Plan Name,** der **DBRM Name** und die **Query Number** von Db2 wie folgt bestimmt:

Parameter	Beschreibung
`plan name`	Ist der Name des Plans.
`DBRM name`	Ist der Name des DBRM, das das SQL-Statement enthält.
`query number`	Ist gleich der Statement-Nummer (*`stmtno`*), die durch den Db2-Precompiler generiert wird.

Bind Package EXPLAIN

Bindet ein Package mit der Option **EXPLAIN YES**, wobei die Erklärung in die `PLAN_TABLE` des Eigentümers des Package eingefügt wird.

Wenn eine Erklärung eingefügt wird, werden der **Collection Name**, der **Package Name** und die **Query Number** von Db2 wie folgt ermittelt:

Parameter	Beschreibung
`collection name`	Ist der Name der Sammlung, die das Package enthält.
`package name`	Ist der Name des Package, das das SQL-Statement enthält.
`query number`	Ist gleich der Statement-Nummer (*`stmtno`*), die durch den Db2-Precompiler generiert wird.

Funktion EXPLAIN_TABLE aufrufen

Erklärungen (**Explanations**) können entweder nach **Plan Name**, **DBRM Name** und **Query Number** oder nach **Collection Name**, **Package Name** und **Query Number** ausgewählt werden.

Wenn Sie ein `EXPLAIN`-Kommando mehrmals absetzen, ist es möglich, dass mehrere Erklärungen durch eine gegebene Kombination dieser Auswahlfelder identifiziert werden. Sie können also entweder alle Erklärungen oder nur die jüngste auswählen. Es wird eine Liste mit allen ausgewählten Erklärungen angezeigt, aus der Sie einzelne Zeilen für eine genauere Beschreibung auswählen können:

- Die einzelnen Zeilen einer `PLAN_TABLE` werden nacheinander angezeigt.
- Zeilen, die dasselbe SQL-Statement beschreiben, werden zusammen als eine Erklärung angezeigt.
- Unterschiedliche Erklärungen werden durch Leerzeilen getrennt.

Sie können durch die Liste blättern und einen detaillierten Report für einzelne Erklärungen auswählen.

Wenn Zeilen in Ihre PLAN_TABLE als Ergebnis eines Natural-Systemkommandos LISTSQL eingefügt worden sind, werden auch die Namen der Natural Library und des Programms angezeigt.

Um die Funktion Explain PLAN_TABLE aufzurufen:

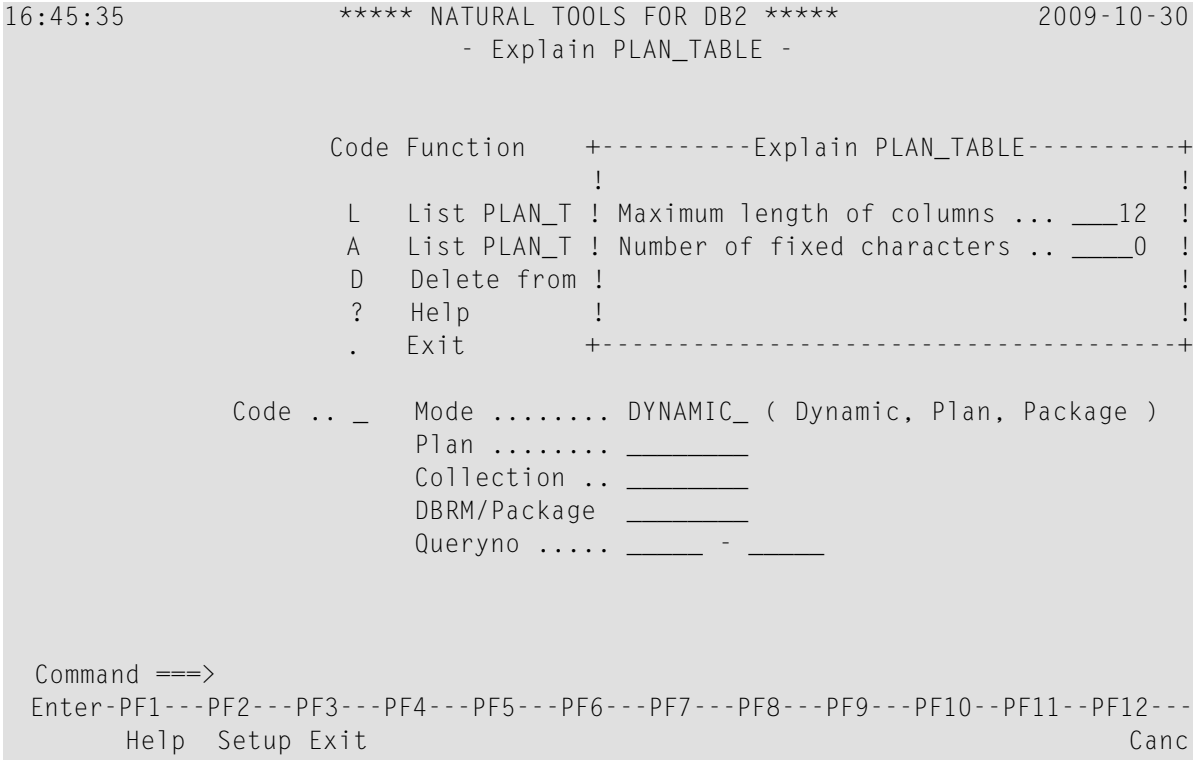
- Geben Sie im **Natural Tools for DB2 Main Menu** den Funktionscode X ein.

  Der Bildschirm **Explain PLAN_TABLE** wird angezeigt:

```
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
```

  Mit PF2 (Setup) kann die maximale Länge einer Spalte und die Anzahl der festen Zeichen beim Blättern nach links festgelegt werden. Die Standardwerte für beide Parameter können im Subprogramm CONFIG in der Bibliothek SYSDB2 geändert werden.

  Wenn ein Spaltenwert länger als die maximale Länge ist, wird er abgeschnitten und mit einem Größer-als-Symbol (>) gekennzeichnet, was bedeutet, dass Zeichenketten am rechten Ende abgeschnitten werden, oder mit einem Kleiner-als-Symbol (<), was bedeutet, dass Zahlen am linken Ende abgeschnitten werden. Beachten Sie, dass für weitere Kommandos in einer Zeile, z. B. das Zeilenkommando I, nur der sichtbare Wert als Eingabe verwendet werden kann. Dies bedeutet, dass Kommandos in Zeilen fehlschlagen, wenn die Werte für die weitere Verarbeitung abgeschnitten werden.



Die folgenden Funktionen sind verfügbar:

Code	Beschreibung
L	Die Funktion **List PLAN_TABLE - Latest Explanations** listet die letzte Erklärung für eine beliebige Kombination der unten beschriebenen Parameter auf.
A	Die Funktion **List PLAN_TABLE - All Explanations** listet alle Erklärungen für eine beliebige Kombination der unten beschriebenen Parameter auf.
D	Die Funktion **Delete from PLAN_TABLE** löscht die angegebenen Erklärungen aus Ihrer PLAN_TABLE.

Die folgenden Parameter können angegeben werden:

Parameter	Beschreibung
Mode	Gibt den Explanation Mode an (Dynamic, Plan oder Package).
Plan *plan-name*	Gibt einen gültigen Plan-Namen an. Der Parameter Plan ist nur im Plan Mode erforderlich.
Collection *collection-name*	Gibt einen gültigen Collection-Namen an. Der Parameter Collection ist nur im Package Mode erforderlich.
DBRM/Package *dbrm/package-name*	Gibt im Plan Mode einen gültigen DBRM-Namen an. Gibt im Package Mode einen gültigen Package-Namen an.

Parameter	Beschreibung
	Gibt im Dynamic Mode das DBRM an, das vom dynamischen SQL-Prozessor verwendet wird. Wenn ein Wert gefolgt von einem Stern (*) angegeben wird, werden alle DBRMs/Packages des angegebenen Plans/der angegebenen Collection berücksichtigt, deren Namen mit dem angegebenen Wert beginnen. Wird nur Stern-Notation benutzt, werden alle DBRMs/Packages des angegebenen Plans/der angegebenen Collection berücksichtigt. Mit dem Parameter `DBRM/Package` kann die Anzeige auf einzelne DBRM/Packages beschränkt werden.
`Queryno` *`no.1 - no.2`*	Dieser Parameter gibt einen gültigen Bereich von Query-Nummern an, für den die folgenden Regeln gelten: ■ Wenn keine Query-Nummer angegeben wird, werden alle Query-Nummern angezeigt. ■ Wenn nur die erste Query-Nummer angegeben wird, wird nur diese Query-Nummer angezeigt. ■ Wenn nur die zweite Query-Nummer angegeben wird, werden alle Query-Nummern bis zur zweiten Query-Nummer und einschließlich dieser angezeigt. ■ Wenn beide Query-Nummern angegeben werden, werden alle Query-Nummern zwischen der ersten und der zweiten Query-Nummer (einschließlich) angezeigt.

List PLAN_TABLE - Latest Explanations

Diese Funktion listet nur die neueste Erklärung für eine beliebige Kombination von entweder **Plan Name**, **DBRM Name** und **Query Number** oder **Package Name**, **Collection Name** und **Query Number** auf.

List PLAN_TABLE - All Explanations

Diese Funktion listet alle Erklärungen für eine beliebige Kombination von entweder **Plan Name**, **DBRM Name** und **Query Number** oder **Package Name**, **Collection Name** und **Query Number** auf. Die **Query Number**-Parameter werden wie oben interpretiert.

Beispiel für eine Auflistung von Erklärungen

```
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           -           ----  --O-
 __ 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           -           ----  --O-
 __ 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
```

Verfügbare Kommandos

Die folgenden Zeilenkommandos sind in den Auflistungen der Funktion **Explain PLAN_TABLE** verfügbar. Die Zeilenkommandos werden vor einer der Zeilen der gewünschten Erklärung(en) eingegeben.

Kommando	Beschreibung
I	Zeigt ein Fenster an, in dem zusätzliche Informationen zu einer Erklärung ausgewählt werden können.
S	Wählt eine Erklärung aus, die mit dem unten beschriebenen INFO-Kommando verwendet werden soll.
U	Macht die Auswahl einer Erklärung zur Verwendung mit dem INFO-Kommando rückgängig.

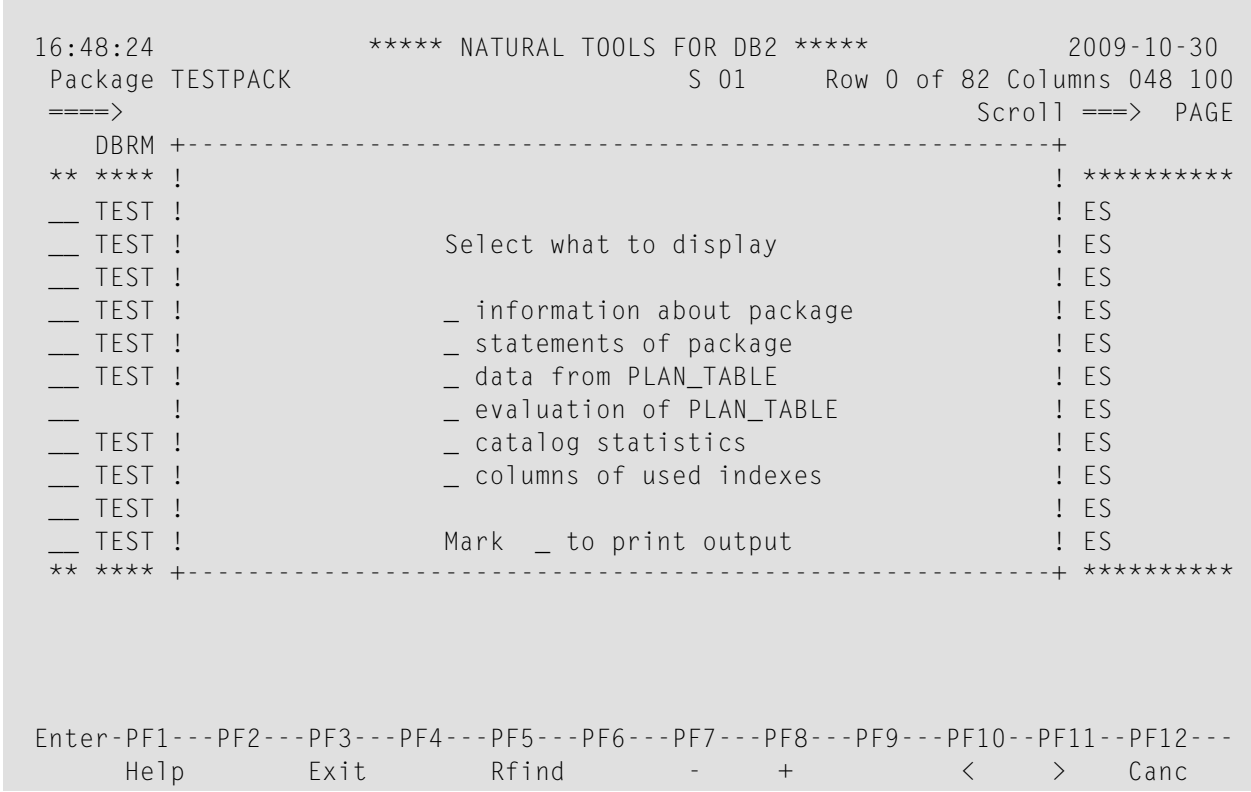
Eine Liste der verfügbaren Zeilenkommandos kann als Fenster aufgerufen werden, indem das Hilfezeichen, d.h. ein Fragezeichen (?), vor einer der aufgelisteten Zeilen eingegeben wird.

Neben den Zeilenkommandos kann auch das INFO-Kommando angegeben werden. Das INFO-Kommando muss in der Kommandozeile des Auflistungsbildschirms eingegeben werden und ist das Äquivalent zum Zeilenkommando I. Das INFO-Kommando zeigt ein Fenster, in dem zusätzliche Informationen zu allen zuvor mit dem Zeilenkommando S ausgewählten Erläuterungen ausgewählt werden können.

Im Plan Mode wird das folgende Fenster angezeigt, in dem Sie auswählen können, welche Zusatzinformationen angezeigt oder gedruckt werden sollen.

```
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
```

Analog dazu wird im Package Mode das folgende Fenster angezeigt:



Das Blättern in den angezeigten Daten erfolgt mit Browse-Kommandos, von denen die wichtigsten auch über PF-Tasten abgesetzt werden können; siehe *Mit den Natural Tools for Db2 editieren*.

Option	Beschreibung
Information about plan/package	Wenn ein Plan/Package-Name angegeben wurde, enthält diese Option Informationen aus dem Db2-Katalog, z. B. Datum und Uhrzeit der Bindung sowie verschiedene Bindungsoptionen. Im Dynamic Mode ist diese Option nicht verfügbar.
Statements of plan/package	Wenn ein Plan/Package-Name angegeben wurde, liefert diese Option Informationen zu den erklärten SQL-Statements, die in diesem Package enthalten sind. Diese Informationen werden dem Db2-Katalog entnommen. Im Dynamic Mode ist diese Option nicht verfügbar.
Data from PLAN_TABLE	Diese Option liefert Informationen aus der `PLAN_TABLE` über die ausgewählten Zeilen.
Evaluation of PLAN_TABLE	Diese Option liefert eine Beschreibung der PLAN_TABLE. Sie beschreibt für jeden Ausführungsschritt: ■ die von Db2 gewählten Sperren, ■ ob eine Join-Operation durchgeführt wird, ■ ob die Daten sortiert werden und warum die Sortierung durchgeführt wird,

Option	Beschreibung
	■ den Zugriffspfad im Detail.
Catalog statistics	Diese Option liefert statistische Informationen aus dem Db2-Katalog.
Columns of used indexes	Diese Option liefert die Spalten der verwendeten Indizes einschließlich der Katalogstatistiken zu diesen Spalten.

Delete from PLAN_TABLE

Die Funktion **Delete from PLAN_TABLE** wird auch verwendet, um `PLAN_TABLE`-Erklärungen in Abhängigkeit von der angegebenen Kombination von entweder **Plan Name**, **DBRM Name** und **Query Number** oder **Collection Name**, **Package Name** und **Query Number** auszuwählen. Diesmal werden die ausgewählten `PLAN_TABLE`-Erklärungen jedoch nicht angezeigt, sondern gelöscht.

Die Funktion **Delete from PLAN_TABLE** ist nützlich, um alte Daten zu löschen, bevor ein Plan gebunden oder neu gebunden wird oder bevor ein SQL `EXPLAIN`-Kommando ausgeführt wird.

Um zu verhindern, dass `PLAN_TABLE`-Erklärungen ungewollt gelöscht werden, werden Sie um eine Bestätigung gebeten:

```
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
```

Abgesehen von den globalen PF-Tasten-Belegungen wird bei der Funktion **Delete from PLAN_TABLE** der Funktion **Explain PLAN_TABLE** die Taste PF5 (Del) verwendet, um das Löschen von zuvor ausgewählten Erklärungen zu bestätigen.

Explain PLAN_TABLE-Funktion für Massen- und Stapelverarbeitung

Für die Online-Massenverarbeitung und die Ausführung im Batch-Modus steht auch eine angepasste **Explain PLAN_TABLE**-Funktion zur Verfügung.

Funktion EXPLAINB für die Massenverarbeitung

Für die Online-Massenverarbeitung steht eine modifizierte Version der Funktion **Explain PLAN_TABLE** zur Verfügung.

Um die modifizierte Version der Explain PLAN_TABLE-Funktion aufzurufen:

1 Melden Sie sich in der Natural Library SYSDB2 an.

2 Geben Sie in der Kommandozeile das Kommando `EXPLAINB` ein und drücken Sie Enter.

Folgendes wird angezeigt:

```
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
```

Zusätzlich zu den Funktionscodes `L` (Funktion **List PLAN_TABLE - Latest Explanations**) und `A` (Funktion **List PLAN_TABLE - All Explanations**) steht der Funktionscode `O` (**Output Options**) zur Verfügung.

Mit der Funktion **Output Options** können Sie die Ausgabe von Informationen zu `PLAN_TABLE`-Einträgen einschränken. Die verschiedenen Optionen werden in einem Fenster aufgelistet, das durch Eingabe des Funktionscodes `O` im obigen Menü **Explain PLAN_TABLE** aufgerufen wird. Das Fenster ähnelt demjenigen, das durch die Online-Kommandos `I` oder `INFO` aufgerufen wird.

```
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
```

Wurde die Funktion **Output Options** gewählt, werden nur die Informationen gedruckt, die unter die für die Ausgabe markierten Optionen fallen.

Wurde der Funktionscode `O` nicht gewählt, werden alle Informationen zu den Einträgen der `PLAN_TABLE` gedruckt, die unter die im obigen Fenster aufgeführten Optionen fallen.

In beiden Fällen werden Sie aufgefordert, einen Drucker anzugeben.

EXPLAINB im Batch-Modus

Die Funktionalität von `EXPLAINB` ist neben der Online-Massenverarbeitung insbesondere für die Batch-Verarbeitung vorgesehen. Wird `EXPLAINB` im Batch-Modus verwendet, erfolgt die Ausgabe in ein Dataset, das mit dem DD-Namen `CMPRT01` (logischer Drucker 1) referenziert wird.

11 File Server-Statistiken

Wenn ein File Server installiert wurde, wird der File Server-Statistikteil der **Natural Tools for Db2** verwendet, um Statistiken über die Nutzung des File Servers anzuzeigen.

Um die Funktion File Server-Statistik aufzurufen:

- Geben Sie im Bildschirm **Natural Tools for DB2 Main Menu** den Funktionscode `F` ein und drücken Sie Enter.

  Der Bildschirm **File Server - Generation Statistics** wird angezeigt:

```
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
```

Dieser Bildschirm enthält Informationen zu den Parametern, die bei der Generierung des File Servers angegeben werden müssen.

Wenn das Speichermedium des File Servers der Software AG Editor Buffer Pool ist, sieht der Bildschirm **File Server - Generation Statistics** wie folgt aus:

```
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
```

Wenn Sie PF11 (Next) drücken, wird ein zweiter Bildschirm angezeigt, der Bildschirm **File Server - User Statistics**, der die Statistiken anzeigt, die seit der Installation des File Servers geführt wurden - **Statistics since Generation** -, und Statistiken über die aktuelle Natural-Sitzung - **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
```

Wenn Sie PF10 (Prev) drücken, gelangen Sie zurück zum Bildschirm **File Server - Generation Statistics**.

Die Statistiken werden jedes Mal aktualisiert, wenn Sie Enter, PF10 oder PF11 drücken.

Wenn das Speichermedium des File Servers der Software AG Editor Buffer Pool ist, sieht der Bildschirm **File Server - User Statistics** wie folgt aus:

```
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
```

Beachten Sie, dass der Abschnitt **Statistics since Generation** bei dieser Anzeige nicht bereitgestellt werden konnte.

Für File Server-VSAM-Dateien bietet Natural for Db2 auch die Anzeige und Verwaltung des File Server-Verzeichnisses.

Wenn Sie PF9 (Dire) drücken, werden die aktiven Verzeichniseinträge mit den Sitzungskennungen und den ihnen zugeordneten File Server-Blöcken aufgelistet. Die Anzeige sieht wie folgt aus:

```
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 TCRO     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  ↩
 ↩
```

In der Spalte **Birth** wird das ungefähre Entstehungsdatum der File Server-Sitzung angezeigt, wenn sie mit Natural for Db2 Version 4.3 erstellt wurde. Wurde die File Server-Sitzung mit einer früheren Version von Natural for Db2 erstellt, wird das Entstehungsdatum der File Server-Sitzung als `pre NDB43` (vor NDB-Version 4.3) angezeigt.

Der Bildschirm **Directory Entries** bietet die Möglichkeit, durch die Verzeichniseinträge zu blättern und einen bestimmten Eintrag an den Anfang des Bildschirms zu stellen.

Darüber hinaus können Sie hier alle File Server-Blocknummern eines Verzeichniseintrags auflisten (PF4, Zeilenkommando `L`) oder einen Verzeichniseintrag vom File Server löschen (PF10, Zeilenkommando `D`). Sie sollten Verzeichniseinträge nur löschen, wenn Sie sicher sind, dass die zugehörige Natural-Sitzung nicht mehr aktiv ist, da sonst die File Server-Struktur durch das Löschen zerstört werden könnte.

Verzeichniseinträge spiegeln die File Server-Sitzungen zu einem bestimmten Zeitpunkt wider. Wenn Sie PF11 drücken, wird die Anzeige aus der Datei zu einem anderen (aktuellen) Zeitpunkt aktualisiert.

12 Db2-Kommandos aus Natural absetzen

Mit dem Teil **DB2 Command Execution** der **Natural Tools for DB2** können Sie Db2-Kommandos aus einer Natural-Umgebung heraus absetzen.

Für jeden Benutzer wird eine Datei in der Systemdatei FUSER geführt. Diese Datei wird unter dem Objektnamen `DB2$CMD` in der Natural Library des aktuellen Benutzers gespeichert.

Sie können ein Kommando auswählen und übergeben, die Kommandodatei speichern und den Ausgabereport speichern und/oder drucken.

Funktion Execute DB2 Command aufrufen

Um die Funktion aufzurufen:

- Geben Sie im **Natural Tools for DB2 Main Menu** den Funktionscode `D` ein und drücken Sie Enter.

  Der Bildschirm **Execute DB2 Command** wird angezeigt:

```
 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
```

  Die folgenden Funktionen sind verfügbar:

Code	Beschreibung
C	Zeigt Ihre Kommandodatei an. Wenn Sie noch keine Kommandodatei gespeichert haben, wird eine Standarddatei angezeigt.
O	Wenn eine Ausgabedatei vorhanden ist, wird der Ausgabereport angezeigt.

Die folgenden Parameter können angegeben werden:

Parameter	Beschreibung
Library	Sie können einen Benutzernamen oder eine Library angeben. Der Standardwert ist die aktuelle Benutzerkennung.

Kommando-Datei anzeigen

Um die Kommando-Datei anzuzeigen:

- Geben Sie im Menü **Execute DB2 Command** den Funktionscode C ein und drücken Sie Enter.

  Der Bildschirm **DB2 Commands** wird angezeigt:

```
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
```

Mit PF11 (Next) können Sie zur nächsten Seite blättern.

Sie können den Inhalt der Kommandodatei ändern. Speichern Sie Ihre Änderungen mit PF5 (Save).

Um ein Kommando auszuführen:

- Markieren Sie das Kommando mit einem S und drücken Sie PF4 (Subm).

Die Ergebnisse werden auf dem Bildschirm **DB2 Commands Output** angezeigt.

```
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
```

Um die Kommandodatei zu speichern:

1 Drücken Sie PF5 (Save).

Die Ausgabedatei wird unter dem Objektnamen DB2$OUT in der Natural Library des aktuellen Benutzers gespeichert.

2 Drücken Sie PF3 (Exit) um zur Kommandodatei zurückzukehren.

Sie können weitere Kommandos zur Ausführung übergeben.

Ausgabereport anzeigen

Um den letzten Ausgabe-Datensatz anzuzeigen:

- Geben Sie im Menü **Execute DB2 Command** den Funktionscode 0 ein und drücken Sie Enter.

  Der Bildschirm **DB2 Commands Output** wird angezeigt:

```
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
```

  Drücken Sie PF5 (Print), um den Ausgabe-Datensatz zu drucken.

13 Natural-Systemkommandos für Db2 benutzen

Die folgenden Natural-Systemkommandos wurden in die **Natural Tools for Db2** integriert:

Natural-Systemkommando	Erläuterung
`LISTSQL`	Listet Natural-DML-Statements und die entsprechenden SQL-Statements auf.
`LISTSQLB`	Liefert Erklärungen zu SQL-Statements für ein bestimmtes Objekt.
`SQLERR`	Liefert Informationen über die SQL Communication Area (SQLCA) bei einem Db2-Fehler.
`SQLDIAG`	Liefert Diagnoseinformationen über das zuletzt ausgeführte SQL-Statement (mit Ausnahme eines `GET DIAGNOSTICS`-Statements).
`LISTDBRM`	Zeigt entweder eine Liste der Database Request Modules (DBRM) für ein bestimmtes Natural-Programm oder eine Liste der Natural-Programme an, die auf ein bestimmtes DBRM verweisen.

Eine Beschreibung dieser Kommandos finden Sie in der Natural-*Systemkommandos*-Dokumentation.

14 Natural-Datendefinitionsmodule (DDMs) generieren

Damit Natural auf eine Db2-Tabelle zugreifen kann, muss ein logisches Natural-Datendefinitionsmodul (DDM) für die Tabelle generiert werden. Dies geschieht entweder mit Predict (Einzelheiten finden Sie in der entsprechenden *Predict*-Dokumentation) oder mit der Natural Utility SYSDDM. Siehe auch *DDM-Editor (SYSDDM Utility)* in der Natural *Editoren*-Dokumentation.

Wenn Sie Predict nicht installiert haben, verwenden Sie die SYSDDM-Funktion **SQL Services**, um Natural DDMs aus Db2-Tabellen zu erzeugen. Diese Funktion wird über das Hauptmenü der Utility SYSDDM aufgerufen und auf den folgenden Seiten beschrieben.

Weitere Informationen zu Natural DDMs finden Sie unter *Datendefinitionsmodule (DDMs)* im Natural-*Leitfaden zur Programmierung* .

SQL Services (NDB)

Die Funktion **SQL Services (NDB)** der Natural-Utility SYSDDM (siehe *SYSDDM Pflege- und Service-Funktionen benutzen* in der Natural-*Editoren*-Dokumentation) wird für den Zugriff auf Db2-Tabellen verwendet. Sie können damit auf den Katalog des Db2-Servers zugreifen, mit dem Sie verbunden sind, indem Sie z.B. die Funktion **Environment Setting** benutzen, wie in *Natural Tools for DB2* beschrieben, oder indem Sie den Namen eines Servers in das Feld **Server Name** im Menü **SQL Services** eingeben. Der Name des Db2-Servers, mit dem Sie verbunden sind, wird dann in der oberen linken Ecke des Bildschirms **SQL Services Menu** angezeigt. Sie können auf jeden Db2-Server zugreifen, der sich entweder auf einem Großrechner (z/OS) oder einer UNIX-Plattform befindet, wenn die Server über DRDA (Distributed Relational Database Architecture) verbunden wurden. Weitere Einzelheiten zur Verbindung von Db2-Servern und Informationen zur Bindung des Anwendungs-Package (SYSDDM verwendet das E/A-Modul `NDBIOMO`) für den Zugriff auf Daten auf entfernten Servern finden Sie in der entsprechenden IBM-Literatur.

Die Funktion **SQL Services** ermittelt, ob Sie mit einem Großrechner-Db2 oder einem UNIX-Db2 verbunden sind, greift auf den entsprechenden Db2-Katalog zu und führt die unten aufgeführten Funktionen aus.

Anmerkung: Wenn Sie die SYSDDM **SQL Services** in einer CICS-Umgebung ohne File Server verwenden, müssen Sie `CONVERS=ON` im `NTDB2`-Makro angeben, andernfalls erhalten Sie möglicherweise SQLCODE -518.

- SQL Services benutzen
- SQL-Tabelle aus einer Liste auswählen – Funktion: Select SQL Table from a List
- DDM aus einer SQL-Tabelle generieren – Funktion: Generate DDM from an SQL Table
- Spalten einer SQL-Tabelle auflisten – Funktion: List Columns

- Eine User Exit Routine verfügbar machen

SQL Services benutzen

Um die Funktion SQL Services aufzurufen:

1 Geben Sie in der Kommandozeile das Natural-Systemkommando `SYSDDM` ein und drücken Sie Enter.

Oder:

1. Wählen Sie im Natural-Hauptmenü den Menüpunkt **Maintenance and Transfer Utilities**, um das Menü **Maintenance and Transfer Utilities** aufzurufen.
2. Wählen Sie im Menü **Maintenance and Transfer Utilities** die Option **Maintain DDMs**.

Das Menü der Utility SYSDDM wird angezeigt. Die Felder und Funktionen im Menü der Utility SYSDDM werden im Kapitel *SYSDDM Pflege- und Service-Funktionen benutzen* erläutert.

2 Geben Sie im Feld `Code` des Menüs der Utility SYSDDM den Code `B` ein und drücken Sie Enter.

Das Menü **SQL Services** wird angezeigt.

```
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
```

Die auf diesem Bildschirm verfügbaren Funktionen werden in den entsprechenden Abschnitten beschrieben.

SQL-Tabelle aus einer Liste auswählen – Funktion: Select SQL Table from a List

Die Funktion **Select SQL Table** from a List dient dazu, eine Db2-Tabelle aus einer Liste zur weiteren Bearbeitung auszuwählen.

Um die Funktion Select SQL Table from a List aufzurufen:

- Geben Sie im Bildschirm **SQL Services Menu** den Funktionscode `S` ein.
  - Wenn Sie nur den Funktionscode eingeben, erhalten Sie eine Liste aller im Db2-Katalog definierten Tabellen.
  - Wenn Sie nicht eine Liste aller Tabellen, sondern nur einen bestimmten Bereich von Tabellen auflisten möchten, können Sie zusätzlich zum Funktionscode einen Wert in den Feldern **Table Name** und/oder **Creator** (Ersteller) angeben. Sie können Stern-Notation (*) oder das Größer-als-Zeichen (>) als Anfangswert verwenden.

  Drücken Sie Enter.

  Der Bildschirm **Select SQL Table From a List** wird aufgerufen. Er zeigt eine Liste aller angeforderten Db2-Tabellen an. In der Liste können Sie eine Db2-Tabelle mit einem Funktionscode markieren:

Code	Funktion	Beschreibung
`G`	**Generate DDM from an SQL Table**	Mit dieser Funktion können Sie auf der Basis der Definitionen im Db2-Katalog ein Natural DDM aus einer Db2-Tabelle generieren.
`L`	**List Columns of an SQL Table**	Diese Funktion listet alle Spalten einer bestimmten Db2-Tabelle auf.

DDM aus einer SQL-Tabelle generieren – Funktion: Generate DDM from an SQL Table

Mit dieser Funktion können Sie auf der Basis der Definitionen im Db2-Katalog ein Natural DDM aus einer Db2-Tabelle generieren.

Die folgenden Themen werden in diesem Abschnitt behandelt:

- Funktion Generate DDM from an SQL Table aufrufen
- Zuweisung von Standardwerten - Generierung von DDMs im Batch-Modus
- DBID/FNR- Zuweisung
- Generierung langer Felder

- Längenindikator für Felder mit variabler Länge: VARBINARY, VARCHAR, LONG VARCHAR, VARGRAPHIC, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB
- Nullwerte
- Locator-Feld für LOB-Spalte

Funktion Generate DDM from an SQL Table aufrufen

Um die Funktion aufzurufen:

- Geben Sie im Bildschirm **SQL Services Menu** den Funktionscode `G` zusammen mit dem Namen und dem Ersteller (Creator) der Tabelle ein, für die Sie einen DDM generieren möchten.

  - Wenn Sie den Tabellennamen oder den Ersteller nicht kennen, können Sie die Funktion **Select SQL Table from a List** benutzen, um die gewünschte Tabelle auszuwählen.
  - Wenn Sie nicht möchten, dass der Ersteller der Tabelle Teil des DDM-Namens ist, geben Sie ein `N` (Nein) in das Feld **DDM Name with Creator** ein. Die Standardeinstellung ist `Y` (Ja).
  - Wenn Sie ein DDM für eine Tabelle generieren möchten, für die bereits ein DDM existiert, und Sie möchten, dass das bestehende DDM durch das neu generierte DDM ersetzt wird, geben Sie ein `Y` (Ja) in das Feld **Replace** (Ersetzen) ein.

    Standardmäßig ist **Replace** auf `N` (Nein) eingestellt, um zu verhindern, dass ein vorhandenes DDM versehentlich ersetzt wird.
  - Im Feld **Remark** können Sie den Inhalt der DDM-Spalte **Remark** angeben. Geben Sie ein:

`O` (Overwrite)	für SQL-Spaltenbemerkungen, falls definiert, überschrieben durch Feldinformationen, die von Natural generiert werden, falls vorhanden. Dies ist die Standardeinstellung.
	`S` (SQL) für SQL-Spaltenbemerkungen, falls definiert, sonst leer.
`C` (Comment)	für Feldinformationen, die von Natural generiert werden, falls verfügbar. SQL-Spaltenbemerkungen werden in eine separate DDM-Kommentarzeile kopiert.

    Standardmäßig ist **Remark** auf `O` (Overwrite) gesetzt.
  - Um einen Standardwert für die Felder **Code**, **Table Name** (Tabellenname), **Creator** (Ersteller), **Replace** (Ersetzen), **DDM Name with Creator** (DDM-Name mit Ersteller) oder **Remark** (Bemerkung) zu definieren oder zu ändern, verwenden Sie den User-Exit `NDBDDM-2` und seinen Datenbereich NDBDDM-L, der in der Library SYSDB2 enthalten ist. Siehe *Eine User Exit Routine verfügbar machen*. Detaillierte Informationen zur Handhabung von `NDBDDM-2` finden Sie in den Erläuterungen im Quellcode.

Wichtig: Da die Angabe von Sonderzeichen als Teil eines Feld- oder DDM-Namens den Natural-Namenskonventionen widerspricht, müssen alle in Db2 zulässigen Sonderzeichen vermieden werden. Auch abgegrenzte Db2-Bezeichner müssen vermieden werden.

Zuweisung von Standardwerten - Generierung von DDMs im Batch-Modus

Um das Auftreten von Fenstern mit Benutzerinteraktion während der DDM-Feldgenerierung zu vermeiden, kann der in der Library SYSDB2 enthaltene User-Exit `NDBDDM-1` und sein Datenbereich `NDBDDM-L` verwendet werden. Detaillierte Informationen zur Handhabung von `NDBDDM-1` finden Sie in den Anmerkungen im Quellcode. Siehe auch *Eine User Exit Routine verfügbar machen*.

DBID/FNR- Zuweisung

Wenn die Funktion **Generate DDM from an SQL Table** für eine Tabelle aufgerufen wird, für die zum ersten Mal ein DDM generiert werden soll, wird der Bildschirm **DBID/FNR Assignment** angezeigt. Wenn ein DDM für eine Tabelle generiert werden soll, für die bereits ein DDM existiert, werden die vorhandene DBID und FNR verwendet und der Bildschirm **DBID/ FNR-Assignment** erscheint nicht.

Geben Sie auf dem Bildschirm **DBID/FNR-Assignment** eine der bei der Natural-Installation gewählten Datenbankkennungen (DBIDs) und die Dateinummer (FNR) ein, die der Db2-Tabelle zugewiesen werden soll. Natural benötigt diese Angaben nur zu Identifikationszwecken.

Der Bereich der DBIDs, der für Db2-Tabellen reserviert ist, wird im `NTDB`-Makro des Natural-Parametermoduls (siehe Natural *Parameter Referenz*-Dokumentation) für den Datenbanktyp Db2 angegeben. Jede DBID, die nicht in diesem Bereich liegt, wird nicht akzeptiert. Die FNR kann eine beliebige gültige Dateinummer innerhalb der Datenbank sein (zwischen `1` und `65535`).

Nachdem eine gültige DBID und FNR zugewiesen worden sind, wird automatisch ein DDM aus der angegebenen Tabelle erzeugt.

Generierung langer Felder

Die von Natural unterstützte maximale Feldlänge beträgt 1 GB-1 (1073741823 Bytes). Wenn eine Db2-Tabelle eine Spalte enthält, die länger als 253 Bytes ist, oder wenn eine Db2-Spalte als Db2-LOB-Feld definiert ist, wird automatisch das Fenster **Long Field Generation** aufgerufen. Ein Db2-LOB-Feld kann als einfache Natural-Variable mit einer maximalen Länge von 1 GB-1 oder als dynamische Natural-Variable definiert werden.

Ein Feld, das länger als 253 Bytes ist und kein Db2-LOB-Feld ist, kann als einfaches Natural-Feld mit einer maximalen Länge von 1GB-1 oder als Array definiert werden. Im DDM wird ein solches Array als eine multiple Variable dargestellt.

Wenn z.B. eine Db2-Spalte eine Länge von 2000 Bytes hat, können Sie eine Array-Elementlänge von 200 Bytes angeben, und Sie erhalten ein multiples Feld mit 10 Ausprägungen, jede Ausprägung mit einer Länge von 200 Bytes.

Da generierte lange Felder keine multiplen Felder im Sinne von Natural sind, ergibt die Natural-C*-Notation hier keinen Sinn und wird daher nicht unterstützt.

Wenn ein solches generiertes langes Feld in einer Natural-Datensicht (View) definiert ist, die von Natural-SQL-Statements referenziert werden soll (d. h. von Host-Variablen, die multiple Felder darstellen), muss der angegebene Bereich von Ausprägungen (Indexbereich) sowohl bei der Definition als auch bei der Referenzierung immer mit Ausprägung 1 beginnen. Wenn nicht, wird ein Natural-Syntaxfehler zurückgegeben.

Beispiel:

```
UPDATE table SET varchar = #arr(*)
SELECT ... INTO #arr(1:5)
```

Anmerkung: Wenn ein solches generiertes langes Feld mit dem Natural DML `UPDATE`-Statement aktualisiert wird, muss darauf geachtet werden, dass jede Ausprägung ordnungsgemäß aktualisiert wird.

Längenindikator für Felder mit variabler Länge: VARBINARY, VARCHAR, LONG VARCHAR, VARGRAPHIC, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB

Für jeden der oben aufgeführten Spaltentypen wird im DDM ein zusätzliches Längenindikatorfeld (Format/Länge I2 bzw. I4 für LOB-Felder) generiert. Die Länge wird immer in der Anzahl der Zeichen und nicht in Bytes gemessen. Um die Anzahl der Bytes eines `VARGRAPHIC`-, `LONG VARGRAPHIC`- oder `DBCLOB`-Feldes zu erhalten, muss die Länge mit 2 multipliziert werden.

Der Name eines Längenindikatorfeldes beginnt mit L@, gefolgt vom Namen des entsprechenden Feldes. Der Wert des Längenindikatorfelds kann von einem Natural-Programm überprüft oder aktualisiert werden.

Ist das Längenindikatorfeld nicht Teil der Natural-View und handelt es sich bei dem entsprechenden Feld um ein redefiniertes langes Feld, wird die Länge dieses Feldes bei `UPDATE`- und `STORE`-Operationen ohne nachgestellte Leerzeichen berechnet.

Nullwerte

Mit Natural ist es möglich, zwischen einem Wert NULL und dem tatsächlichen Wert Null (0) oder Blank (Leerzeichen) in einer Db2-Spalte zu unterscheiden.

Wenn ein Natural DDM aus dem Db2-Katalog generiert wird, wird ein zusätzliches NULL-Indikatorfeld für jede Spalte generiert, die NULL sein kann, d.h. für die weder NOT NULL noch NOT NULL WITH DEFAULT angegeben ist.

Der Name des NULL-Indikatorfeldes beginnt mit N@, gefolgt von dem Namen des entsprechenden Feldes.

Wenn die Spalte aus der Datenbank gelesen wird, enthält das entsprechende Indikatorfeld entweder Null (0) (wenn die Spalte einen Wert enthält, einschließlich des Wertes 0 oder Leerzeichen) oder -1 (wenn die Spalte keinen Wert enthält).

Beispiel:

Die Spalte NULLCOL CHAR(6) in einer Db2-Tabellendefinition ergibt die folgenden View-Felder führen:

```
NULLCOL          A 6.0
N@NULLCOL    I 2.0
```

Wenn das Feld NULLCOL aus der Datenbank gelesen wird, enthält das zusätzliche Feld N@NULLCOL:

- 0 (Null), wenn NULLCOL einen Wert enthält (einschließlich des Wertes 0 oder Leerzeichen),
- -1 (minus Eins), wenn NULLCOL keinen Wert enthält.

Ein Nullwert kann in einem Datenbankfeld gespeichert werden, indem -1 als Eingabe für das entsprechende NULL-Indikatorfeld eingegeben wird.

Anmerkung: Wenn eine Spalte NULL ist, wird ein impliziter RESET für das entsprechende Natural-Feld durchgeführt.

Locator-Feld für LOB-Spalte

Für jede LOB-Spalte wird ein zusätzliches Locator-Feld im Format I4 generiert.

Ein LOB Locator kann verwendet werden, um einen LOB-Wert im Db2-Datenbankserver zu referenzieren, wenn ein LOB-Wert nicht lokal in einem Programm benötigt wird.

Spalten einer SQL-Tabelle auflisten – Funktion: List Columns

Diese Funktion listet alle Spalten einer bestimmten Db2-Tabelle auf.

Um die Funktion List Columns aufzurufen:

- Geben Sie im Bildschirm **SQL Services Menu** den Funktionscode `L` sowie den Namen und den Ersteller (Creator) der Tabelle ein, deren Spalten Sie auflisten möchten, und drücken Sie Enter.

  Der Bildschirm **List Columns** für diese Tabelle wird aufgerufen. Er listet alle Spalten der angegebenen Tabelle auf und zeigt für jede Spalte die folgenden Informationen an:

Variable	Inhalt	
`Name`	Db2-Name der Spalte.	
`Type`	Der Spaltentyp.	
`Length`	Die Länge (oder Genauigkeit, wenn der Typ `DECIMAL` ist) der Spalte, wie im Db2-Katalog definiert.	
`Scale`	Die Dezimalskalierung der Spalte (nur anwendbar, wenn der Typ `DECIMAL` ist).	
`Update`	`Y`	Die Spalte kann aktualisiert werden.
	`N`	Die Spalte kann nicht aktualisiert werden.
`Nulls`	`Y`	Die Spalte kann Nullwerte enthalten.
	`N`	Die Spalte kann keine Nullwerte enthalten.
`Not`	Eine Spalte, deren Skalierungslänge oder deren Typ von Natural nicht unterstützt wird, wird mit einem Stern (*) gekennzeichnet. Für eine solche Spalte kann kein View-Feld generiert werden. Die maximal unterstützte Skalierungslänge beträgt 7 Bytes. Die folgenden SQL-Typen werden unterstützt: `BIGINT`, `BINARY`, `VARBINARY`, `DECFLOAT`, `XML` `CHAR`, `VARCHAR`, `LONG VARCHAR`, `GRAPHIC`, `VARGRAPHIC`, `LONG VARGRAPHIC`, `DECIMAL`, `INTEGER`, `SMALLINT`, `DATE`, `TIME`, `TIMESTAMP`, `FLOAT`, `ROWID`, `BLOB`, `CLOB` und `DBCLOB`.	

  Die Datentypen `DATE`, `TIME`, `TIMESTAMP`, `FLOAT` und `ROWID` werden in numerische oder alphanumerische Felder unterschiedlicher Länge konvertiert: `DATE` in A10, `TIME` in A8, `TIMESTAMP` in A26, `FLOAT` in F8 und `ROWID` in A40. `DATE` und `TIME` können alternativ auf Natural `DATE` bzw. Natural `TIME` abgebildet werden.

  Für Db2 bietet Natural eine Db2 `TIMESTAMP`-Spalte als alphanumerisches Feld (A26) im Format *`YYYY-MM-DD-HH.II.SS.MMMMMM`*. Alternativ können Sie das Natural `TIME`-Feld (Datenformat T) als Db2 `TIMESTAMP`-Datentyp generieren, wenn die Option `DBTSTI` des Systemkommandos `COMPOPT` auf `ON` gesetzt ist (siehe *Systemkommandos*-Dokumentation).

  Sie können das Natural-Subprogramm `NDBSTMP` verwenden, um `TIMESTAMP`-Felder (A26) zu berechnen.

Eine User Exit Routine verfügbar machen

Sie können die Maske **Generating Natural Data Definition Modules (DDMs)** mit der User-Exit-Routine `NDBDDM-1` oder `NDBDDM-2` anpassen.

Um die User-Exit-Routine NDBDDM-1 oder NDBDDM-1 verfügbar zu machen:

1 Katalogisieren Sie das Quellcode-Objekt `NDBDDM-`*num* in der Library SYSDB2 unter dem Namen `NDBDDMU`*num*.

Anmerkung: Die Namen des Quellcode-Objekts und des katalogisierten Objekts der User-Exit-Routine müssen unterschiedlich sein, damit beim Überschreiben des Quellcode-Objekts bei einer Update-Installation das katalogisierte Objekt nicht verändert wird.

2 Kopieren Sie `NDBDDMU`*num* in die Steplib SYSLIBS.

Ein von der Utility SYSDDM verwendetes Subprogramm sucht in der Steplib SYSLIBS nach `NDBDDMU`*num*.

15 Dynamische und statische SQL-Unterstützung

Dieses Kapitel beschreibt die von Natural geleistete dynamische und statische SQL-Unterstützung.

Verwandte Dokumentation

- Eine Liste der Fehlermeldungen, die bei der statischen Generierung ausgegeben werden können, finden Sie unter *Static Generation Messages and Codes Issued under NDB* in der Natural-*Messages and Codes*-Dokumentation.
- Informationen zu Static SQL mit Natural Security finden Sie unter ***Integration mit Natural Security***.

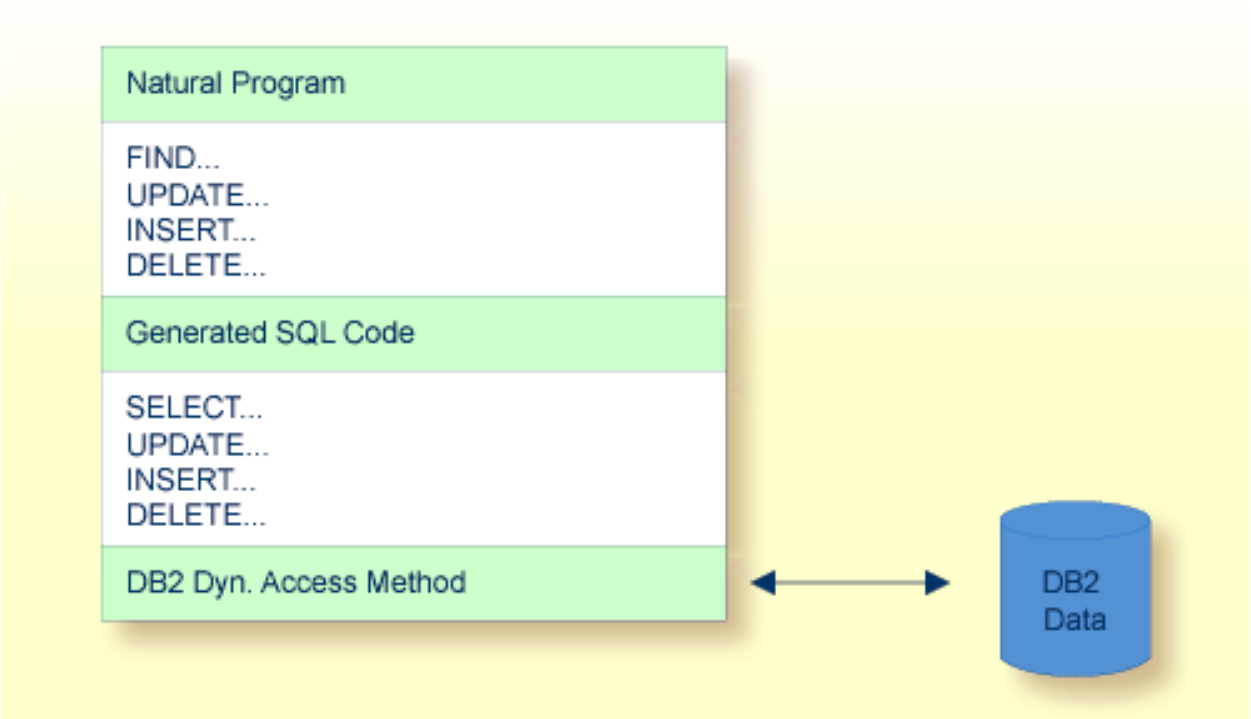
SQL-Unterstützung – Allgemeine Informationen

Die SQL-Unterstützung von Natural kombiniert die Flexibilität der dynamischen SQL-Unterstützung mit der hohen Leistungsfähigkeit der statischen SQL-Unterstützung.

Im Gegensatz zur statischen SQL-Unterstützung muss bei der dynamischen SQL-Unterstützung von Natural keine besondere Rücksicht auf den Betrieb der SQL-Schnittstelle genommen werden. Alle SQL-Statements, die zur Ausführung einer Anwendungsanfrage erforderlich sind, werden automatisch generiert und können mit dem Natural-Kommando `RUN` sofort ausgeführt werden. Bevor Sie ein Programm ausführen, können Sie sich den generierten SQLCODE mit dem Kommando `LISTSQL` ansehen.

Der Zugriff auf Db2 über Natural erfolgt unabhängig davon, ob die dynamische oder die statische SQL-Unterstützung verwendet wird, in derselben Form. Bei statischer SQL-Unterstützung können dieselben SQL-Statements in einem Natural-Programm sowohl im dynamischen als auch im statischen Modus ausgeführt werden. Ein SQL-Statement kann innerhalb eines Natural-Programms kodiert und zu Testzwecken mit dynamischem SQL ausgeführt werden. Ist der Test erfolgreich, bleibt das SQL-Statement unverändert und es kann statisches SQL für dieses Programm generiert werden.

Während der Anwendungsentwicklung arbeitet der Programmierer also im dynamischen Modus und alle SQL-Statements werden dynamisch ausgeführt, während statisches SQL nur für Anwendungen erstellt wird, die in den Produktionsstatus überführt wurden.



Interne Behandlung dynamischer Statements

Natural sorgt automatisch für die Vorbereitung und Ausführung jedes SQL-Statements und verwaltet das Öffnen und Schließen von Cursors, die zum Scannen einer Tabelle verwendet werden.

Die folgenden Themen werden behandelt:

- E/A-Modul NDBIOMO für die dynamische Ausführung von SQL-Statements
- Statement-Tabelle
- Verarbeitung von SQL-Statements, die von Natural ausgegeben werden

E/A-Modul NDBIOMO für die dynamische Ausführung von SQL-Statements

Da jede dynamische Ausführung eines SQL-Statements ein statisch definiertes `DECLARE STATEMENT` und `DECLARE CURSOR`-Statement erfordert, wird ein spezielles E/A-Modul namens `NDBIOMO` bereitgestellt, das eine feste Anzahl dieser Statements und Cursors enthält. Diese Anzahl wird bei der Generierung des `NDBIOMO`-Moduls im Rahmen des Natural for Db2-Installationsvorgangs festgelegt.

Statement-Tabelle

Ein SQL-Statement wird nach Möglichkeit nur einmal vorbereitet und kann dann bei Bedarf mehrfach ausgeführt werden. Zu diesem Zweck pflegt Natural intern eine Tabelle mit allen vorbereiteten SQL-Statements und ordnet jedes dieser Statements einem DECLAREd STATEMENT im Modul NDBIOMO zu. Außerdem werden in dieser Tabelle die von den SQL-Statements FETCH, UPDATE (positioned) und DELETE (positioned) verwendeten Cursor verwaltet.

Jedes SQL-Statement wird eindeutig identifiziert durch:

- den Namen des Natural-Programms, das dieses SQL-Statement enthält,
- die Zeilennummer des SQL-Statements in diesem Programm,
- den Namen der Natural Library, in der dieses Programm per Stow abgelegt wurde,
- den Zeitstempel, wann dieses Programm per Stow abgelegt wurde.

Ein vorbereitetes SQL-Statement kann mit Hilfe des dynamischen SQL-Statements EXECUTE USING DESCRIPTOR oder OPEN CURSOR USING DESCRIPTOR mehrmals mit unterschiedlichen Variablenwerten ausgeführt werden.

Wenn das Fassungsvermögen der Statement-Tabelle erschöpft ist, überschreibt der Eintrag für das nächste vorbereitete Statement den Eintrag für ein freies Statement, dessen letzte Ausführung am wenigsten weit zurückliegt.

Wenn ein neues SELECT-Statement angefordert wird, wird ihm ein freier Eintrag in der Statement-Tabelle mit dem entsprechenden Cursor zugewiesen, und alle nachfolgenden FETCH-, UPDATE- und DELETE-Statements, die sich auf dieses SELECT-Statement beziehen, verwenden diesen Cursor. Nach Beendigung der sequenziellen Abfrage der Tabelle wird der Cursor freigegeben und für eine weitere Zuweisung verfügbar. Solange der Cursor geöffnet ist, wird der Eintrag in der Statement-Tabelle als benutzt markiert und kann nicht von einem anderen Statement wiederverwendet werden.

Wenn die Anzahl der verschachtelten FIND- (SELECT-) Statements die Anzahl der in der Statement-Tabelle verfügbaren Einträge erreicht, wird jedes weitere SQL-Statement zur Ausführungszeit abgelehnt und eine Natural-Fehlermeldung zurückgegeben.

Die Größe der Statement-Tabelle hängt von der für das Modul NDBIOMO angegebenen Größe ab. Da die Statement-Tabelle im Pufferbereich von Db2 enthalten ist, reicht die Einstellung des Natural-Profilparameters DB2SIZE (siehe auch *Natural Parameter Modifications for Natural for DB2* in *Installing Natural for DB2 on z/OS* in der *Installation*-Dokumentation) möglicherweise nicht aus und muss erhöht werden.

Verarbeitung von SQL-Statements, die von Natural ausgegeben werden

Das eingebettete SQL verwendet die Cursor-Logik zur Verarbeitung von SELECT-Statements. Die Vorbereitung und Ausführung eines SELECT-Statements läuft wie folgt ab:

1. Das typische SELECT-Statement wird durch einen Programmablauf vorbereitet, der die folgenden eingebetteten SQL-Statements enthält (dabei sind *X* und *SQLOBJ* SQL-Variablen, keine Programm-Labels):

```
DECLARE SQLOBJ STATEMENT
DECLARE X CURSOR FOR SQLOBJ
INCLUDE SQLDA (copy SQL control block)
```

   Dann wird das folgende Statement in die SQLSOURCE verschoben:

```
SELECT PERSONNEL_ID, NAME, AGE
 FROM EMPLOYEES
 WHERE NAME IN (?, ?)
    AND AGE BETWEEN ? AND ?
```

   Anmerkung: Die Fragezeichen (?) oben sind Parametermarkierungen, die angeben, wo Werte zur Ausführungszeit eingefügt werden sollen.

```
PREPARE SQLOBJ FROM SQLSOURCE
```

2. Then, the SELECT statement is executed as follows:

```
OPEN X USING DESCRIPTOR SQLDA
FETCH X USING DESCRIPTOR SQLDA
```

   Der Deskriptor SQLDA wird verwendet, um eine variable Liste von Programmbereichen anzugeben. Wenn das OPEN-Statement ausgeführt wird, enthält es die Adresse, die Länge und den Typ jedes Wertes, der eine Parametermarkierung in der WHERE-Klausel des SELECT-Statements ersetzt. Bei der Ausführung des FETCH-Statements enthält es die Adresse, die Länge und den Typ aller Programmbereiche, die aus der Tabelle gelesene Felder erhalten.

   Bei der ersten Ausführung des FETCH-Statements wird die Natural-Systemvariable *NUMBER auf einen Wert ungleich Null gesetzt, wenn mindestens ein Satz gefunden wird, der den Suchkriterien entspricht. Anschließend werden alle Datensätze, die die Suchkriterien erfüllen, durch wiederholte Ausführung der FETCH-Statements gelesen.

   Zur Verbesserung der Performance, insbesondere bei der Verwendung verteilter Datenbanken, kann die Db2-spezifische FOR FETCH ONLY-Klausel verwendet werden. Diese Klausel wird generiert und ausgeführt, wenn nur Zeilen abgerufen werden sollen, d.h. wenn keine Änderungen vorgenommen werden sollen.

3. Wenn alle Datensätze gelesen wurden, wird der Cursor durch die Ausführung des folgenden Statements freigegeben:

```
CLOSE X
```

Programme für die statische Ausführung vorbereiten

In diesem Abschnitt wird beschrieben, wie Sie Natural-Programme für die statische Ausführung vorbereiten.

Die folgenden Themen werden behandelt:

- Grundsätzliches
- Generierungsprozedur – Kommando: CMD CREATE
- Vorkompilierung des generierten Assembler-Programms
- Änderungsprozedur – Kommando: CMD MODIFY
- Vorkompiliertes DBRM einbinden – Kommando BIND

Eine Erklärung der Symbole, die in diesem Abschnitt zur Beschreibung der Syntax von Natural-Statements verwendet werden, finden Sie unter *Syntax-Symbole* in der *Statements*-Dokumentation.

Grundsätzliches

Statisches SQL wird im Natural-Batch-Modus für eine oder mehrere Natural-Anwendungen generiert, die aus einem oder mehreren Natural-Objektprogrammen bestehen können. Die Anzahl der Programme, die in einem Durchlauf des Generierungsverfahrens für die statische Ausführung modifiziert werden können, ist auf 999 begrenzt.

Während des Generierungsvorgangs werden die in den angegebenen Natural-Objekten enthaltenen Datenbankzugriffs-Statements extrahiert, in Arbeitsdateien geschrieben und in ein temporäres Assembler-Programm umgewandelt. Wenn kein Natural-Programm gefunden wird, das einen SQL-Zugriff enthält, oder wenn bei der statischen SQL-Generierung ein Fehler auftritt, bricht das Batch Natural ab und gibt den Bedingungscode 40 zurück, was bedeutet, dass alle weiteren JCL-Schritte nicht mehr ausgeführt werden dürfen.

Die Natural-Module `NDBCHNK` und `NDBSTAT` müssen sich in einer Steplib des Generierungsschrittes befinden. Beide werden bei der Ausführung des Generierungsschrittes dynamisch geladen.

Das temporäre Assembler-Programm wird in eine temporäre Datei (das Natural Workfile `CMWKF06`) geschrieben und vorkompiliert. Die Größe dieser Arbeitsdatei ist proportional zur maximalen Anzahl der Programme, der Anzahl der SQL-Statements und der Anzahl der in den SQL-Statements verwendeten Variablen. Während des Vorkompilierungsschritts wird ein Datenbankanforderungsmodul (DBRM) erstellt, und nach dem Vorkompilierungsschritt wird die Precompiler-Ausgabe aus dem Assembler-Programm extrahiert und in die entsprechenden Natural-Objekte geschrieben,

was bedeutet, dass die Natural-Objekte für die statische Ausführung modifiziert (vorbereitet) werden. Das temporäre Assembler-Programm wird nicht mehr verwendet und gelöscht.

Ein statisches Datenbankanforderungsmodul wird entweder mit dem auf dem Installationsmedium mitgelieferten Beispiel-Job oder mit einem entsprechenden Job, der mit der Funktion **Create DBRM** erstellt wurde, erstellt.

Generierungsprozedur – Kommando: CMD CREATE

Um statisches SQL für Natural-Programme zu generieren:

- Statisches SQL für Natural-Programme generieren
- Statischer Name
- USING-Klausel

Statisches SQL für Natural-Programme generieren

Um statisches SQL für Natural-Programme zu generieren:

1 Melden Sie sich bei der Natural System Library SYSDB2 an.

Da bei der Installation von Natural for Db2 eine neue SYSDB2-Library angelegt wurde, müssen Sie sicherstellen, dass diese alle Predict-Schnittstellenprogramme enthält, die für die Generierung von statischem SQL erforderlich sind. Diese Programme werden zum Zeitpunkt der Predict-Installation in SYSDB2 geladen (siehe die entsprechende *Predict*-Produktdokumentation).

2 Geben Sie das Kommando `CMD CREATE` und die für die statische SQL-Generierung erforderlichen Natural-Eingaben an. Das Kommando `CMD CREATE` hat die folgende Syntax:

```
CMD CREATE DBRM static-name USING using-clause
{application-name,object-name,excluded-object}
:
:
```

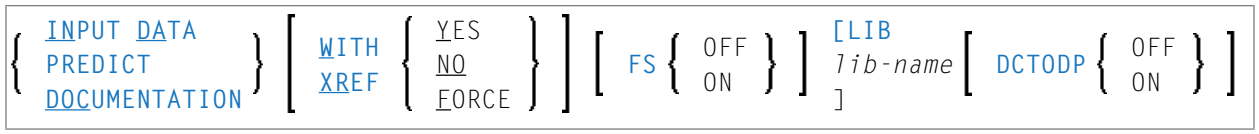
Die Generierungsprozedur liest die angegebenen Natural-Objekte, verändert sie aber nicht. Wenn eines der angegebenen Programme nicht gefunden wurde oder keinen SQL-Zugriff hatte, wird am Ende des Generierungsschritts der Rückmeldecode 4 zurückgegeben.

Statischer Name

Wenn die Option `PREDICT DOCUMENTATION` verwendet werden soll, muss ein entsprechender statischer Predict-SQL-Eintrag vorhanden sein und der *static-name* muss dem Namen dieses Eintrags entsprechen. Außerdem muss der *static-name* dem Namen des DBRM entsprechen, das bei der Vorkompilierung erzeugt werden soll. Der *static-name* kann bis zu 8 Zeichen lang sein und muss den Assembler-Namenskonventionen entsprechen.

USING-Klausel

Die *using-clause* gibt die Natural-Objekte an, die im DBRM enthalten sein sollen. Diese Objekte können entweder explizit als `INPUT DATA` in der JCL angegeben werden oder als `PREDICT DOCUMENTATION` von Predict bezogen werden.



Wenn die anzugebenden Parameter nicht in eine Zeile passen, geben Sie die Kommandokennung (`CMD`) und die verschiedenen Parameter in separaten Zeilen an und verwenden Sie sowohl das Eingabetrennzeichen (wie mit dem Natural-Profil-/Session-Parameter `ID` angegeben - Standardeinstellung ist das Komma (`,`) - als auch das Fortsetzungszeichen - wie mit dem Natural-Profil-/Session-Parameter `CF` angegeben - Standardeinstellung ist das Prozentzeichen (`%`) - wie im folgenden Beispiel gezeigt:

Beispiel:

```
CMD
CREATE,DBRM,static,USING,PREDICT,DOCUMENTATION,WITH,XREF,NO,%
LIB,library
```

Alternativ können Sie auch Abkürzungen verwenden, wie im folgenden Beispiel gezeigt:

Beispiel:

```
CMD CRE DBRM static US IN DA W XR Y FS OFF LIB library
```

Die Reihenfolge der Parameter `USING`, `WITH`, `FS` und `LIB` ist optional.

INPUT DATA

Als Eingabedaten müssen in den nachfolgenden Zeilen des Jobs die Anwendungen und die Namen der Natural-Objekte angegeben werden, die in das DBRM aufgenommen werden sollen (*application-name*,*object-name*). Eine Teilmenge dieser Objekte kann auch wieder ausgeschlossen werden (*excluded-objects*). Objekte in Libraries, deren Namen mit `SYS` beginnen, können ebenfalls für die statische Generierung verwendet werden.

Die Anwendungen und Namen von Natural-Objekten müssen durch das Eingabetrennzeichen getrennt werden - wie mit dem Natural-Profilparameter `ID` angegeben - Standard ist ein Komma (`,`). Wenn Sie alle Objekte angeben möchten, deren Namen mit einer bestimmten Zeichenfolge beginnen, verwenden Sie einen *`object-name`* oder einen *`excluded-objects`*, der mit einem Stern (*) endet. Um alle Objekte in einer Anwendung anzugeben, verwenden Sie nur die Stern-Notation (*).

Beispiel:

```
LIB1,ABC*
  LIB2,A*,AB*
  LIB2,*
  :
  .
```

Die Angabe von Anwendungen/Objekten muss durch eine Zeile abgeschlossen werden, die nur einen Punkt (`.`) enthält.

PREDICT DOCUMENTATION

Da Predict statisches SQL für Db2 unterstützt, können Sie Predict auch die Eingabedaten für die Erstellung von statischem SQL liefern lassen, indem Sie bereits vorhandene `PREDICT DOCUMENTATION` verwenden.

WITH XREF-Option

Da Predict Active References statisches SQL für Db2 unterstützt, kann das erzeugte statische DBRM in Predict dokumentiert werden, und die Dokumentation kann mit Natural verwendet und aktualisiert werden.

Wenn die Option `WITH XREF` angegeben ist, können Sie bei jeder Erstellung eines statischen DBRM die Cross-Referenzdaten für einen statischen SQL-Eintrag in Predict speichern (`YES`). Sie können stattdessen angeben, dass keine Cross-Referenzdaten gespeichert werden (`NO`) oder dass geprüft wird, ob bereits ein statischer SQL-Eintrag in Predict für dieses statische DBRM existiert (`FORCE`). Wenn ja, werden die Cross-Referenzdaten gespeichert, wenn nicht, wird die Erstellung des statischen DBRM nicht zugelassen. Nähere Informationen zu Predict Active References finden Sie in der entsprechenden Predict-Dokumentation.

Wenn `WITH XREF (YES/FORCE)` angegeben ist, werden `XREF`-Daten sowohl für den statischen Predict-SQL-Eintrag (falls in Predict definiert) als auch für jedes generierte statische Natural-Programm geschrieben. Die statische Generierung mit `WITH XREF (YES/FORCE)` ist jedoch nur möglich, wenn die entsprechenden Natural-Programme mit `XREF ON` katalogisiert wurden.

Die Option `WITH XREF FORCE` gilt nur für die Option `USING INPUT DATA`.

Anmerkung: Wenn Sie Predict nicht verwenden, muss die Option XREF weggelassen oder auf NO gesetzt werden und das Modul NATXRF2 braucht nicht mit dem Natural-Kern verknüpft zu werden.

FS-Option

Wenn die Option FS (File Server) auf ON gesetzt ist, wird ein zweites SELECT für den Natural File Server für Db2 erzeugt. ON ist die Standardeinstellung.

Wenn die FS-Option auf OFF gesetzt ist, wird kein zweites SELECT generiert, was dazu führt, dass weniger SQL-Statements in Ihrem statischen DBRM generiert werden und somit ein kleineres DBRM entsteht.

LIB-Option

Mit der Option LIB (Library) kann eine andere Predict-Library als die Standard-Library (*SYSSTA*) angegeben werden, die den statischen Predict-SQL-Eintrag und die XREF-Daten enthält. Der Name der Library kann bis zu acht Zeichen lang sein.

DCTODP-Option

Die Option DCTODP ist nur relevant, wenn Sie eine statische Generierung für Programme durchführen, die mit dem Natural-Parameter DC=',' katalogisiert wurden.

Mit dem DCTODP-Parameter kann festgelegt werden, ob die statische Generierung dezimale Literale in SQL-Statements durch das Ersetzen des Dezimalzeichens Komma (,) durch das Dezimalzeichen Punkt (.) im generierten statischen SQL-Assembler-Programm ändern soll. Dies ist notwendig, da der Db2-Precompiler kein Komma in dezimalen Literalen unterstützt.

Wenn DCTODP OFF angegeben ist, findet keine Konvertierung von Komma in Punkt statt. DCTODP OFF ist der Standardwert.

Wenn DCTODP ON angegeben ist, wird bei der statischen Generierung ein Komma (,) als Dezimalzeichen in einen Punkt (.) umgewandelt.

Wenn DCTODP ON verwendet wird, sollten Sie sicherstellen, dass Ihre Programme eindeutig kodiert sind, wenn Sie das Komma als Trennzeichen in Funktionsaufrufen und verschiedenen SQL-Klauseln wie IN oder ORDER BY verwenden, damit das Komma als Trennzeichen in Elementlisten nicht als Dezimalpunkt eines Dezimal-Literales fehlinterpretiert werden kann.

Wenn Sie zum Beispiel eine SQL IN-Klausel unter Verwendung von DC=',' wie folgt kodieren:

Column-name IN (10,20,30,40).

Die statische Generierung mit DCTODP ON ändert die IN-Klausel in

Column-name IN (10,20 , 30,40).

Die IN-Liste enthält zwei Werte 10.20 und 30.40.

Die gleiche IN-Klausel bei statischer Generierung mit DCTODP OFF (Standardwert) wird geändert in

Column-name IN (10,20 , 30,40).

Die IN-Liste enthält vier Werte

Wenn Sie Folgendes codiert hätten:

Column-name IN (10 , 20 , 30 , 40).

Die statische Generierung generiert die IN-Klausel immer mit DCTODP entweder ON oder OFF zu

Column-name IN (10 , 20 , 30 , 40).

Vorkompilierung des generierten Assembler-Programms

In diesem Schritt wird der Precompiler aufgerufen, um das generierte temporäre Assembler-Programm vorzukompilieren. Die Ausgabe des Precompilers besteht aus dem DBRM und einem vorkompilierten temporären Assembler-Programm, das alle Datenbankzugriffs-Statements enthält, die von SQL-Statements in Assembler-Statements umgewandelt wurden.

Später dient das DBRM als Input für den BIND-Schritt und das Assembler-Programm als Input für den Änderungsschritt.

Änderungsprozedur – Kommando: CMD MODIFY

Der Änderungsvorgang ändert die beteiligten Natural-Objekte, indem er Precompiler-Informationen in das Objekt schreibt und den Objekt-Header mit dem *static-name* markiert, der mit dem Kommando CMD CREATE angegeben wurde.

Darüber hinaus werden alle vorhandenen Kopien dieser Objekte im globalen Natural-Buffer Pool (falls vorhanden) gelöscht und XREF-Daten in Predict geschrieben (falls beim Generierungsvorgang angegeben).

Um den Änderungsvorgang auszuführen:

1 Melden Sie sich bei der Natural Library SYSDB2 an.

2 Geben Sie das Kommando CMD MODIFY mit der folgenden Syntax ein:

```
CMD MODIFY [XREF]
```

Der Input für den Modifizierungsschritt ist die Precompiler-Ausgabe, die sich in einem Datensatz befinden muss, der als Natural-Arbeitsdatei `CMWKF01` definiert ist.

Die Ausgabe besteht aus Precompiler-Informationen, die in die entsprechenden Natural-Objekte geschrieben werden. Außerdem wird eine Meldung zurückgegeben, die angibt, ob ein Objekt zum ersten Mal für die statische Ausführung modifiziert wurde (`modified`) oder ob es bereits zuvor modifiziert wurde (`re-modified`).

Assembler/Natural-Cross-Referenzen

Wenn Sie die Option `XREF` des Kommandos `CMD MODIFY` angeben, wird eine Ausgabeliste in der Arbeitsdatei `CMWKF02` erstellt, die den DBRM-Namen und die Assembler Statement-Nummer jedes statisch erzeugten SQL-Statements zusammen mit der entsprechenden Natural-Quellcode-Zeilennummer, dem Programmnamen, dem Library-Namen, der Datenbankkennung und der Dateinummer enthält.

```
-------------------------------------------------------------------------....
  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     ....
```

Spalte	Erläuterung
`DBRMNAME`	Name des DBRM, das das statische SQL-Statement enthält.
`STMTNO`	Assembler-Statement-Nummer des statischen SQL-Statements.
`LINE`	Entsprechende Natural-Quellcode-Zeilennummer.
`NATPROG`	Name des Natural-Programms, das das statische SQL-Statement enthält.
`NATLIB`	Name der Natural Library, die das Natural-Programm enthält.
`DB / FNR`	Natural-Datenbankkennung und -Dateinummer.
`COMMENT`	Typ des SQL-Statements, wobei 2ND anzeigt, dass das entsprechende Statement für eine erneute Auswahl verwendet wird. Siehe auch *File Server-Konzept*.

Vorkompiliertes DBRM einbinden – Kommando BIND

Wir empfehlen Ihnen, das Db2-Kommando `BIND` nach dem Kommando `CMD MODIFY` auszuführen.

Das Db2-Kommando `BIND` bindet das DBRM in ein Db2 Package ein. Sie können ein oder mehrere Db2-Packages in einen Db2-Anwendungsplan einbinden. Zusätzlich zu den Packages der statischen DBRMs, die mit dem Kommando `CMD CREATE` erstellt wurden, kann dieser Anwendungsplan auch das Package der DBRM des `NDBIOMO`-Moduls enthalten, das Natural für die dynamische SQL-Ausführung bereitstellt.

Ein DBRM kann in eine beliebige Anzahl von Packages eingebunden werden und die Packages können bei Bedarf in eine beliebige Anzahl von Anwendungsplänen eingebunden werden. Ein Plan ist physisch unabhängig von der Umgebung, in der das Programm ausgeführt werden soll. Sie können Ihre Packages jedoch logisch in Plänen gruppieren, die entweder für die Batch- oder die Online-Verarbeitung verwendet werden sollen, wobei dasselbe Package sowohl Teil eines Batch-Plans als auch eines Online-Plans sein kann.

Sofern Sie keine Planumschaltung verwenden, kann nur ein Plan pro Natural-Sitzung ausgeführt werden. Daher müssen Sie sicherstellen, dass der im `BIND`-Schritt angegebene Plan-Name mit dem Namen übereinstimmt, der für die Ausführung von Natural verwendet wird.

Natural im statischen Modus

Um Natural im statischen Modus ausführen zu können, müssen alle Natural-Benutzer das Db2-Privileg `EXECUTE PLAN/PACKAGE` für den im `BIND`-Schritt erstellten Plan besitzen.

Um statisches SQL auszuführen, müssen Sie Natural starten und das entsprechende Natural-Programm ausführen. Intern wertet die Natural-Laufzeitschnittstelle die in das Natural-Objekt geschriebenen Precompiler-Daten aus und führt dann die statischen Zugriffe aus.

Für den Benutzer gibt es keinen Unterschied zwischen dynamischer und statischer Ausführung.

Natural im gemischten dynamischen/statischen Modus

Es ist möglich, Natural in einem gemischten statischen und dynamischen Modus zu betreiben, wobei für einige Programme statisches SQL generiert wird und für andere nicht.

Der Modus, in dem ein Programm ausgeführt wird, wird durch das Natural-Objektprogramm selbst bestimmt. Wenn im ausführenden Programm ein statisches DBRM referenziert wird, werden alle Statements in diesem Programm im statischen Modus ausgeführt.

Anmerkung: Natural-Programme, die einen Laufzeitfehler zurückgeben, werden nicht automatisch im dynamischen Modus ausgeführt. Stattdessen muss entweder der Fehler

behoben werden oder das Natural-Programm muss vorübergehend neu katalogisiert werden, um im dynamischen Modus ausgeführt werden zu können.

Innerhalb ein und derselben Natural-Sitzung können statische und dynamische Programme ohne weitere Angaben gemischt werden. Die Entscheidung, welcher Modus verwendet werden soll, wird von jedem einzelnen Natural-Programm getroffen.

Meldungen und Codes

Eine Liste der Fehlermeldungen, die während der statischen Generierung ausgegeben werden können, finden Sie unter *Static Generation Messages and Codes Issued under NDB* in der Natural *Messages and Codes*-Dokumentation.

Anwendungspläne wechseln

In diesem Abschnitt wird beschrieben, wie Sie Anwendungspläne innerhalb derselben Natural-Sitzung in verschiedenen TP-Monitorumgebungen oder im Batch-Modus wechseln können.

Die folgenden Themen werden behandelt:

Grundsätzliches zum Planwechsel

Durch „Application Plan Switching“ können Sie innerhalb einer Natural-Sitzung auf einen anderen Anwendungsplan umschalten.

Soll ein zweiter Anwendungsplan verwendet werden, so kann dieser durch Ausführen des Natural-Programms `NATPLAN` festgelegt werden. `NATPLAN` ist in der Natural System Library SYSDB2 enthalten und kann entweder aus einem Natural-Programm heraus oder dynamisch durch Eingabe des Kommandos `NATPLAN` an der `NEXT`-Eingabeaufforderung aufgerufen werden. Der einzige Eingabewert, der für `NATPLAN` benötigt wird, ist ein achtstelliger Plan-Name. Wenn kein Plan-Name angegeben ist, werden Sie vom System dazu aufgefordert, dies zu tun.

Bevor Sie NATPLAN ausführen, müssen Sie sicherstellen, dass alle offenen Db2-Wiederherstellungseinheiten geschlossen sind.

Da das Programm NATPLAN auch in Quellcodeform zur Verfügung gestellt wird, können benutzerdefinierte Planumschaltprogramme mit ähnlicher Logik erstellt werden.

Der tatsächliche Wechsel von einem Plan zu einem anderen unterscheidet sich in den verschiedenen unterstützten Umgebungen. Die Funktion ist unter Com-plete, CICS und IMS TM MPP verfügbar. Bei Verwendung der Call Attachment Facility (CAF) oder Resource Recovery Services Attachment Facility (RRSAF) ist sie auch in TSO- und Batch-Umgebungen verfügbar.

In einigen dieser Umgebungen muss anstelle eines Plan-Namens eine Transaktionskennung oder ein Code angegeben werden.

Planumschaltung unter CICS

Unter CICS haben Sie die Möglichkeit, entweder die Planumschaltung durch Transaktionskennung (Standard) oder dynamische Planauswahl-Exit-Routinen zu verwenden. Indem Sie das Feld #SWITCH-BY- TRANSACTION-ID im Programm NATPLAN auf TRUE oder FALSE setzen, wird entweder die Subroutine CMTRNSET oder der gewünschte Plan-Name in eine temporäre Speicherwarteschlange geschrieben.

Weitere Informationen zur Aktivierung der Planumschaltung unter CICS finden Sie unter *Installation Steps Specific to CICS* in der *Installing Natural for DB2 on z/OS*-Dokumentation.

Nachfolgend finden Sie Informationen zu:

- Plan Switching by CICS/DB2 Exit Routine

Plan Switching by CICS/DB2 Exit Routine

Wenn #SWITCH-BY-TRANSACTION-ID auf FALSE gesetzt ist, wird der gewünschte Plan-Name in eine temporäre Speicherwarteschlange für eine CICS/Db2-Exit-Routine geschrieben, die als PLANExit-Attribut eines DB2ENTRY oder der DB2CONN-Definition angegeben ist. Das NATPLAN-Programm muss vor dem ersten Db2-Zugriff aufgerufen werden. Natural for Db2 bietet NDBUEXT als CICS-Db2-Planauswahl-Exit-Programm. Weitere Informationen zu CICS/Db2-Exit-Routinen finden Sie in der entsprechenden IBM-Literatur.

Der Name der temporären Speicherwarteschlange lautet PLAN*xxxx*, wobei *ssss* die Kennung des entfernten oder lokalen CICS-Systems und *tttt* die CICS-Terminalkennung ist.

In einer CICSplex-Umgebung muss die CICS-Warteschlange PLAN*xxxx*, die den Namen des Plans enthält, mit TYPE=SHARED oder TYPE=REMOTE in einem CICS-TST definiert werden.

Für jede neue Db2-Wiederherstellungseinheit wird automatisch die entsprechende Planauswahl-Exit-Routine aufgerufen. Diese Exit-Routine liest den temporären Speichersatz und verwendet den darin enthaltenen Plan-Namen für die Planauswahl.

Wenn für die Natural-Sitzung kein temporärer Speichersatz existiert, kann ein im Plan-Exit enthaltener Standard-Plan-Name verwendet werden. Wenn durch den Exit kein Plan-Name angegeben

ist, wird der Name des Plans verwendet, der dem Namen des statischen Programms (DBRM) entspricht, das den SQL-Aufruf absetzt. Wenn kein solcher Plan-Name existiert, kommt es zu einem SQL-Fehler.

Planumschaltung unter Com-plete

In Com-plete-Umgebungen wird der Planwechsel mit Hilfe der Call Attachment Facility (CAF) durchgeführt, die den verwendeten Thread freigibt und einen anderen mit einem anderen Plan-Namen anhängt.

Sobald die Db2-Verbindung hergestellt ist, wird derselbe Plan-Name so lange verwendet, bis der Plan explizit über die IBM Call Attachment Language-Schnittstelle (`DSNALI`) geändert wird. Weitere Informationen über die CAF-Schnittstelle finden Sie in der einschlägigen IBM-Literatur.

Unter Com-plete gibt das `NATPLAN`-Programm zunächst ein `END TRANSACTION`-Statement aus und ruft dann unter Verwendung von `DB2SERV` eine Assembler-Routine auf.

Die Assembler-Routine führt die eigentliche Umschaltung durch. Sie gibt eine `CLOSE`-Anforderung an `DSNALI` aus, um die Db2-Verbindung zu beenden (falls eine besteht). Anschließend sendet sie eine `OPEN`-Anforderung, um die Db2-Verbindung wiederherzustellen und die für die Ausführung des angegebenen Plans erforderlichen Ressourcen zuzuordnen.

Wenn `NATPLAN` nicht vor dem ersten SQL-Aufruf ausgeführt wurde, wird standardmäßig der Plan verwendet, der in den Startparametern von Com-plete definiert wurde. Sobald ein Plan mit `NDBPLAN` geändert wurde, bleibt er eingeplant, bis ein anderer Plan von `NDBPLAN` eingeplant wird oder bis zum Ende der Natural-Sitzung.

Planumschaltung unter IMS TM

In einer IMS-MPP-Umgebung wird der Wechsel durch direkte oder verzögertes Message Switching erreicht. Da jedem IMS-Anwendungsprogramm ein anderer Anwendungsplan zugeordnet ist, wird beim Message Switching von einem Transaktionscode zu einem anderen automatisch der verwendete Anwendungsplan gewechselt.

Da Natural-Anwendungen direktes oder verzögertes Message Switching durchführen können, indem sie die entsprechenden mitgelieferten Routinen aufrufen, ist die Verwendung des Programms `NATPLAN` für die Planumschaltung optional.

`NATPLAN` ruft die Assembler-Routine `CMDEFSW` auf, die den neuen Transaktionscode festlegt, der bei der nächstfolgenden Terminal-E/A verwendet werden soll.

Planumschaltung unter TSO und im Batch-Modus

In TSO- und Batch-Umgebungen wird der Planwechsel mit Hilfe der Call Attachment Facility (CAF) oder der Resource Recovery Services Attachment Facility (RRSAF) durchgeführt. Beide Einrichtungen geben den verwendeten Thread frei und hängen einen anderen an, der einen anderen Plan-Namen hat.

Nachfolgend finden Sie Informationen über:

- Planauswahl mit CAF
- Planauswahl mit RRSAF

Planauswahl mit CAF

Die initialen Verbindungs- und Planeinstellungen können mit den Subparametern `DB2PLAN` und `DB2SSID` des `NTDB2`-Makros oder des Profilparameters `DB2` vorgenommen werden, ohne das Programm NATPLAN zu verwenden. Die ursprünglichen Einstellungen könnten jedoch durch die Verwendung des Programms `NATPLAN` überschrieben werden.

Bei Verwendung der Call Attachment Facility (CAF) erfolgt die Planauswahl entweder implizit oder explizit. Wenn vor dem ersten SQL-Aufruf noch keine Db2-Verbindung hergestellt wurde, wird ein Plan-Name von Db2 ausgewählt. In diesem Fall ist der verwendete Plan-Name derselbe wie der Name des Programms (DBRM), das den SQL-Aufruf absetzt.

Sobald die Db2-Verbindung hergestellt ist, wird der Plan-Name beibehalten, bis er explizit von der IBM Call Attachment Language Interface (`DSNALI`) geändert wird. Weitere Informationen über die CAF-Schnittstelle finden Sie in der entsprechenden IBM-Literatur.

Under TSO and in batch mode, the `NATPLAN` program first issues an `END TRANSACTION` statement and then invokes an Assembler routine by using `DB2SERV`.

Unter TSO und im Batch-Modus gibt das `NATPLAN`-Programm zuerst ein `END TRANSACTION`-Statement aus und ruft dann unter Verwendung von `DB2SERV` eine Assembler-Routine auf.

Anmerkung: Ändern Sie das `NATPLAN`-Programm, indem Sie das Feld #SSM auf den aktuellen Db2-Subsystem-Namen setzen. Der der Standardname ist `DB2`.

Die Assembler-Routine führt die eigentliche Umschaltung durch. Sie sendet eine `CLOSE`-Anforderung an `DSNALI`, um eine mögliche Db2-Verbindung zu beenden. Anschließend sendet sie eine `OPEN`-Anforderung, um die Db2-Verbindung wiederherzustellen und die für die Ausführung des angegebenen Plans erforderlichen Ressourcen zuzuordnen.

Wenn `NATPLAN` nicht vor dem ersten SQL-Aufruf ausgeführt wurde, erfolgt die Planauswahl durch `DSNALI`. In diesem Fall ist der verwendete Plan-Name derselbe wie der Name des Programms, das den SQL-Aufruf tätigt. Die verwendete Subsystemkennung ist diejenige, die bei der Db2-Installation angegeben wurde. Wenn kein solcher Plan-Name oder keine Subsystemkennung existiert, wird eine Natural-Fehlermeldung zurückgegeben.

Wenn ein statisches DBRM den SQL-Aufruf ausführt, muss ein Plan-Name mit demselben Namen wie der des statischen DBRM vorhanden sein.

Wenn dynamisches SQL verwendet wird, muss ein Db2-Plan existieren, der ein Package mit dem DBRM des NDBIOMO-Moduls enthält. Wenn der Name des Db2-Plans weder im NATPLAN-Programm noch mit dem Schlüsselwort-Subparameter DB2PLAN definiert wurde, verwendet die Db2 Call Attachment Facility (CAF) den Namen des NDBIOMO-DBRM als Standard-Plan-Namen.

Anmerkung: Um Verwirrung bezüglich des gewählten Plan-Namens und/oder der Subsystemkennung zu vermeiden, sollten Sie NATPLAN immer vor dem ersten SQL-Aufruf aufrufen.

Planauswahl mit RRSAF

Die initialen Verbindungs- und Planeinstellungen können mit den Schlüsselwort-Subparametern DB2COLL, DB2GROV, DB2PLAN, DB2SSID und DB2XID des Makros NTDB2 oder des Profilparameters DB2 vorgenommen werden, ohne das Programm NATPLAN zu verwenden. Die initialen Einstellungen können jedoch durch die Verwendung des Programms NATPLAN überschrieben werden.

Bei Verwendung der Resource Recovery Services Attachment Facility (RRSAF) erfolgt die Planauswahl explizit.

RRSAF wird verwendet, wenn das DSNRLI-Schnittstellenmodul von IBM mit Natural verbunden ist. Sobald die Db2-Verbindung hergestellt ist, wird der Name des Plans beibehalten, bis er explizit mit RRSAF geändert wird. Weitere Informationen zu RRSAF finden Sie in der entsprechenden IBM-Literatur.

Das Programm NATPLAN führt die eigentliche Umschaltung durch. Es sendet eine TERMINATE IDENTIFY-Anforderung an DSNRLI, um eine mögliche Db2-Verbindung zu beenden. NATPLAN gibt dann eine IDENTIFY-Anforderung aus, um die Db2-Verbindung wiederherzustellen. Auf diese Anforderung folgen die Anforderungen SIGNON und CREATE THREAD.

In einer RRSAF-Umgebung können bis zu vier der folgenden Parameter in NATPLAN angegeben werden, wobei #PLAN obligatorisch ist:

Parameter	Standardwert	Format	Erläuterung
#PLAN	None	A8	Name des Plans, der für die SQL-Verarbeitung in dem erzeugten Thread (CREATE THREAD) verwendet wird.
#SSM	DB2	A4	Subsystemkennung des angeschlossenen Db2-Servers (IDENTIFY).
#COLLID	COLLID	A18	Wird nur verwendet, wenn das erste Zeichen von #PLAN ein Fragezeichen (?) ist. Collection ID, die für die SQL-Verarbeitung in dem erstellten Thread (CREATE THREAD) verwendet wird.

Parameter	Standardwert	Format	Erläuterung
#XID	1	I4	Gibt an, dass eine globale Transaktionskennung verwendet wird. Wenn auf 0 (SIGNON) gesetzt, wird keine globale Transaktionskennung verwendet.

Beispiel für Planauswahl mit RRSAF unter TSO

Das folgende Beispiel zeigt die Planauswahl unter TSO unter Verwendung von RRSAF:

```
NATPLAN
<Enter>
Please enter new plan name  NDBPLAN4
            ,SUB SYSTEM ID DB27
            ,COLLECTION ID
         ,global XID (0/1) __________1
<Enter>
```

Beispiel für Planauswahl mit RRSAF im Batch-Modus

Das folgende Beispiel veranschaulicht die Planauswahl im Batch-Modus unter Verwendung von RRSAF:

```
NATPLAN NDBPLAN4,DB27, ,1
```

16 Natural-Statements und Systemvariablen benutzen

Dieses Kapitel behandelt besondere Aspekte, die zu berücksichtigen sind, wenn native Natural-DML-Statements und Natural-Systemvariablen zusammen mit Natural-SQL-Statements und Db2 verwendet werden.

Es handelt sich um eine Übersicht über Informationen, die in der Natural-Statements- bzw. -Systemvariablen-Dokumentation ausführlich vorhanden sind.

Eine Erklärung der Symbole, die in diesem Abschnitt zur Darstellung der Syntax von Natural-Statements verwendet werden, finden Sie unter *Syntax-Symbole* in der Natural-*Statements*-Dokumentation.

Informationen zur Protokollierung der in einem Natural-Programm enthaltenen SQL-Statements finden Sie unter

DBLOG Trace-Bildschirm für SQL-Statements in der *Debugger und Dienstprogramme (Utilities)* -Dokumentation.

Besondere Aspekte der speziellen Db2-Register

Natural for Db2 aktualisiert die folgenden speziellen Db2-Registerr automatisch auf die Werte, die für die zuletzt ausgeführte Transaktion galten.

- `CURRENT SQLID`
- `CURRENT SCHEMA`
- `CURRENT PATH`
- `CURRENT PACKAGE PATH`

Die Inhalte der folgenden Db2-Spezialregister aktualisiert Natural for Db2 nur dann automatisch auf die Werte, die für die zuletzt ausgeführte Transaktion galten, wenn der Parameter `REFRESH=ON` gesetzt ist.

- `CURRENT PACKAGESET`
- `CURRENT SERVER`

Diese speziellen Register werden unabhängig davon aufgefrischt, ob die zuvor ausgeführte Transaktion rückgängig gemacht (Rollback) oder festgeschrieben (Commit) wurde.

Alle anderen speziellen Register werden von Natural for Db2 nicht implizit beeinflusst.

Verwendung von nativen Natural DML-Statements

Dieser Abschnitt fasst bestimmte Punkte zusammen, die Sie beachten müssen, wenn Sie Natural-DML-Statements mit Db2 benutzen. Alle Natural-Statements, die in diesem Abschnitt nicht erwähnt werden, können ohne Einschränkung mit Db2 verwendet werden.

Eine Übersicht über die Natural-DML- und SQL-Statements finden Sie im Abschnitt *Datenbankzugriffe und Datenbankänderungen* in der Natural-Statements-Dokumentation.

Im Folgenden finden Sie Informationen zu den folgenden Natural-DML-Statements:

- BACKOUT TRANSACTION
- DELETE
- END TRANSACTION
- FIND
- HISTOGRAM
- READ
- STORE
- UPDATE

BACKOUT TRANSACTION

Das native Natural DML-Statement `BACKOUT TRANSACTION` macht alle Datenbanktransaktionen rückgängig, die seit dem Beginn der letzten logischen Transaktion vorgenommen wurden. Logische Transaktionen können entweder nach dem Beginn einer Sitzung oder nach der letzten `SYNCPOINT`-, `END TRANSACTION`- oder `BACKOUT TRANSACTION`-Statement beginnen.

Wie das Statement umgesetzt wird und welches Kommando tatsächlich abgesetzt wird, hängt von der TP-Monitor-Umgebung ab:

- Wenn dieses Kommando innerhalb einer Natural Stored Procedure oder Natural User-Defined Function (UDF) ausgeführt wird, führt Natural for Db2 die zugrunde liegende Rollback-Operation aus. Dadurch wird der Aufrufer in einen Must-Rollback-Zustand versetzt. Wenn dieses Kommando innerhalb einer Natural-Stored-Procedure oder einer UDF für Natural-Fehlerverarbeitung (implizites `ROLLBACK`) ausgeführt wird, führt Natural for Db2 die zugrunde liegende Rollback-Operation nicht aus, so dass der Aufrufer den ursprünglichen Natural-Fehler erhalten kann.
- Unter CICS wird das Statement `BACKOUT TRANSACTION` in ein `EXEC CICS ROLLBACK`-Kommando übersetzt. Im Pseudo-Conversational Mode werden jedoch nur die Änderungen rückgängig gemacht, die seit der letzten Terminal-E/A an der Datenbank vorgenommen wurden. Dies liegt an der CICS-spezifischen Transaktionsverarbeitung, siehe ***Natural for DB2 unter CICS***.

  Anmerkung: Beachten Sie, dass Natural bei Terminaleingaben in Datenbankschleifen in den Conversational Mode wechselt, wenn kein File Server verwendet wird.

- Im Batch-Modus und unter TSO wird das BACKOUT TRANSACTION-Statement in ein SQL ROLLBACK-Kommando übersetzt.

  Anmerkung: In einer DSNMTV01-Umgebung wird das BACKOUT TRANSACTION-Statement ignoriert, wenn der verwendete PSB ohne die Option CMPAT=YES generiert wurde.

- Unter IMS TM wird das BACKOUT TRANSACTION-Statement in ein IMS-Rollback-Kommando (ROLB) übersetzt. Es werden jedoch nur die Änderungen an der Datenbank rückgängig gemacht, die seit der letzten Terminal-E/A vorgenommen wurden. Dies liegt an der IMS TM-spezifischen Transaktionsverarbeitung, siehe *Natural for DB2 unter IMS TM*.

Da alle Cursors geschlossen werden, wenn eine logische Arbeitseinheit endet, darf ein BACKOUT TRANSACTION-Statement nicht innerhalb einer Datenbankschleife stehen, sondern muss außerhalb einer solchen Schleife oder nach der äußersten Schleife von geschachtelten Schleifen stehen.

Wenn ein externes Programm, das in einer anderen Standardprogrammiersprache geschrieben wurde, von einem Natural-Programm aufgerufen wird, darf dieses externe Programm kein eigenes ROLLBACK-Kommando enthalten, wenn das Natural-Programm ebenfalls Datenbankaufrufe tätigt. Das aufrufende Natural-Programm muss das BACKOUT TRANSACTION-Statement für das externe Programm absetzen.

Versucht ein Programm, Änderungen zurückzunehmen (Backout), die bereits festgeschrieben (Commit) wurden, z.B. durch eine Terminal-E/A, so wird eine entsprechende Natural-Fehlermeldung (NAT3711) zurückgegeben.

DELETE

Das native Natural DML-Statement DELETE wird verwendet, um eine Zeile aus einer Tabelle zu löschen, die mit einem vorangegangenen FIND-, READ- oder SELECT-Statement gelesen wurde. Es entspricht dem SQL-Statement DELETE WHERE CURRENT OF *cursor-name*, was bedeutet, dass nur die zuletzt gelesene Zeile gelöscht werden kann.

Beispiel:

```
FIND EMPLOYEES WITH NAME = 'SMITH'
     AND FIRST_NAME = 'ROGER'
DELETE
```

Natural übersetzt die obigen Natural-Statements in SQL und vergibt einen Cursornamen (z.B. CURSOR1) wie folgt:

```
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
```

Sowohl das SELECT- als auch das DELETE-Statement beziehen sich auf denselben Cursor.

Natural übersetzt ein natives Natural-DML-DELETE-Statement in ein Natural-SQL-DELETE-Statement auf die gleiche Weise, wie es ein natives Natural-DML-FIND-Statement in ein Natural-SQL-SELECT-Statement übersetzt.

Eine mit einem FIND SORTED BY-Statement gelesene Zeile kann aufgrund der beim FIND-Statement beschriebenen Db2-Einschränkungen nicht gelöscht werden. Eine mit READ LOGICAL gelesene Zeile kann ebenfalls nicht gelöscht werden.

DELETE bei Verwendung des File Servers

Wenn eine auf den File Server ausgelagerte Zeile gelöscht werden soll, liest Natural automatisch die ursprüngliche Zeile aus der Datenbank ein und vergleicht sie mit ihrem auf dem File Server gespeicherten Abbild. Wenn die ursprüngliche Zeile in der Zwischenzeit nicht geändert wurde, wird die DELETE-Operation durchgeführt. Bei der nächsten Terminal-E/A wird die Transaktion beendet, und die Zeile wird aus der aktuellen Datenbank gelöscht.

Wenn die DELETE-Operation auf einen scrollbaren Cursor wirkt, wird die Zeile auf dem File Server als „freier Datenbereich durch Löschen“ (DELETE-Hole) markiert und aus der Basistabelle gelöscht.

Wird jedoch eine Änderung festgestellt, so wird die Zeile nicht gelöscht und Natural gibt die Fehlermeldung NAT3703 für nicht scrollbare Cursor aus.

Wenn das DELETE auf einen scrollbaren Cursor wirkt, simuliert Natural for Db2 zwecks Konformität mit Db2 den SQLCODE -224 THE RESULT TABLE DOES NOT AGREE WITH THE BASE TABLE USING.

Wenn das DELETE auf einen scrollbaren Cursor wirkt und die Zeile zu einem „freien Datenbereich durch Löschen“ (DELETE-Hole) geworden ist, simuliert Natural for Db2 den SQLCODE -222 AN UPDATE OR DELETE OPERATION WAS ATTEMPTED AGAINST A HOLE.

Da ein DELETE-Statement erfordert, dass Natural eine einzelne Zeile erneut liest, muss ein eindeutiger Index für die entsprechende Tabelle vorhanden sein. Alle Spalten, die den eindeutigen Index bilden, müssen Teil der entsprechenden Natural-View sein.

END TRANSACTION

Das native DML-Statement `END TRANSACTION` zeigt das Ende einer logischen Transaktion an und gibt alle während der Transaktion gesperrten Db2-Daten frei. Alle Datenänderungen werden übergeben (Commit) und dauerhaft gemacht.

Wie das Statement übersetzt wird und welches Kommando tatsächlich ausgegeben wird, hängt von der TP-Monitor-Umgebung ab:

- Wenn dieses Kommando aus einer Natural Stored Procedure oder einer User Defined Function (UDF) ausgeführt wird, führt Natural for Db2 die zugrunde liegende Commit-Operation nicht aus. Dadurch kann die Stored Procedure oder UDF Änderungen (Updates) gegen Nicht-Db2-Datenbanken vornehmen (Commit).
- Unter CICS wird das `END TRANSACTION`-Statement in ein `EXEC CICS SYNCPOINT`-Kommando übersetzt. Wenn der File Server verwendet wird, wird nach jeder Terminal-E/A ein implizites Transaktionsende (End of Transaction) abgesetzt. Dies liegt an der CICS-spezifischen Transaktionsverarbeitung im Pseudo-Conversational Mode, siehe ***Natural for DB2 unter CICS***.
- Im Batch-Modus und unter TSO wird das `END TRANSACTION`-Statement in ein SQL-Kommando `COMMIT WORK` übersetzt.

  Anmerkung: In einer `DSNMTV01`-Umgebung wird das `END TRANSACTION`-Statement ignoriert, wenn der verwendete PSB ohne die Option `CMPAT=YES` erzeugt wurde.

- Unter IMS TM wird das `END TRANSACTION`-Statement nicht in einen IMS CHKP-Aufruf übersetzt, sondern ignoriert. Aufgrund der IMS TM-spezifischen Transaktionsverarbeitung (siehe ***Natural for DB2 unter IMS TM***) wird nach jeder Terminal-E/A ein implizites Transaktionsende (End of Transaction) ausgegeben.

Außer in Kombination mit der `WITH HOLD`-Klausel (siehe `SELECT - SQL` unter ***Verwendung von Natural SQL-Statements***) darf ein `END TRANSACTION`-Statement nicht innerhalb einer Datenbankschleife stehen, da alle Cursor geschlossen werden, wenn eine logische Arbeitseinheit endet. Stattdessen muss sie außerhalb einer solchen Schleife bzw. bei geschachtelten Schleifen nach der äußersten Schleife stehen.

Wird ein externes Programm, das in einer anderen Standardprogrammiersprache geschrieben wurde, von einem Natural-Programm aufgerufen, darf dieses externe Programm kein eigenes `COMMIT`-Kommando enthalten, wenn das Natural-Programm ebenfalls Datenbankaufrufe tätigt. Das aufrufende Natural-Programm muss das `END TRANSACTION`-Statement für das externe Programm absetzen.

Anmerkung: Bei Db2 kann das `END TRANSACTION`-Statement nicht verwendet werden, um Transaktionsdaten zu speichern.

FIND

Das native Natural-DML-Statement `FIND` entspricht dem Natural SQL-Statement `SELECT`.

Beispiel:

Native Natural-DML-Statements:

```
FIND EMPLOYEES WITH NAME = 'BLACKMORE'
   AND AGE EQ 20 THRU 40
OBTAIN PERSONNEL_ID NAME AGE
```

Äquivalentes Natural-SQL-Statement:

```
SELECT PERSONNEL_ID, NAME, AGE
  FROM EMPLOYEES
    WHERE NAME = 'BLACKMORE'
      AND AGE BETWEEN 20 AND 40
```

Natural übersetzt intern ein `FIND`-Statement in ein SQL-`SELECT`-Statement. Siehe Beschreibung in *Verarbeitung von SQL-Statements, die von Natural ausgegeben werden* im Abschnitt *Interne Behandlung dynamischer Statements*. Das `SELECT`-Statement wird von einem `OPEN CURSOR`-Statement, gefolgt von einem `FETCH`-Kommando, ausgeführt. Das `FETCH`-Kommando wird so oft ausgeführt, bis entweder alle Sätze gelesen wurden oder der Programmablauf die `FIND`-Verarbeitungsschleife verlässt. Ein `CLOSE CURSOR`-Kommando beendet die `SELECT`-Verarbeitung.

Die `WITH`-Klausel eines `FIND`-Statements wird in die `WHERE`-Klausel des `SELECT`-Statements umgesetzt. Das Basissuchkriterium für eine Db2-Tabelle kann auf die gleiche Weise wie für eine Adabas-Datei angegeben werden. Das bedeutet, dass nur Datenbankfelder, die als Deskriptoren definiert sind, zum Aufbau von Basissuchkriterien verwendet werden können und dass Deskriptoren nicht mit anderen Feldern der Natural-View (d.h. Datenbankfeldern) verglichen werden können, sondern nur mit Programmvariablen oder Konstanten.

Anmerkung: Da jedes Datenbankfeld (Spalte/Column) einer Db2-Tabelle für die Suche verwendet werden kann, kann ein beliebiges Datenbankfeld als Deskriptor in einem Natural-DDM definiert werden.

Die `WHERE`-Klausel des `FIND`-Statements wird von Natural ausgewertet, *nachdem* die Zeilen über die `WITH`-Klausel ausgewählt worden sind. Innerhalb der `WHERE`-Klausel können Nicht-Deskriptoren verwendet werden und Datenbankfelder können mit anderen Datenbankfeldern verglichen werden.

Anmerkung: Db2 kennt keine Sub-, Super- oder phonetischen Deskriptoren.

Ein `FIND NUMBER`-Statement wird in ein `SELECT`-Statement mit einer `COUNT(*)`-Klausel übersetzt. Die Anzahl der gefundenen Zeilen wird in der Natural-Systemvariablen `*NUMBER` zurückgegeben. Beschreibung siehe Natural-*Systemvariablen*-Dokumentation.

Das FIND UNIQUE-Statement kann verwendet werden, um sicherzustellen, dass nur ein Datensatz für die Verarbeitung ausgewählt wird. Wenn das FIND UNIQUE-Statement von einem UPDATE-Statement referenziert wird, wird eine UPDATE-Operation ohne Cursor (Searched) anstelle einer cursororientierten (Positioned) UPDATE-Operation generiert. Daher können Sie es verwenden, wenn Sie einen Db2-Primärschlüssel aktualisieren möchten. Es wird jedoch empfohlen, zur Aktualisierung eines Primärschlüssels das Natural SQL Searched UPDATE-Statement zu verwenden.

Im statischen Modus werden die Statements FIND NUMBER und FIND UNIQUE in ein SELECT SINGLE-Statement übersetzt. Siehe Abschnitt *Verwendung von Natural SQL-Statements*.

Das FIND FIRST-Statement kann nicht verwendet werden. Die Klauseln PASSWORD, CIPHER, COUPLED und RETAIN können ebenfalls nicht verwendet werden.

Die SORTED BY-Klausel eines FIND-Statements wird in die SQL-Klausel SELECT ... ORDER BY übersetzt, die auf das Suchkriterium folgt. Da dies eine schreibgeschützte Ergebnistabelle ergibt, kann eine mit einem FIND-Statement gelesene Zeile, die eine SORTED BY-Klausel enthält, nicht aktualisiert oder gelöscht werden.

Bei der Installation kann eine Grenze für die Tiefe der verschachtelten Datenbankschleifen festgelegt werden. Wird diese Grenze überschritten, so wird eine Natural-Fehlermeldung ausgegeben.

Anmerkungen:

1. Wenn eine Verarbeitungsgrenze als konstante Ganzzahl angegeben wird, z. B. FIND (5), wird der Grenzwert in eine FETCH FIRST integer ROWS ONLY-Klausel im generierten SQL-String übersetzt.
2. Natural for Db2 unterstützt die Verarbeitung mehrerer Zeilen in Db2 im Rahmen der MULTIFETCH-Klausel des FIND-Statements.

FIND bei Verwendung des File Servers

Hinsichtlich der Nutzung des File Servers gibt es keine programmtechnischen Einschränkungen bei den Auswahl-Statements. Es ist jedoch empfehlenswert, sich mit der Funktionalität vertraut zu machen, um die Leistungsfähigkeit und den Platzbedarf des File Servers zu berücksichtigen.

HISTOGRAM

Das Natural DML-Statement HISTOGRAM gibt die Anzahl der Zeilen in einer Tabelle zurück, die in einer bestimmten Spalte den gleichen Wert haben. Die Anzahl der Zeilen wird in der Natural-Systemvariablen *NUMBER zurückgegeben. Siehe Natural-*Systemvariablen*-Dokumentation.

Beispiel:

Native Natural-DML-Statements:

```
HISTOGRAM EMPLOYEES FOR AGE
OBTAIN AGE
```

Äquivalentes Natural-SQL-Statement:

```
SELECT COUNT(*), AGE FROM EMPLOYEES
  WHERE AGE > -999
  GROUP BY AGE
  ORDER BY AGE
```

Natural übersetzt das HISTOGRAM-Statement in ein SQL-SELECT-Statement, d.h. der Kontrollfluss ist ähnlich wie beim FIND-Statement beschrieben.

Anmerkung: Mit dem Universal Database Server for z/OS Version 8 unterstützt Natural for Db2 die Verarbeitung mehrerer Zeilen in Db2 im Rahmen der MULTIFETCH-Klausel des HISTOGRAM-Statements.

READ

Das native Natural-DML-Statement READ kann auch für den Zugriff auf Db2-Tabellen verwendet werden. Natural übersetzt ein READ-Statement in ein SQL SELECT-Statement.

READ PHYSICAL und READ LOGICAL können verwendet werden. READ BY ISN kann jedoch nicht verwendet werden, da es keine Db2-Entsprechung zu Adabas-ISNs gibt. Die Klauseln PASSWORD und CIPHER können auch nicht verwendet werden.

Da ein READ LOGICAL-Statement in ein SELECT ... ORDER BY-Statement übersetzt wird, das eine schreibgeschützte Tabelle erzeugt, kann eine mit einem READ LOGICAL-Statement gelesene Zeile nicht geändert oder gelöscht werden (siehe Beispiel 1). Der Startwert kann nur eine Konstante oder eine Programmvariable sein; ein anderes Feld der Natural-Ansicht (d.h. ein Datenbankfeld) kann nicht verwendet werden.

Ein READ PHYSICAL-Statement wird in ein SELECT-Statement ohne ORDER BY-Klausel übersetzt und kann daher aktualisiert oder gelöscht werden (siehe Beispiel 2).

Beispiel 1:

Natives Natural-DML-Statement:

```
READ PERSONNEL BY NAME
OBTAIN NAME FIRSTNAME DATEOFBIRTH
```

Äquivalentes Natural-SQL-Statement:

```
SELECT NAME, FIRSTNAME, DATEOFBIRTH FROM PERSONNEL
  WHERE NAME >= ' '
    ORDER BY NAME
```

Beispiel 2:

Native Natural-DML-Statements:

```
READ PERSONNEL PHYSICAL
OBTAIN NAME
```

Äquivalentes Natural-SQL-Statement:

```
SELECT NAME FROM PERSONNEL
```

Wenn das READ-Statement eine WHERE-Klausel enthält, wird diese Klausel vom Natural-Prozessor ausgewertet, *nachdem* die Zeilen gemäß dem (den) im Suchkriterium angegebenen Deskriptorwert(en) ausgewählt wurden.

Verarbeitungsgrenze

Wenn eine Verarbeitungsgrenze als konstante Ganzzahl, z.B. READ (5), im generierten SQL-String angegeben ist, wird der Wert, der die Grenze definiert, in die Klausel übersetzt:

```
FETCH FIRST integer ROWS ONLY
```

Cursors für Db2-Klauseln

Natural for Db2 verwendet insensitive scrollbare Cursors, um das folgende READREAD-Statement zu verarbeiten:

```
READ .. [IN] [LOGICAL] VARIABLE/DYNAMIC operand5 [SEQUENCE]
```

Natural for Db2 verwendet insensitive scrollbare Cursors, um das folgende READ-Statement zu verarbeiten. Wenn es sich um ein Positioned UPDATE- oder Positioned DELETE-Statement handelt, verwendet Natural for Db2 insensitive statische Cursors.

```
READ .. [IN] [PHYSICAL] DESCENDING/VARIABLE/DYNAMIC operand5 [SEQUENCE]
```

operand5

Der Wert A wird in einen FETCH FIRST/NEXT-Db2-Zugriff und der Wert D in einen FETCH LAST/PRIOR-Db2-Zugriff übersetzt.

Anmerkung: Natural for Db2 unterstützt die Db2-Verarbeitung mehrerer Zeilen im Rahmen der MULTIFETCH-Klausel im READ-Statement.

READ bei Verwendung des File Servers

Hinsichtlich der Verwendung des File Servers gibt es keine programmtechnischen Einschränkungen bei den Auswahl-Statements. Es ist jedoch empfehlenswert, sich mit der Funktionalität unter Berücksichtigung von Performance und Platzbedarf des File Servers vertraut zu machen.

STORE

Das Native Natural-DML-Statement `STORE` wird verwendet, um eine Zeile zu einer Db2-Tabelle hinzuzufügen. Das STORE-Statement entspricht dem SQL- Statement `INSERT`.

Beispiel:

Natives Natural-DML-Statement:

```
STORE RECORD IN EMPLOYEES
  WITH PERSONNEL_ID = '2112'
       NAME         = 'LIFESON'
       FIRST_NAME   = 'ALEX'
```

Äquivalentes Natural SQL-Statement:

```
INSERT INTO EMPLOYEES (PERSONNEL_ID, NAME, FIRST_NAME)
  VALUES ('2112', 'LIFESON', 'ALEX')
```

Die Klauseln `PASSWORD`, `CIPHER` und `USING/GIVING NUMBER` des `STORE`-Statements können nicht verwendet werden.

UPDATE

Das native Natural-DML-Statement `UPDATE` aktualisiert eine Zeile in einer Db2-Tabelle, die mit einem vorangegangenen `FIND`, `READ` oder `SELECT`-Statement gelesen wurde. Sie entspricht dem SQL-Statement `UPDATE WHERE CURRENT OF` *`cursor-name`* (Positioned `UPDATE`), was bedeutet, dass nur die zuletzt gelesene Zeile aktualisiert werden kann.

UPDATE bei Verwendung des File Servers

Wenn eine auf den File Server ausgelagerte Zeile aktualisiert werden soll, liest Natural automatisch die ursprüngliche Zeile aus der Datenbank ein und vergleicht sie mit ihrem auf dem File Server gespeicherten Abbild. Wenn die ursprüngliche Zeile in der Zwischenzeit nicht geändert wurde, wird die `UPDATE`-Operation durchgeführt. Bei der nächsten Terminal-E/A wird die Transaktion beendet und die Zeile endgültig in der Datenbank aktualisiert.

Wenn die `UPDATE`-Operation auf einen scrollbaren Cursor wirkt, werden die Zeile auf dem File Server und die Zeile in der Basistabelle aktualisiert. Erfüllt die Zeile nach der Aktualisierung nicht mehr die Suchkriterien des zugehörigen `SELECT`-Statements, wird die Zeile auf dem File Server als „freier Datenbereich durch Aktualisieren“ (`UPDATE`-Hole) markiert.

Wird jedoch eine Änderung festgestellt, wird die Zeile nicht aktualisiert und Natural gibt die Fehlermeldung NAT3703 für nicht scrollbare Cursor aus.

Wenn das UPDATE auf einen scrollbaren Cursor wirkt, simuliert Natural for Db2 zwecks Konformität mit Db2 den SQLCODE -224 THE RESULT TABLE DOES NOT AGREE WITH THE BASE TABLE USING.

Wenn das UPDATE auf einen scrollbaren Cursor wirkt und die Zeile ein „freier Datenbereich durch Aktualisieren“ (UPDATE-Hole) geworden ist, simuliert Natural for Db2 den SQLCODE -222 AN UPDATE OR DELETE OPERATION WAS ATTEMPTED AGAINST A HOLE.

Da ein UPDATE-Statement das erneute Lesen einer einzelnen Zeile durch Natural erfordert, muss ein eindeutiger Index für diese Tabelle vorhanden sein. Alle Spalten, die den eindeutigen Index umfassen, müssen Teil der entsprechenden Natural-View sein.

UPDATE mit FIND/READ

Wie beim nativen Natural-DML-Statement FIND beschrieben, übersetzt Natural ein FIND-Statement in ein SQL SELECT-Statement. Wenn ein Natural-Programm ein DML UPDATE-Statement enthält, wird dieses Statement in ein SQL UPDATE-Statement übersetzt und dem SELECT-Statement eine FOR UPDATE OF-Klausel hinzugefügt.

Beispiel:

```
FIND EMPLOYEES WITH SALARY < 5000
  ASSIGN SALARY = 6000
  UPDATE
```

Natural übersetzt die obigen Natural-Anweisungen in SQL und vergibt einen Cursornamen (z. B. CURSOR1) wie folgt:

```
DECLARE CURSOR1 CURSOR FOR
SELECT SALARY FROM EMPLOYEES WHERE SALARY < 5000
  FOR UPDATE OF SALARY
UPDATE EMPLOYEES SET SALARY = 6000
  WHERE CURRENT OF CURSOR1
```

Sowohl das SELECT- als auch das UPDATE-Statement beziehen sich auf denselben Cursor.

Aufgrund der Db2-Logik kann eine Spalte (ein Feld) nur dann aktualisiert werden, wenn sie in der FOR UPDATE OF-Klausel enthalten ist. Andernfalls wird das Aktualisieren dieser Spalte (dieses Feldes) abgelehnt. Natural nimmt automatisch alle Spalten (Felder) in die FOR UPDATE OF-Klausel auf, die an beliebiger Stelle im Natural-Programm geändert wurden oder die als Teil einer Natural-Maske (Map) Eingabefelder sind.

Eine Db2-Spalte wird jedoch nicht aktualisiert, wenn die Spalte (das Feld) im Natural-DDM als „nicht aktualisierbar“ gekennzeichnet ist. Solche Spalten (Felder) werden ohne Warnung oder

Fehlermeldung aus der FOR UPDATE OF-Liste entfernt. Die in der FOR UPDATE OF-Liste enthaltenen Spalten (Felder) können mit dem Kommando LISTSQL überprüft werden.

Der Adabas-Kurzname im Natural DDM bestimmt, ob eine Spalte (ein Feld) aktualisiert werden kann.

Die folgende Tabelle zeigt, welche Bereiche gelten:

Kurzbezeichnungsbereich	Feldtyp
AA - N9	Nicht-Schlüsselfeld, das aktualisiert werden kann.
Aa - Nz	Nicht-Schlüsselfeld, das aktualisiert werden kann.
OA - O9	Primärschlüsselfeld.
PA - P9	Aufsteigendes Schlüsselfeld, das aktualisiert werden kann.
QA - Q9	Absteigendes Schlüsselfeld, das aktualisiert werden kann.
RA - X9	Nicht-Schlüsselfeld, das nicht aktualisiert werden kann.
Ra - Xz	Nicht-Schlüsselfeld, das nicht aktualisiert werden kann.
YA - Y9	Aufsteigendes Schlüsselfeld, das nicht aktualisiert werden kann.
ZA - Z9	Absteigendes Schlüsselfeld, das nicht aktualisiert werden kann.
1A - 9Z	Nicht-Schlüsselfeld, das nicht aktualisiert werden kann.
1a - 9z	Nicht-Schlüsselfeld, das nicht aktualisiert werden kann.

Beachten Sie, dass ein Primärschlüsselfeld niemals Teil einer FOR UPDATE OF-Liste ist. Ein Primarschlüsselfeld kann nur mit einer UPDATE-Operation ohne Cursor aktualisiert werden (siehe auch das Natural-SQL-UPDATE-Statement im Abschnitt *Verwendung von Natural SQL Statements*.

Beachten Sie, dass ein Primärschlüsselfeld niemals Teil einer FOR UPDATE OF-Liste ist, außer wenn die Compiler-Option DB2PKYU auf ON gesetzt ist. Wenn DB2PKYU auf OFF gesetzt ist, was der Standardwert ist, können Sie ein Primärschlüsselfeld nur mit einer UPDATE-Operation ohne Cursor aktualisieren. Weitere Informationen finden Sie auch unter Natural SQL UPDATE-Statement im Abschnitt *Verwendung von Natural SQL-Statements*.

Eine mit einem FIND-Statement gelesene Zeile, die eine SORTED BY-Klausel enthält, kann nicht aktualisiert werden (aufgrund von Db2-Einschränkungen, wie bei dem FIND-Statement beschrieben). Eine mit einem READ LOGICAL-Statement gelesene Zeile kann ebenfalls nicht aktualisiert werden (wie bei dem READ-Statement beschrieben).

Wenn eine Spalte aktualisiert werden soll, die als Array redefiniert ist, wird dringend empfohlen, die gesamte Spalte zu aktualisieren und nicht einzelne Ausprägungen. Andernfalls sind die Ergebnisse nicht vorhersehbar. Dazu können Sie im Reporting Mode das Statement OBTAIN verwenden, das auf alle Feldausprägungen in der zu aktualisierenden Spalte angewendet werden muss. Im Structured Mode hingegen müssen alle diese Ausprägungen in der entsprechenden Natural-View definiert sein.

Die durch ein UPDATE-Statement gesperrten Daten werden freigegeben, wenn ein Statement END TRANSACTION (COMMIT WORK) oder BACKOUT TRANSACTION (ROLLBACK WORK) vom Programm ausgeführt wird.

Anmerkung: Wird in einem Natural-Programm ein Längen-Indikatorfeld oder ein NULL-Indikatorfeld aktualisiert, ohne dass das Feld (die Spalte), auf das es sich bezieht, aktualisiert wird, so wird die Aktualisierung der Spalte für Db2 nicht generiert und es erfolgt somit keine Aktualisierung.

UPDATE bei SELECT

Generell kann das native Natural-DML-Statement UPDATE sowohl im Structured als auch im Reporting Mode verwendet werden. Nach einem SELECT-Statement ist jedoch nur die für den Natural Structured Mode definierte Syntax zulässig:

```
UPDATE [RECORD] [IN] [STATEMENT] [(r)]
```

Dies liegt daran, dass das native Natural-DML-Statement UPDATE in Kombination mit dem SELECT-Statement nur im folgenden speziellen Fall erlaubt ist:

```
...
SELECT ...
   INTO VIEW view-name
   ...
```

Es kann also nur eine ganze Natural-View aktualisiert werden, einzelne Spalten (Felder) nicht.

Beispiel:

```
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 Kombination mit dem nativen Natural-DML-Statement `UPDATE` wird jede andere Form des `SELECT`-Statements zurückgewiesen und eine Fehlermeldung zurückgegeben.

Im übrigen kann das native Natural-DML-Statement `UPDATE` mit dem `SELECT`-Statement genauso wie mit dem Natural-Statement `FIND` verwendet werden.

Verwendung von Natural-SQL-Statements

Dieser Abschnitt behandelt Punkte, die bei der Verwendung von Natural SQL-Statements mit Db2 zu beachten sind. Diese Db2-spezifischen Punkte bestehen zum Teil aus Syntaxerweiterungen, die zum Extended Set der Natural SQL-Syntax gehören. Das Extended Set wird zusätzlich zum Common Set angeboten, um datenbankspezifische Besonderheiten zu unterstützen. Siehe *Common Set und Extended Set* in der *Statements*-Dokumentation.

Informationen zur Protokollierung der in einem Natural-Programm enthaltenen SQL-Statements finden Sie unter *DBLOG Trace-Bildschirm für SQL-Statements* unter *DBLOG Utility* in der *Debugger und Dienstprogramme (Utilities)*-Dokumentation.

Nachfolgend finden Sie Informationen zu den folgenden Natural SQL-Statements und zu gemeinsamen syntaktischen Elementen:

- Gemeinsame syntaktische Elemente für 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

Gemeinsame syntaktische Elemente für Natural SQL-Statements

Die folgenden gemeinsamen syntaktischen Elemente sind entweder Db2-spezifisch und entsprechen nicht den Standard-SQL-Syntaxdefinitionen (d.h. dem Common Set der Natural SQL Syntax) oder unterliegen Einschränkungen bei der Verwendung mit Db2 (siehe auch *Natural-SQL-Statements benutzen* in der Natural-*Statements*-Dokumentation).

Im Folgenden finden Sie Informationen zu den gemeinsamen syntaktischen Elementen:

- atom
- comparison

- factor
- scalar-function
- column-function (aggregate-function)
- scalar-operator
- special-register
- units
- case-expression

atom

Ein *atom* kann entweder ein Parameter (d.h. eine Natural-Programm- oder Host-Variable) oder eine Konstante sein. Bei der dynamischen Ausführung ist die Verwendung von Host-Variablen jedoch durch Db2 eingeschränkt. Weitere Einzelheiten sind in der entsprechenden Db2-Literatur von IBM zu finden.

comparison

Die für Db2 spezifischen Vergleichsoperatoren gehören zum Natural SQL Extended Set. Eine Beschreibung finden Sie unter *comparison-predicate* in *Suchbedingungen* in der Statements-Dokumentation.

factor

Die folgenden Faktoren sind Db2-spezifisch und gehören zum Natural SQL Extended Set:

```
special-register
scalar-function(scalar-expression, ...)
scalar-expression unit
case-expression
```

scalar-function

Eine **scalar-function** ist eine eingebaute Funktion, die bei der Formulierung von skalaren Berechnungsausdrücken verwendet werden kann. **Scalar-functions** sind Db2-spezifisch und gehören zum Natural SQL Extended Set.

Die **scalar-functions**, die Natural for Db2 unterstützt, sind unten in alphabetischer Reihenfolge aufgeführt:

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
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

A - H	I - R	S - Z
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	

Auf jede **scalar-function** folgen ein oder mehrere **scalar-expressions** in Klammern. Die Anzahl der **scalar-expressions** hängt von der **scalar-function** ab. Mehrere **scalar-expressions** müssen durch Kommata voneinander getrennt werden.

Beispiel:

```
SELECT NAME
  INTO NAME
  FROM SQL-PERSONNEL
  WHERE SUBSTR ( NAME, 1, 3 ) = 'Fri'
       ...
```

column-function (aggregate-function)

Eine **column-function** gibt ein einwertiges Ergebnis für das Argument zurück, das sie erhält. Das Argument ist eine Menge gleichartiger Werte, z. B. die Werte einer Spalte. **Column-functions** werden auch **aggregate-functions** genannt.

Die folgenden **column-functions** entsprechen dem SQL-Standard. Sie sind nicht Db2-spezifisch:

```
AVG
COUNT
MAX
MIN
SUM
```

Die folgenden **column-functions** sind nicht konform mit dem SQL-Standard. Sie sind Db2-spezifisch und gehören zum 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

Der Verkettungsoperator (`CONCAT` oder `||`) beim **scalar-operator** entspricht nicht dem SQL-Standard. Er ist Db2-spezifisch und gehört zum Natural Extended Set.

special-register

Mit Ausnahme von `USER` entsprechen die folgenden **special-registers** nicht dem SQL-Standard. Sie sind Db2-spezifisch und gehören zum 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
```

Eine Referenz auf ein spezielles Register gibt einen skalaren Wert zurück.

Mit dem Kommando `SET CURRENT SQLID` kann der Ersteller-Name (`creator`) einer Tabelle durch die aktuelle SQLID ersetzt werden. Damit ist es möglich, auf identische Tabellen mit demselben Tabellennamen, aber mit unterschiedlichen Ersteller-Namen zuzugreifen.

units

Units (Einheiten), auch **durations** genannt, sind Db2-spezifisch und gehören zum Natural SQL Extended Set.

Die folgenden **units** werden unterstützt:

```
DAY
DAYS
HOUR
HOURS
MICROSECOND
MICROSECONDS
MINUTE
MINUTES
MONTH
MONTHS
SECOND
SECONDS
YEAR
YEARS
```

case-expression

Case-expressions entsprechen nicht dem Standard-SQL und werden daher nur vom Natural SQL Extended Set unterstützt.

Beispiel:

```
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

Das Natural SQL-Statement `CALLDBPROC` wird für den Aufruf von Db2 Stored Procedures verwendet. Es unterstützt den Ergebnismengenmechanismus (`result-set`) von Db2 und ermöglicht den Aufruf von Db2 Stored Procedures. Weitere Einzelheiten und die Statement-Syntax finden Sie unter *CALLDBPROC (SQL)* in der *Statements*-Dokumentation.

Die folgenden Themen werden behandelt:

- Statische und dynamische Ausführung
- Result Sets (Ergebnismengen)
- Liste der Parameterdatentypen
- CALLMODE=NATURAL

- Beispiel für CALLDBPROC/READ RESULT SET

Statische und dynamische Ausführung

Wenn das CALLDBPROC-Statement dynamisch ausgeführt wird, werden alle Parameter und Konstanten auf die Variablen des folgenden Db2 SQL-Statement abgebildet:

```
CALL :hv USING  DESCRIPTOR :sqlda statement
```

:hv bezeichnet eine Host-Variable, die den Namen der aufzurufenden Prozedur enthält.

:sqlda ist eine dynamisch generierte sqlda, die die Parameter beschreibt, die an die Stored Procedure übergeben werden sollen.

Wenn das CALLDBPROC-Statement statisch ausgeführt wird, werden die Konstanten des CALLDBPROC-Statements auch als Konstanten in dem generierten Assembler SQL-Quellcode für den Db2 Precompiler erzeugt.

Result Sets (Ergebnismengen)

Wenn der von dem CALL-Statement erzeugte SQLCODE anzeigt, dass es Ergebnismengen gibt (SQLCODE +466 und +464), führt die Natural for Db2-Laufzeit ein DESCRIBE PROCEDURE *:hv* INTO : *sqlda*-Statement aus, um die result set locator-Werte der von der aufgerufenen Stored Procedure erzeugten Ergebnismengen abzurufen. Diese Werte werden in die RESULT SETS-Variablen gestellt, die in dem CALLDBPROC-Statement angegeben sind. Jede in einem CALLDBPROC angegebene RESULT SETS-Variable, für die kein result set locator-Wert vorhanden ist, wird auf Null zurückgesetzt. Die result set locator-Werte können zum Lesen der Ergebnismengen mit dem READ RESULT SET-Statement verwendet werden, solange die Datenbanktransaktion, die die Ergebnismenge erzeugt hat, noch kein COMMIT oder ROLLBACK abgesetzt hat.

Wurde die Ergebnismenge durch einen Cursor WITH HOLD erzeugt, bleibt der result set locator-Wert auch nach einer COMMIT-Operation gültig.

Im Gegensatz zu anderen Natural SQL-Statements können Sie bei CALLDBPROC (optional!) nach dem Schlüsselwort GIVING eine SQLCODE-Variable angeben, die den SQLCODE des zugrunde liegenden CALL-Statements enthält. Wenn GIVING angegeben wird, ist es Aufgabe des Natural-Programms, auf den SQLCODE zu reagieren (die Fehlermeldung NAT3700 wird von der Laufzeit nicht ausgegeben).

Liste der Parameterdatentypen

Im Folgenden werden die vom Statement CALLDBPROC unterstützten Parameterdatentypen aufgeführt:

Natural-Format/Länge	DB2-Datentyp
A*n*	CHAR(*n*)
B2	SMALLINT
B4	INT
B*n* (*n* = not equal 2 or 4)	CHAR(*n*)
F4	REAL
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 **Anmerkung:** Das Natural-Format T hat einen größeren Datenbereich als der entsprechende Db2 TIME-Datentyp. Im Vergleich zu Db2 TIME verfügt die Natural-Variable T außerdem über einen Datumsanteil (Jahr, Monat, Tag) und die Zehntelsekunde. Daher schneidet Natural for Db2 bei der Konvertierung einer Natural-Variablen T in einen Db2-TIME-Wert den Datumsanteil und die Zehntelsekunde ab. Bei der Konvertierung von Db2 TIME in das Natural-Format T wird der Datumsanteil auf 0000-01-02 und der Zehntelsekundenanteil in Natural auf 0 zurückgesetzt.

CALLMODE=NATURAL

Dieser Parameter wird verwendet, um Natural Stored Procedures aufzurufen, die mit PARAMETER STYLE GENERAL/WITH NULL definiert wurden.

Wenn der Parameter CALLMODE=NATURAL angegeben ist, wird ein zusätzlicher Parameter, der die an die Natural Stored Procedure übergebenen Parameter beschreibt, vom Client, d.h. dem Aufrufer, an den Server, d.h. den Natural for Db2 Server Stub, übergeben. Der Parameter ist der Stored Procedure Control Block (STCB; siehe auch *STCB Layout* in *PARAMETER STYLE* im Abschnitt *Verarbeitung von Natural Stored Procedures und UDFs*) und hat aus Sicht von Db2 das Format VARCHAR. Daher muss jede Natural Stored Procedure, die mit PARAMETER STYLE GENERAL/WITH

NULL definiert ist, mit dem CREATE PROCEDURE-Statement definiert werden, indem dieser VARCHAR-Parameter als erster in seiner PARMLIST-Zeile verwendet wird.

Aus der Sicht des Aufrufers, d.h. des Natural-Programms, und aus der Sicht der Stored Procedure, d.h. des Natural-Subprogramms, ist der STCB unsichtbar. Er wird als erster Parameter von der Natural for Db2-Laufzeit übergeben und auf der Server-Seite verwendet, um die Kopie der übergebenen Daten im Natural-Thread und dem entsprechenden CALLNAT-Statement zu erstellen. Außerdem dient dieser Parameter als Container für Fehlerinformationen, die während der Ausführung der Natural-Stored-Procedure durch die Natural-Laufzeit erzeugt werden. Er enthält auch Informationen über die Library, in der Sie angemeldet sind, und das aufzurufende Natural-Subprogramm.

Beispiel für CALLDBPROC/READ RESULT SET

Nachfolgend finden Sie ein Beispielprogramm für die Ausführung der Statements CALLDBPROC und READ RESULT SET:

```
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

Das Natural SQL-Statement `COMMIT` zeigt das Ende einer logischen Transaktion an und gibt alle während der Transaktion gesperrten Db2-Daten frei. Alle Datenänderungen werden permanent gemacht. Weitere Einzelheiten und die Statement-Syntax finden Sie unter *COMMIT (SQL)* in der *Statements*-Dokumentation.

`COMMIT` ist ein Synonym für das native Natural DML-Statement `END TRANSACTION`, das im Abschnitt ***Verwendung von nativen Natural DML-Statements*** beschrieben ist.

Mit dem `COMMIT`-Statement können keine Transaktionsdaten mitgegeben werden.

Wenn dieses Statement von einer Natural Stored Procedure oder einer benutzerdefinierten Funktion (UDF) ausgeführt wird, führt Natural for Db2 die zugrunde liegende Commit-Operation nicht aus. Dadurch kann die Natural Stored Procedure bzw. die benutzerdefinierte Funktion (UDF) ein Commit auf Updates bei Nicht-Db2-Datenbanken ausführen.

Unter CICS wird das `COMMIT`-Statement in ein `EXEC CICS SYNCPOINT`-Kommando übersetzt. Wird der File Server verwendet, wird nach jeder Terminal-E/A ein implizites Transaktionsende ausgegeben. Dies liegt an der CICS-spezifischen Transaktionsverarbeitung im Pseudo-Conversational Mode, siehe ***Natural for DB2 unter CICS***.

Unter IMS TM wird das `COMMIT`-Statement nicht in ein IMS `CHECKPOINT`-Kommando übersetzt, sondern ignoriert. Ein implizites Transaktionsende wird nach jeder Terminal-E/A ausgegeben. Dies liegt an der IMS TM-spezifischen Transaktionsverarbeitung, siehe ***Natural for DB2 unter IMS TM***.

Ein COMMIT-Statement darf nicht innerhalb einer Datenbankschleife stehen, es sei denn, es wird in Kombination mit der WITH HOLD-Klausel verwendet (siehe *Syntax 1 - Cursor-orientierte Selektion* in *SELECT (SQL)* in der *Statements*-Dokumentation), da alle Cursor am Ende einer logischen Arbeitseinheit geschlossen werden. Stattdessen muss sie außerhalb einer solchen Schleife bzw. bei geschachtelten Schleifen nach der äußersten Schleife stehen.

Wenn ein externes Programm, das in einer anderen Standardprogrammiersprache geschrieben wurde, von einem Natural-Programm aufgerufen wird, darf dieses externe Programm kein eigenes COMMIT-Kommando enthalten, wenn das Natural-Programm ebenfalls Datenbankaufrufe tätigt. Das aufrufende Natural-Programm muss das COMMIT-Statement für das externe Programm absetzen.

DELETE - SQL

Im Rahmen von Natural SQL werden sowohl das cursororientierte bzw. Positioned DELETE als auch das nicht-cursororientierte bzw. Searched DELETE unterstützt. Die Funktionalität des Positioned DELETE entspricht derjenigen des Natural DML DELETE. Weitere Einzelheiten und die Statement-Syntax finden Sie unter *DELETE (SQL)* in der Natural-*Statements*-Dokumentation.

Bei Db2 kann ein Tabellenname in der *FROM-Klausel* eines Searched DELETE-Statements mit einem *correlation-name* versehen werden. Dies entspricht nicht der Standard-SQL-Syntaxdefinition und gehört daher zum Natural SQL Extended Set.

Das Searched DELETE-Statement muss z.B. verwendet werden, um eine Zeile aus einer selbstreferenzierenden Tabelle zu löschen, da bei selbstreferenzierenden Tabellen ein Positioned DELETE von Db2 nicht zugelassen wird.

INSERT - SQL

Das Natural SQL INSERT-Statement wird verwendet, um eine oder mehrere neue Zeilen zu einer Tabelle hinzuzufügen.

Da das INSERT-Statement einen *select-expression* enthalten kann, gelten alle oben beschriebenen Db2-spezifischen **gemeinsamen Syntaxelemente**.

Weitere Einzelheiten und die Statement-Syntax finden Sie unter *INSERT (SQL)* in der Natural-*Statements*-Dokumentation.

MERGE - SQL

Das MERGE-Statement ist ein hybrides SQL-Statement, das aus einer UPDATE- und einer INSERT-Komponente besteht. Es ermöglicht Ihnen, entweder eine Zeile in eine Db2-Tabelle einzufügen oder eine Zeile einer Db2-Tabelle zu aktualisieren, wenn die Eingabedaten mit einer bereits existierenden Zeile einer Tabelle übereinstimmen.

Das MERGE-Statement gehört zum SQL Extended Set.

Weitere Einzelheiten und die Statement-Syntax finden Sie unter *MERGE (SQL)* in der Natural-*Statements*-Dokumentation.

PROCESS SQL

Das Natural PROCESS SQL-Statement wird verwendet, um SQL-Statements an die zugrunde liegende Datenbank zu senden. Die Statements werden in einem *statement-string* angegeben, der auch Konstanten und Parameter enthalten kann. Der Set von Statements, die ausgegeben werden können, wird auch als Flexible SQL bezeichnet und umfasst diejenigen Statements, die mit dem SQL-Statement EXECUTE ausgegeben werden können.

Darüber hinaus umfasst Flexible SQL die folgenden Db2-spezifischen 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
```

Anmerkungen:

1. SQL-Statements, die über PROCESS SQL ausgegeben werden, werden bei der statischen Generierung übersprungen. Sie werden daher immer dynamisch über NDBIOMO ausgeführt.
2. Um Probleme bei der Transaktionssynchronisation zwischen der Natural-Umgebung und Db2 zu vermeiden, dürfen die Statements COMMIT und ROLLBACK im PROCESS SQL nicht verwendet werden.

Weitere Einzelheiten und die Statement-Syntax finden Sie unter *PROCESS SQL* in der *Statements*-Dokumentation.

READ RESULT SET - SQL

Das Natural SQL-Statement READ RESULT SET liest eine Ergebnismenge (*result-set*), die von einer Natural Stored Procedure erzeugt wurde, die durch ein CALLDBPROC-Statement aufgerufen wurde. Wie Sie die Scroll-Richtung mit Hilfe der Variablen *scroll-hv* angeben, erfahren Sie beim SELECT-Statement.

Weitere Einzelheiten und die Statement-Syntax siehe *READ RESULT SET (SQL)* in der *Statements*-Dokumentation.

ROLLBACK - SQL

Das Natural SQL-Statement ROLLBACK macht alle Datenbankänderungen rückgängig, die seit Beginn der letzten logischen Transaktion vorgenommen wurden. Logische Transaktionen können entweder nach dem Beginn einer Sitzung oder nach dem letzten COMMIT/END TRANSACTION-Statement oder der ROLLBACK/BACKOUT TRANSACTION-Statement beginnen. Alle Datensätze, die während der Transaktion gehalten wurden, werden freigegeben.

Das Natural SQL-Statement ROLLBACK macht alle Datenbankänderungen rückgängig, die seit Beginn der letzten logischen Transaktion vorgenommen wurden. Logische Transaktionen können entweder nach dem Beginn einer Sitzung oder nach dem letzten COMMIT-/END TRANSACTION- oder der ROLLBACK-/BACKOUT TRANSACTION-Statement beginnen. Alle Datensätze, die während der Transaktion gehalten wurden, werden freigegeben.

Weitere Einzelheiten und die Statement-Syntax finden Sie unter *ROLLBACK (SQL)* in der *Statements*-Dokumentation.

ROLLBACK ist ein Synonym für das Natural-Statement BACKOUT TRANSACTION, das im Abschnitt ***Verwendung von nativen Natural DML-Statements*** beschrieben ist.

Wenn dieses Kommando von einer Natural Stored Procedure oder einer benutzerdefinierten Funktion (UDF) aus ausgeführt wird, führt Natural for Db2 die zugrunde liegende Rollback-Operation aus. Dadurch wird der Aufrufer (Caller) in einen Must-Rollback-Zustand versetzt. Wenn dieses Kommando von einer Natural-Fehlerverarbeitung (implizites ROLLBACK) ausgeführt wird, führt Natural for Db2 die zugrunde liegende Rollback-Operation nicht aus, so dass der Aufrufer den ursprünglichen Natural-Fehler erhalten kann.

Unter CICS wird das Statement ROLLBACK in einen EXEC CICS ROLLBACK-Kommando übersetzt. Bei Verwendung des File Servers werden jedoch nur die seit der letzten Terminal-E/A an der Datenbank vorgenommenen Änderungen rückgängig gemacht. Dies liegt an der CICS-spezifischen Transaktionsverarbeitung im Pseudo-Conversational Mode, siehe ***Natural for DB2 unter CICS***.

Unter IMS TM wird das ROLLBACK-Statement in ein IMS-Rollback-Kommando (ROLB) übersetzt. Es werden jedoch nur die Änderungen an der Datenbank rückgängig gemacht, die seit der letzten Terminal-E/A vorgenommen wurden. Dies liegt an der IMS TM-spezifischen Transaktionsverarbeitung, siehe ***Natural for DB2 unter IMS TM***.

Da alle Cursor am Ende einer logischen Arbeitseinheit geschlossen werden, darf ein ROLLBACK-Statement nicht innerhalb einer Datenbankschleife stehen, sondern muss außerhalb einer solchen Schleife oder bei geschachtelten Schleifen nach der äußersten Schleife stehen.

Wenn ein externes Programm, das in einer anderen Standardprogrammiersprache geschrieben wurde, von einem Natural-Programm aufgerufen wird, darf dieses externe Programm kein eigenes ROLLBACK-Kommando enthalten, wenn das Natural-Programm auch Datenbankaufrufe absetzt. Das aufrufende Natural-Programm muss das ROLLBACK-Statement für das externe Programm absetzen.

SELECT - SQL

Das Natural SQL SELECT-Statement unterstützt sowohl die cursororientierte Auswahl, mit der eine beliebige Anzahl an Zeilen abgerufen werden kann, als auch die nicht-cursororientierte Auswahl (Singleton SELECT), mit der höchstens eine einzige Zeile abgerufen werden kann.

Alle Einzelheiten und die Statement-Syntax finden Sie unter *SELECT (SQL)* in der *Statements*-Dokumentation.

SELECT - Cursor-orientiert

Wie das native DML-Statement FIND wird das cursororientierte SELECT-Statement verwendet, um eine Reihe von Zeilen (Datensätzen) aus einer oder mehreren Db2-Tabellen anhand eines Suchkriteriums auszuwählen. Da eine Datenbankschleife initiiert wird, muss die Schleife durch ein LOOP-Statement (im Reporting Mode) oder durch ein END-SELECT-Statement (im Structured Mode) geschlossen werden. Bei dieser Vorgehensweise verwendet Natural die gleiche Schleifenverarbeitung wie bei einem FIND-Statement. Außerdem ist keine Cursor-Verwaltung durch das Anwendungsprogramm erforderlich. Sie wird automatisch von Natural durchgeführt.

Weitere Einzelheiten und die Syntax finden Sie unter *Syntax 1 – Cursor-orientierte Auswahl* in *SELECT (SQL)* in der *Statements*-Dokumentation.

SELECT SINGLE - Nicht cursor-orientiert

Das Natural SQL-Statement `SELECT SINGLE` bietet die Funktionalität einer Nicht-Cursor-Auswahl (Singleton `SELECT`), d. h. eines *`select-expression`*, der maximal eine Zeile ohne Verwendung eines Cursors abruft.

Da Db2 das Singleton `SELECT`-Kommando nur in statischem SQL unterstützt, wird das Natural-`SELECT SINGLE`-Statement im dynamischen Modus wie ein Set-Level-`SELECT`-Statement ausgeführt, was zu einer Cursor-Operation führt. Natural prüft jedoch die Anzahl der von Db2 zurückgegebenen Zeilen. Wenn mehr als eine Zeile ausgewählt wird, wird eine entsprechende Fehlermeldung zurückgegeben.

Weitere Einzelheiten und die Syntax finden Sie unter *Syntax 2 - Nicht cursor-orientierte Auswahl* in *SELECT (SQL)* in der *Statements*-Dokumentation.

UPDATE - SQL

Sowohl das cursororientierte oder Positioned `UPDATE` als auch die nicht-cursororientierte oder Searched `UPDATE`-Statement wird als Teil von Natural SQL unterstützt. Beide referenzieren entweder eine Tabelle oder eine Natural-View.

Bei Db2 kann der Name einer Tabelle oder einer Natural-View, die durch ein Searched `UPDATE` referenziert werden soll, mit einem *`correlation-name`* versehen werden. Dieser entspricht nicht der Standard-SQL-Syntaxdefinition und gehört daher zum Natural Extended Set.

Das Searched `UPDATE` Statement muss z.B. zur Aktualisierung eines Primärschlüsselfeldes verwendet werden, da Db2 die Aktualisierung von Spalten eines Primärschlüssels mit einem Positioned `UPDATE`-Statement nicht erlaubt.

Anmerkung: Wenn Sie die Notation `SET *` verwenden, werden alle Felder der referenzierten Natural-View zu den Listen `FOR UPDATE OF` und `SET` hinzugefügt. Stellen Sie daher sicher, dass Ihre View nur Felder enthält, die aktualisiert werden können. Andernfalls wird von Db2 ein negativer SQLCODE zurückgegeben.

Weitere Einzelheiten und die Syntax finden Sie unter *UPDATE (SQL)* in der *Statements*-Dokumentation.

Verwendung von Natural-Systemvariablen

Bei der Verwendung mit Db2 gibt es Einschränkungen und/oder zu beachtende Punkte in Bezug auf die folgenden Natural-Systemvariablen:

- `*ISN`
- `*NUMBER`
- `*ROWCOUNT`

Informationen über Einschränkungen und/oder zu beachtende Punkte finden Sie im Abschnitt *Datenbank-spezifische Anmerkungen* in der entsprechenden *Systemvariablen*-Dokumentation.

Verarbeitung mehrerer Zeilen

Dieser Abschnitt beschreibt die Multiple Row Processing-Funktionalität für Db2 Datenbanken.

Sie müssen mit Db2 for z/OS Version 8 oder höher arbeiten, um diese Funktionalität nutzen zu können.

Natural for Db2 bietet zwei Arten von Funktionen zur Verarbeitung mehrerer Zeilen:

- **Standard-Mehrzeilenverarbeitung**
- Diese Funktion hat keinen Einfluss auf die Programmlogik. Obwohl Natural Native DML und Natural SQL DML Klauseln zur Angabe des Multi-Fetch-Faktors bereitstellen, arbeitet das Natural-Programm mit einer Datenbankzeile und aus Sicht des Programms wird nur eine Zeile von der Datenbank empfangen oder an die Datenbank gesendet.
- **Erweiterte Mehrzeilenverarbeitung**

  Diese Funktion ist nur mit Natural SQL DML verfügbar und hat erhebliche Auswirkungen auf die Programmlogik, da sie das Abrufen mehrerer Zeilen aus der Datenbank in den Programmspeicher durch ein einziges Natural SQL `SELECT`-Statement in ein Set von Arrays ermöglicht. Außerdem ist es möglich, mehrere Zeilen aus einem Set von Arrays mit dem Natural SQL `INSERT`-Statement in die Datenbank einzufügen.

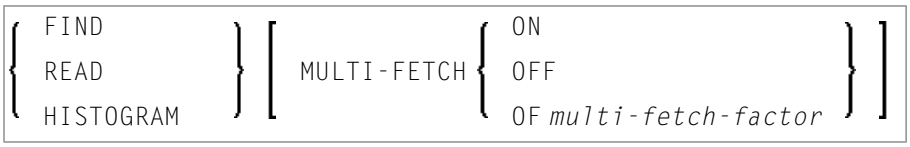
Im Folgenden finden Sie Informationen zu den folgenden Themen:

- Mehrere Zeilen aus Programm (Erweitert)

Zweck der Multi-Fetch-Funktion (Standard)

Im Standardmodus liest Natural nicht mehrere Datensätze mit einem einzigen Datenbankaufruf, sondern arbeitet immer im Modus „Ein Datensatz pro Abruf“. Diese Art des Betriebs ist zuverlässig und stabil, kann aber einige Zeit in Anspruch nehmen, wenn eine große Anzahl von Datenbanksätzen verarbeitet wird.

Um die Leistung dieser Programme zu verbessern, können Sie die Multi-Fetch-Klausel in den Natural DML-Statements `FIND`, `READ` oder `HISTOGRAM` verwenden. Damit können Sie die Anzahl der pro Datenbankzugriff gelesenen Datensätze angeben.



wobei *multi-fetch-factor* entweder eine Konstante oder eine Variable mit dem Format Integer (I4) ist.

Um die Performance des Natural SQL `SELECT`-Statements zu erhöhen, können Sie die `WITH ROWSET POSITIONING FOR`-Klausel verwenden, um einen Multi-Fetch-Faktor anzugeben.

Zur Ausführungszeit des Statements prüft die Laufzeit, ob ein Multi-Fetch-Faktor größer als 1 für das Datenbank-Statement angegeben ist.

Ist der *multi-fetch-factor*

kleiner oder gleich 1,	wird der Datenbankaufruf im üblichen Modus „Ein Datensatz pro Zugriff“ fortgesetzt.
größer als 1,	wird der Datenbankaufruf dynamisch vorbereitet, um mehrere Sätze (z.B. 10) mit einem einzigen Datenbankzugriff in einen Hilfspuffer (Multi-Fetch-Puffer) zu lesen. Bei Erfolg wird der erste Datensatz in die zugrunde liegende Datensicht übertragen. Bei der Ausführung der nächsten Schleife wird die Datensicht direkt aus dem Multi-Fetch-Puffer gefüllt, ohne Datenbankzugriff. Nachdem alle Datensätze aus dem Multi-Fetch-Puffer geholt wurden, bewirkt die nächste Schleife, dass der nächste Datensatz aus der Datenbank gelesen wird. Wird die Datenbankschleife beendet (entweder durch End-of-Records, `ESCAPE`, `STOP` usw.), wird der Inhalt des Multi-Fetch-Puffers freigegeben.

Bei Multi-Fetch-Nutzung (Standard) zu berücksichtigende Punkte

- Das Programm erhält nicht bei jeder Schleife „frische“ Datensätze aus der Datenbank, sondern arbeitet mit Abbildern, die beim letzten Multi-Fetch-Zugriff abgerufen wurden.
- Wenn für ein Natural DML `READ`- oder `HISTOGRAM`-Statement ein dynamischer Richtungswechsel (`IN DYNAMIC...SEQUENCE`) kodiert wird, ist die Multi-Fetch-Funktion nicht möglich und führt zu einem entsprechenden Syntaxfehler beim Kompilieren.
- Die Größe, die eine Datenbankschleife im Multi-Fetch-Puffer belegt, wird nach der folgenden Regel ermittelt:

```
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))
```

Dabei ist:

header	Länge des Header eines Eintrags im Db2-Multifetch-Puffer, d.h. 32
sqldaheader	Länge des Header einer sqlda, d.h. 16
columns	Anzahl der aufnehmenden Felder einer SQL-Anfrage
sqlvar	Länge eines sqlvar, d. h. 44
lise	Länge einer Natrual für Db2 spezifischen sqlvar-Erweiterung
mf	Multifetch-Faktor, d.h. die Anzahl der Zeilen, die durch einen Datenbankaufruf geholt werden
collen	Länge des aufnehmenden Feldes
LF(column)	Größe des Längenfeldes des aufnehmenden Feldes, d. h. 0 für Felder mit fester Länge, 2 für Felder mit variabler Länge und 4 für große Objektspalten (LOBs)
nullind	Länge eines Null-Indikators, d. h. 2

Größe des Multi-Fetch-Puffers (Standard)

Der Multi-Fetch-Puffer wird bei der Terminal-Eingabe im Pseudo-Conversional Mode freigegeben. Daher gibt es keine Größenbeschränkung für den Db2-Multi-Fetch-Puffer (`DB2SIZE6`). Der Puffer wird bei Bedarf automatisch vergrößert.

Unterstützung von TEST DBLOG Q (Standard)

Wenn Multi-Fetch verwendet wird, werden „reale“ Datenbankaufrufe nur übermittelt, um einen neuen Satz an Datensätzen zu erhalten.

Die Funktion `TEST DBLOG Q` wird auch vom Natural for Db2 Multi-Fetch-Handler für jeden Rowset-Fetch aus Db2 und für jeden aus dem Multi-Fetch-Puffer in den Programmspeicher verschobenen Datensatz aufgerufen. Die Ereignisse werden durch das Literal `MULTI FETCH ...` und `<BUFF FETCH ...` unterschieden.

Beispiel: 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 ===>
```

wobei die Spalte **No** für Folgendes steht:

1	ist ein offener Cursor-Db2-Aufruf.
2	ist ein „realer“ Datenbankaufruf, der eine Reihe von Datensätzen über Multi-Fetch liest (siehe `MULTI FETCH NEX` in der Spalte **SQL Statement**).
3-17	sind keine „realen“ Datenbankaufrufe, sondern nur Einträge, die dokumentieren, dass das Programm diese Sätze aus dem Multi-Fetch-Puffer erhalten hat (siehe `<BUFF FETCH NEX` in der Spalte **SQL-Statement**).

Mehrere Zeilen an Programm (erweitert)

Diese Funktion ermöglicht es Programmen, mehrere Zeilen aus Db2 in Arrays abzurufen.

Diese Funktion ist nur beim SELECT-Statement verfügbar.

Voraussetzungen

Um diese Funktion zu nutzen:

1. Setzen Sie die Compiler-Option DB2ARRY=ON (unter Verwendung eines OPTIONS-Statements oder des COMPOPT-Kommandos oder des Profilparameters CMPO).
2. Geben Sie eine Liste von empfangenden Arrays in der INTO-Klausel (siehe *into-clause*) des SELECT-Statements an.
3. Geben Sie die Anzahl der Zeilen an, die mit einer einzigen FETCH-Operation mit der WITH ROWSET POSITIONING-Klausel aus der Datenbank abgerufen werden sollen.
4. Geben Sie eine Variable an, die die Anzahl der aus der Datenbank abgerufenen Zeilen über die ROWS_RETURNED-Klausel erhält.

DB2ARRY=ON

DB2ARRY=ON ist notwendig, um die Angabe von Arrays in der INTO-Klausel zu ermöglichen (siehe *into-clause*).

DB2ARRY=ON verhindert auch die Verwendung von Arrays als abgebende oder aufnehmende Felder bei Db2 CHAR/VARCHAR /GRAPHIC/VARGRAPHIC-Spalten. Stattdessen müssen Natural-Skalarfelder mit der entsprechenden Länge verwendet werden.

INTO Clause

Jedes in der INTO-Klausel (siehe *into-clause*) angegebene Array muss zusammenhängend sein (eine Ausprägung folgt unmittelbar auf eine andere, dies wird von Db2 erwartet) und muss eindimensional sein. Die Arrays werden von der ersten Ausprägung (low) bis zur letzten Ausprägung (high) aufgefüllt. Die ersten Array-Ausprägungen bilden die erste Zeile des erhaltenen Rowset, die zweiten Array-Ausprägungen bilden die zweite Zeile des erhaltenen Rowset. Die Array-Ausprägungen des n-ten Index bilden die n-te Zeile, die von Db2 zurückgegeben wird.

Wenn eine *LINDICATOR-Klausel* oder *INDICATOR-Klausel* in der INTO-Klausel für Arrays verwendet wird, müssen die angegebenen Längenindikatoren oder Nullindikatoren ebenfalls Arrays sein. Die Anzahl der Ausprägungen von LINDICATOR- und INDICATOR-Arrays muss gleich oder größer sein als die Anzahl der Ausprägungen des Master-Arrays.

WITH ROWSET POSITIONING-Klausel

Die WITH_ROWSET_POSITIONING-Klausel wird verwendet, um die Anzahl der Zeilen anzugeben, die in einem Verarbeitungszyklus aus der Datenbank abgerufen werden sollen. Die angegebene Anzahl muss gleich oder kleiner sein als das Minimum der Ausprägungen aller angegebenen Arrays. Wird eine Variable, nicht eine Konstante, angegeben, so wird der aktuelle Inhalt der Variablen bei jedem Verarbeitungszyklus verwendet. Die angegebene Zahl muss größer als 0 und kleiner als 32768 sein.

ROWS_RETURNED-Klausel

Die ROWS_RETURNED-Klausel wird verwendet, um eine Variable anzugeben, die die Anzahl der Zeilen enthält, die während der eigentlichen Fetch-Operation aus der Datenbank gelesen wurden. Das Format der Variablen muss I4 sein.

Einschränkungen und Vorbehalte

Natural-Views: Es ist nicht möglich, Natural-Arrays von Views in der INTO-Klausel zu verwenden (siehe *into-clause*), d.h. die Verwendung des Schlüsselworts VIEW ist nicht möglich.

File Server-Verwendung und Positioned UPDATE und DELETE

Der Zweck dieses Merkmals besteht darin, die Anzahl der Datenbank- und Datenbankschnittstellenaufrufe für die Massenstapelverarbeitung zu reduzieren. Daher ist es nicht empfehlenswert, diese Art der Programmierung in Online-CICS- oder IMS-Umgebungen zu verwenden, wenn Terminal-Ein- und -Ausgaben innerhalb offener Cursorschleifen erfolgen, d.h. der File Server verwendet wird. Erst recht ist es nicht möglich, nach einer Terminal-E/A ein Positioned UPDATE- oder Positioned DELETE-Statement durchzuführen.

Beispiel:

```
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
```

Mehrere Zeilen aus Programm (Erweitert)

Diese Funktion ermöglicht es Programmen, mehrere Zeilen aus Arrays in eine Db2-Tabelle einzufügen.

Diese Funktion ist nur mit dem Natural-SQL-`INSERT`-Statement verfügbar.

Voraussetzungen

Um dieses Merkmal zu nutzen:

1 Setzen Sie die Compiler-Option `DB2ARRY=ON` (mit einem `OPTIONS`-Statement oder dem `COMPOPT`-Kommando oder dem Profilparameter `CMPO`).

2 Geben Sie eine Liste von sendenden Arrays in der *VALUES-Klausel* des Natural-SQL-`INSERT`-Statement an.

3 Geben Sie mit der *FOR n ROWS-Klausel* die Anzahl der Zeilen an, die mit einem einzelnen Natural-SQL-`INSERT`-Statement in die Datenbank eingefügt werden sollen.

DB2ARRY=ON

`DB2ARRY=ON` ist notwendig, um die Angabe von Arrays in der *VALUES-Klausel* zu ermöglichen. `DB2ARRY=ON` verhindert auch die Verwendung von Arrays als abgebende oder aufnehmende Felder für Db2 `CHAR`/`VARCHAR`/`GRAPHIC`/`VARGRAPHIC`-Spalten. Stattdessen müssen Natural-Skalarfelder mit der entsprechenden Länge verwendet werden.

VALUES-Klausel

Jedes in der *VALUES-Klausel* angegebene Array muss zusammenhängend sein (eine Ausprägung folgt unmittelbar auf eine andere, dies wird von Db2 erwartet) und muss eindimensional sein. Die Arrays werden von der ersten Ausprägung (niedrig) bis zur letzten Ausprägung (hoch) gelesen. Die ersten Ausprägungen des Arrays bilden die erste in die Datenbank eingefügte Zeile, die zweiten Ausprägungen des Arrays bilden die zweite in die Datenbank eingefügte Zeile. Die Array-Ausprägungen des n-ten Index bilden die n-te in die Datenbank eingefügte Zeile.

Wenn eine *LINDICATOR-Klausel* oder *INDICATOR-Klausel* in der *VALUES-Klausel* für Arrays verwendet wird, müssen die angegebenen Längenindikatoren oder Nullindikatoren ebenfalls Arrays sein. Die Anzahl der `LINDICATOR`- und `INDICATOR`-Array-Ausprägungen muss gleich oder größer sein als die Anzahl der Ausprägungen des Master-Array.

FOR n ROWS-Klausel

Die *FOR n ROWS-Klausel* wird verwendet, um anzugeben, wie viele Zeilen durch eine `INSERT`-Anweisung in die Datenbanktabelle eingefügt werden sollen. Die angegebene Anzahl muss gleich der oder kleiner sein als die Mindestanzahl der Ausprägungen aller angegebenen Arrays in der *VALUES-Klausel*. Die angegebene Zahl muss größer als `0` und kleiner als `32768` sein.

Einschränkungen und Vorbehalte

- **Natural-Views**

  Es ist nicht möglich, Natural-Arrays von Views in der *VALUES-Klausel* zu verwenden, d.h. die Verwendung des Schlüsselworts `VIEW` ist nicht möglich.

- **Statische Ausführung**

  Aufgrund von Db2-Einschränkungen ist es nicht möglich, Einfügungen mehrerer Zeilen im statischen Modus auszuführen. Daher werden mehrzeilige Einfügungen nicht statisch generiert, sondern immer dynamisch von Natural for Db2 vorbereitet und ausgeführt.

  Die statische Generierung mit Natural for Db2 erzeugt ein SQL-Programm in Assemblersprache, das vom Db2-Precompiler vorkompiliert wird, der wiederum ein DBRM erzeugt, das für die statische Ausführung erforderlich ist. Der Db2-Assembler-Precompiler bietet jedoch keine Unterstützung für Host-Variablen-Arrays. Sie können nur verwendet werden, wenn sie in einer SQLDA-Struktur angegeben sind, die zur Ausführungszeit erstellt werden muss. Ein statisches `INSERT` mit einer `VALUES`-Klausel für mehrere Zeilen erlaubt jedoch nicht die Angabe einer SQLDA, sondern nur Host-Variablen-Arrays, die vom Db2-Assembler-Precompiler nicht unterstützt werden.

  Es ist nicht möglich, Natural-Arrays von Views in der `INTO`-Klausel zu verwenden (siehe *into-clause*), d.h. die Verwendung des Schlüsselworts `VIEW` ist nicht möglich.

  Beispiel:

```
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
```

Fehlerbehandlung

Im Gegensatz zur normalen Natural-Fehlerbehandlung, bei der entweder ein `ON ERROR`-Statement verwendet wird, um Ausführungszeitfehler abzufangen, oder eine Standard-Fehlermeldung verarbeitet und die Programmausführung beendet wird, bietet die erweiterte Fehlerbehandlung von Natural for Db2 eine anwendungsgesteuerte Reaktion auf den aufgetretenen SQL-Fehler.

Zwei Natural-Subprogramme, `NDBERR` und `NDBNOERR`, werden bereitgestellt, um die übliche Natural-Fehlerbehandlung zu deaktivieren und den aufgetretenen SQL-Fehler auf den zurückgegebenen SQLCODE zu prüfen. Diese Funktionalität ersetzt die `E`-Funktion der `DB2SERV`-Schnittstelle, die weiterhin zur Verfügung steht, aber nicht mehr dokumentiert ist.

Weitere Informationen zu den Natural-Subprogrammen, die für Db2 zur Verfügung stehen, finden Sie im Abschnitt ***Interface-Subprogramme***.

Beispiel:

```
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 Verarbeitung von Natural Stored Procedures und UDFs

Natural for Db2 unterstützt das Schreiben und Ausführen von Natural Stored Procedures und Natural User-Defined Functions (Natural UDFs).

Natural Stored Procedures sind vom Benutzer geschriebene Programme, die durch das SQL-Statement `CALL` aufgerufen und von Db2 im SPAS (Stored Procedure Address Space) ausgeführt werden. SPAS ist ein separater Adressraum, der für Stored Procedures reserviert ist.

Eine Function ist eine Operation, die durch einen Funktionsnamen, gefolgt von null oder mehr Operanden, die in Klammern eingeschlossen sind, bezeichnet wird. Eine Function stellt eine Beziehung zwischen einer Menge von Eingabewerten und einer Menge von Ergebniswerten dar. Wenn eine Function durch ein vom Benutzer geschriebenes Programm implementiert wurde, wird sie in Db2 als User-Defined Function (UDF) bezeichnet.

Die folgenden Themen werden in diesem Kapitel behandelt:

Natural-UDF-Typen

Es gibt zwei Typen von Natural Used Defined Functions (UDF):

- **Skalare UDF**

  Die skalare UDF akzeptiert mehrere Eingangsargumente und gibt einen Ausgangswert zurück. Sie kann durch ein beliebiges SQL-Statement so wie eine eingebaute Db2-Funktion (Db2 Built-in Function) aufgerufen werden.

- **Tabellen-UTF**

  Die Tabellen-UDF akzeptiert mehrere Eingabeargumente und gibt bei jedem Aufruf eine Menge von Ausgabewerten zurück, die eine Tabellenzeile umfassen.

  Sie können eine Tabellen-UDF mit einem Natural-SQL-`SELECT`-Statement aufrufen, indem Sie den Namen der Tabellenfunktion in der *FROM-Klausel* angeben. Eine Tabellen-UDF verhält sich wie eine Db2-Tabelle und wird bei jeder `FETCH`-Operation für die in dem `SELECT`-Statement angegebene Tabellenfunktion aufgerufen.

PARAMETER STYLE

Mit `PARAMETER STYLE` wird die Linkage-Konvention identifiziert, die zur Übergabe von Parametern an eine Db2 Stored Procedure oder eine Db2 User Defined Function (UDFs) verwendet wird.

Dieser Abschnitt beschreibt die `PARAMETER STYLE`s und die STCB, die Natural for Db2 für die Verarbeitung von Natural for Db2 Stored Procedures oder Natural UDFs verwendet.

Anmerkung: `PARAMETER STYLE GENERAL` (oder `GENERAL WITH NULL`) und **STCB Layout** gelten nur für Natural Stored Procedures.

- GENERAL und GENERAL WITH NULL
- STCB Layout
- DB2SQL

GENERAL und GENERAL WITH NULL

Anmerkung: Gilt nur für Natural Stored Procedures

Eine Natural Stored Procedure, die mit `PARAMETER STYLE GENERAL` definiert ist, erhält nur die angegebenen Benutzerparameter.

Eine Natural Stored Procedure, die mit `PARAMETER STYLE GENERAL WITH NULL` definiert ist, erhält die angegebenen Benutzerparameter und zusätzlich ein `NULL`-Indikator-Array, das einen `NULL`-Indikator für jeden Benutzerparameter enthält.

Natural Stored Procedures, die mit `PARAMETER STYLE GENERAL`/`PARAMETER STYLE GENERAL WITH NULL` definiert sind, erfordern, dass die Definition der Stored Procedure im Db2-Katalog einen zusätzlichen Parameter vom Typ `VARCHAR` vor den Benutzerparametern der Stored Procedure enthält.

Dieser Parameter vor den Parametern ist der Stored Procedure Control Block (STCB), siehe auch ***STCB Layout*** weiter unten.

Nachfolgend finden Sie Informationen über:

- Stored Procedure Control Block
- Beispiel für PARAMETER STYLE GENERAL
- Beispiel für GENERAL WITH NULL

Stored Procedure Control Block

Der Stored Procedure Control Block (STCB) enthält Informationen, die der Stub des Natural for Db2-Servers zur Ausführung von Natural Stored Procedures verwendet, z. B. die Library und das aufzurufende Subprogramm. Er enthält auch die Formatbeschreibungen der Parameter, die an die Stored Procedure übergeben werden.

Die STCB ist für die aufgerufene Natural Stored Procedure unsichtbar. Die STCB wird vom Natural for Db2 Server Stub ausgewertet und aus der Parameterliste entfernt, die an die Natural Stored Procedure übergeben wird.

Wenn der Aufrufer einer Natural Stored Procedure, die mit `PARAMETER STYLE GENERAL`/`PARAMETER STYLE GENERAL WITH NULL` definiert ist, ein Natural-Programm ist, muss das Programm ein Natural-SQL-Statement `CALLDBPROC` mit dem Schlüsselwort `CALLMODE=NATURAL` verwenden.

Wenn der Aufrufer der Natural Stored Procedure kein Natural-Programm ist, muss der Aufrufer die STCB für das Db2 `CALL`-Statement einrichten und die STCB als ersten Parameter übergeben.

Wenn bei der Ausführung einer Natural Stored Procedure, die mit `PARAMETER STYLE GENERAL`/`PARAMETER STYLE GENERAL WITH NULL` definiert ist, ein Fehler auftritt, wird der Text der Fehlermeldung an die STCB zurückgegeben.

Wenn der Aufrufer ein Natural-Programm ist, das `CALLDBPROC` und `CALLMODE=NATURAL` verwendet, dann fügt die Natural for Db2-Laufzeit den Fehlertext in die Fehlermeldung NAT3286 ein.

Beispiel für PARAMETER STYLE GENERAL

Definieren Sie in der Natural Stored Procedure die Parameter wie im folgenden Beispielprogramm gezeigt:

```
DEFINE DATA PARAMETER
01 P1 ...
01 P2 ...
...
...
01 Pn ...
LOCAL
...
...
END-DEFINE
```

Beispiel für GENERAL WITH NULL

Definieren Sie in der Natural Stored Procedure die Parameter wie im folgenden Beispielprogramm gezeigt:

```
DEFINE DATA PARAMETER
01 P1 ...
01 P2 ...
...
...
01 Pn ...
01 NULL-INDICATOR-ARRAY  (I2/1:n)
LOCAL
...
...
END-DEFINE
```

STCB Layout

Anmerkung: Gilt nur für Natural Stored Procedures

Die folgende Tabelle beschreibt den ersten Parameter, der zwischen dem Aufrufer und der Natural Stored Procedure übergeben wird, wenn `CALLMODE=NATURAL` in einem Natural SQL `CALLDBPROC`-Statement angegeben ist.

Name	Format	Verarbeitungsmodus-Server
`STCBL`	I2	Eingabe (Größe der folgenden Informationen)
Prozedur-Informationen		
`STCBLENG`	A4	Eingabe (druckbare STCBL)
`STCBID`	A4	Eingabe (STCB)
`STCBVERS`	A4	Eingabe (Version von STCB 310)
`STCBUSER`	A8	Eingabe (Benutzerkennung)
`STCBLIB`	A8	Eingabe (Library)
`STCBPROG`	A8	Eingabe (aufrufendes Programm)
`STCBPSW`	A8	Ungenutzt (Passwort)
`STCBSTNR`	A4	Eingabe (`CALLDBPROC`-Statement-Nummer)
`STCBSTPC`	A8	Eingabe (aufgerufene Prozedur)
`STCBPANR`	A4	Eingabe (Anzahl der Parameter)
Fehlerinformationen		
`STCBERNR`	A5	Ausgabe (Natural-Fehlernummer)
`STCBSTAT`	A1	Ungenutzt (Natural-Fehlerstatus)
`STCBLIB`	A8	Ungenutzt (Natural-Fehler-Library)
`STCBPRG`	A8	Ungenutzt (Natural-Fehlerprogramm)
`STCBLVL`	A1	Ungenutzt (Natural-Fehlerebene)
`STCBOTP`	A1	Ungenutzt (Fehlerobjekttyp)
`STCBEDYL`	A2	Ausgabe (Fehlertextlänge)
`STCBEDYT`	A88	Ausgabe (Fehlertext)
	A100	Reserviert für zukünftige Verwendung
Parameter Informationen		
`STCBPADE`	A variable	Eingabe. Siehe auch *PARAMETER DESCRIPTION (STCBPADE)* weiter unten.

PARAMETER DESCRIPTION (STCBPADE)

`PARAMETER DESCRIPTION` enthält eine Beschreibung zu jedem an die Natural Stored Procedure übergebenen Parameter, bestehend aus Parametertyp, Formatangabe und Länge. Der Parametertyp ist das Attribut `AD` des Natural-`CALLNAT`-Statements, wie in der Natural-*Statements*-Dokumentation beschrieben.

Jeder Parameter hat das folgende Formatbeschreibungselement in der STCBPADE-Zeichenkette

at`l,p[,`*d1*`]....`

Bedeutung:

- *a* ist eine Attributmarkierung, die den Parametertyp angibt:

Markierung	Typ	Äquivalentes AD-Attribut	Äquivalente DB2-Klausel
`M`	änderbar	`AD=M`	`INOUT`
`O`	nicht änderbar	`AD=O`	`IN`
`A`	nur Eingabe	`AD=A`	`OUT`

- *t* ist eines der folgenden Natural-Format-Token:

t	Beschreibung	*l*	*p*	*d1*	Beispiel
A	Alphanumerisch	1-253	0	1-32767 oder -	A30,0 oder A30,0,10
N	Numerisch, nicht gepackt	1-29	0-7	-	N10,3
P	Numerisch, gepackt	1-29	0-7	-	P13,4
I	Ganzzahl	2 oder 4	0	-	I2,0
F	Floating point		0	-	I4,0
B	Binär		0	-	B23,0
D	Datum	6	0	-	D6
T	Uhrzeit	12	0	-	T12
L	Logisch (wird nicht unterstützt)				

- *l* ist eine Ganzzahl, die die Länge/Skalierung des Feldes angibt.

  Bei numerischen und gepackten numerischen Feldern gibt *l* die Gesamtzahl der Ziffern des Feldes an, d. h. die Summe der Ziffern links und rechts vom Dezimalpunkt. Das Natural-Format N7.3 wird z. B. durch N10.3 dargestellt. Siehe auch die obige **Tabelle**.

- *p* ist eine Ganzzahl, die die Genauigkeit des Feldes angibt. Normalerweise ist sie `0`, außer bei numerischen und gepackten Feldern, wo sie die Anzahl der Stellen rechts vom Dezimalpunkt angibt. Siehe auch die obige **Tabelle**.
- *d1* ist ebenfalls eine Ganzzahl, die die Anzahl der Ausprägungen im alphanumerischen Feld angibt (nur alphanumerisch). Siehe auch die obige **Tabelle**.

Dieser Beschreibungs-/Kontrollparameter ist für das aufrufende Natural-Programm und die aufgerufene Natural-Stored-Procedure unsichtbar, muss aber in der Parameterdefinition der Stored-Procedure-Zeile mit dem `CREATE PROCEDURE`-Statement und dem Db2 `PARAMETER STYLE GENERAL/PARAMETER STYLE GENERAL WITH NULL` definiert werden.

Die folgende Tabelle zeigt die Anzahl der Parameter, die mit dem `CREATE PROCEDURE`-Statement für eine mit `PARAMETER STYLE GENERAL` definierte Natural Stored Procedure definiert werden müssen, abhängig von der Anzahl der Benutzerparameter und davon, ob der Client (d.h. der Aufrufer einer Stored Procedure für Db2) und der Server (d.h. die Stored Procedure für Db2) in Natural oder in einer anderen Standardprogrammier-Hostsprache geschrieben ist. *n* bezeichnet die Anzahl der Benutzerparameter.

Client\Server	Natural	Nicht Natural
Natural	*n* + 1	*n* (`CALLMODE=NONE`)
non-Natural	*n* + 1	*n*

DB2SQL

Anmerkung: `PARAMETER DB2SQL` gilt für Natural Stored Procedures und Natural UDFs.

Eine Natural Stored Procedure oder Natural User Defined Function (UDF) mit `PARAMETER STYLE DB2SQL` empfängt zuerst die angegebenen Benutzerparameter und dann die unten unter *Zusätzlich übergebene Parameter* aufgeführten Parameter. Bei einer Natural UDF werden die Eingabeparameter vor den Ausgabeparametern übergeben.

Zusätzlich übergebene Parameter

- Ein `NULL`-Indikator für jeden Benutzerparameter des `CALL`-Statement,
- der SQLSTATE, der an Db2 zurückgegeben wird,
- der qualifizierte Name der Natural Stored Procedure oder UDF,
- der spezifische Name der Natural Stored Procedure oder UDF,
- das Feld SQL `DIAGNOSE` mit einer an Db2 zurückzugebenden Diagnosezeichenfolge.

Der SQLSTATE, der qualifizierte Name, der spezifische Name und das `DIAGNOSE`-Feld sind im Natural-Parameterdatenbereich (PDA) `DB2SQL_P` definiert, der in der Natural System Library SYSDB2 bereitgestellt wird.

Wenn im CREATE FUNCTION-Statement für die Natural UDF zusätzlich das optionale Merkmal SCRATCHPAD *nnn* angegeben ist, wird der Speicherparameter SCRATCHPAD an die Natural UDF übergeben.

Verwenden Sie die folgenden Definitionen:

```
01 SCRATCHPAD A(4+nnn)
01 REDEFINE SCRATCHPAD
02 SCRATCHPAD_LENGTH(I4)
02 ...
```

Definieren Sie das SCRATCHPAD in der Natural UDF entsprechend Ihren Anforderungen um.

Die ersten vier Bytes des SCRATCHPAD-Bereichs enthalten ein Integer-Längenfeld. Vor dem ersten Aufruf der Natural UDF mit einem SQL-Statement setzt Db2 den SCRATCHPAD-Bereich auf x'00' zurück und setzt die für den SCRATCHPAD-Bereich angegebene Größe *nnn* in die ersten vier Bytes als Integer-Wert.

Danach initialisiert Db2 den SCRATCHPAD-Bereich zwischen den Aufrufen der Natural UDF für das aufrufende SQL-Statement nicht neu. Stattdessen wird der Scratchpad-Inhalt nach der Rückkehr von der Natural UDF beibehalten und beim nächsten Aufruf der Natural UDF wiederhergestellt.

Nachfolgend finden Sie Informationen über:

Parameter CALL TYPE

Anmerkung: Dieser Parameter ist optional und gilt nur für Natural UDFs.

Der Parameter CALL TYPE wird übergeben, wenn die Option FINAL CALL für eine skalare Natural UDF angegeben ist oder wenn die Natural UDF eine Tabellen-UDF ist. Der Parameter CALL TYPE ist eine Ganzzahl, die den Typ des Aufrufs angibt, den Db2 für die Natural UDF durchführt.

Einzelheiten zu den im Parameter CALL_TYPE angegebenen Parameterwerten finden Sie im *Db2 SQL GUIDE*.

Parameter DBINFO

Dieser Parameter ist optional.

Wenn die Option DBINFO verwendet wird, wird die DBINFO-Struktur an die Natural-Stored-Procedure oder UDF übergeben. Die DBINFO-Struktur ist in der Natural-PDA DBINFO_P beschrieben, die in der Natural-System-Library SYSDB2 enthalten ist.

Ermitteln von Library, Subprogramm und Parameterformaten

Der Server-Stub von Natural for Db2 ermittelt das Subprogramm und die Library aus dem qualifizierten und spezifischen Namen der Natural Stored Procedure oder UDF. Der SCHEMA-Name wird als Library-Name und der Prozedur- oder Funktionsname als Subprogramm-Name verwendet.

Das Subprogramm ROUTINEN wird in der Natural Library SYSDB2 bereitgestellt. Mit diesem Subprogramm wird auf den Db2-Katalog zugegriffen, um die Formate der für die Natural Stored Procedure oder UDF definierten Benutzerparameter zu ermitteln. Nachdem die Formate ermittelt wurden, werden sie im Natural-Buffer Pool gespeichert. Bei nachfolgenden Aufrufen der Natural Stored Procedure werden die Formate dann aus dem Natural Buffer Pool abgerufen. Dies erfordert, dass für Natural Stored Procedures oder UDFs mit PARAMETER STYLE DB2SQL mindestens READS SQL DATA angegeben ist.

Das Subprogramm ROUTINEN wird statisch generiert. Das DBRM von ROUTINEN ist als Package in der COLLECTION SAGNDBROUTINENPACK gebunden. Bevor der Zugriff auf den Db2-Katalog beginnt, speichert das Subprogramm das CURRENT PACKAGESET und setzt SAGNDBROUTINENPACK auf CURRENT PACKAGESET. Nach der Verarbeitung stellt das Subprogramm ROUTINEN das gespeicherte CURRENT PACKAGESET wieder her.

Aufrufen einer Natural Stored Procedure

Wenn der Aufrufer der Natural Stored Procedure mit PARAMETER STYLE DB2SQL ein Natural-Programm ist, muss der Aufrufer das Natural-SQL-Statement CALLDBPROC mit der Angabe CALLMODE=NATURAL verwenden, was der Standard ist.

Fehlerbehandlung

Wenn während der Ausführung einer Natural Stored Procedure oder UDF mit PARAMETER STYLE DB2SQL ein Natural-Laufzeitfehler auftritt, wird der SQLSTATE auf 38N99 gesetzt und der Diagnose-String enthält den Text der Natural-Fehlermeldung.

Wenn während der Ausführung einer Natural Stored Procedure oder UDF mit PARAMETER STYLE DB2SQL ein Fehler im Natural for Db2 Server Stub auftritt, wird der SQLSTATE auf 38S99 gesetzt und der Diagnosestring enthält den Text der Fehlermeldung.

Wenn die Anwendung eine Fehlerbedingung während der Ausführung einer Natural Stored Procedure oder UDF auslösen will, muss der Parameter SQLSTATE auf einen anderen Wert als

'00000' gesetzt werden. Im entsprechenden *Db2 SQL Guide* finden Sie Angaben zu Benutzerfehlern im SQLSTATE-Parameter.

Zusätzlich kann ein Text, der die Fehler beschreibt, in den DIAGNOSE-Parameter eingegeben werden.

Wenn eine Natural-Tabellen-UDF Db2 signalisieren will, dass sie keine Zeile gefunden hat, die sie zurückgeben kann, muss im SQLSTATE-Parameter '02000' zurückgegeben werden.

Lebensdauer der Natural-Sitzung

Bei einer Natural-UDF, die die Attribute DISALLOW PARALLEL und FINAL CALL enthält, behält der Natural for Db2 Server Stub die zuvor zugeordnete Natural-Sitzung bei. Diese Natural-Sitzung wird dann von allen nachfolgenden UDF-Aufrufen wiederverwendet, bis Natural auf den letzten Aufruf trifft.

Beispiel für DB2SQL - Natural Stored Procedure

Definieren Sie in einer Natural Stored Procedure die Parameter wie im folgenden Beispielprogramm gezeigt:

```
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
```

Beispiel für DB2SQL - Natural UDF

Definieren Sie in einer Natural-UDF die Parameter wie im folgenden Beispielprogramm gezeigt:

```
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
```

Schreiben einer Natural Stored Procedure

Dieser Abschnitt enthält einen allgemeinen Leitfaden, wie eine Natural Stored Procedure geschrieben wird und was dabei zu beachten ist.

Um eine Natural Stored Procedure zu schreiben:

1 Bestimmen Sie das Format und die Attribute der Parameter, die zwischen dem Aufrufer und der Stored Procedure übergeben werden. Erwägen Sie die Erstellung eines Natural-Parame-

terdatenbereichs (PDA). Stored Procedures unterstützen keine Datengruppen und Neudefinitionen innerhalb ihrer Parameter.

2 Bestimmen Sie den `PARAMETER STYLE` der Stored Procedure: `GENERAL`, `GENERAL WITH NULL` oder `DB2SQL`.

- Wenn Sie `GENERAL WITH NULL` verwenden, hängen Sie die Parameter an die Natural Stored Procedure an, indem Sie ein `NULL`-Indikator-Array definieren, das einen `NULL`-Indikator (I2) für jeden anderen Parameter enthält.
- Wenn Sie `DB2SQL` verwenden, fügen Sie die Parameter der Natural Stored Procedure an, indem Sie `NULL`-Indikatoren (einen für jeden Parameter) definieren und den PDA `DB2SQL_P` und den PDA `DBINFO_P` (nur bei Angabe von `DBINFO`) einschließen, falls gewünscht. Siehe auch die entsprechende Db2-Literatur von IBM.

3 Entscheiden Sie, welche und wie viele Ergebnismengen (Result Sets) die Stored Procedure an den Aufrufer zurückgeben soll.

4 Codieren Sie Ihre Stored Procedure als Natural-Subprogramm.

- **Rückgabe von Ergebnismengen**
  Um Ergebnismengen (Result Sets) zurückzugeben, codieren Sie ein Natural SQL `SELECT`-Statement mit der Option `WITH RETURN`.

  Um die gesamte Ergebnismenge zurückzugeben, codieren Sie ein `ESCAPE BOTTOM`-Statement unmittelbar nach dem `SELECT`-Statement.

  Um einen Teil der Ergebnismenge zurückzugeben, codieren Sie ein `IF *COUNTER = 1 ESCAPE TOP END-IF` unmittelbar nach dem `SELECT`-Statement. Dadurch wird sichergestellt, dass Sie die erste leere Zeile, die von dem `SELECT WITH RETURN`-Statement zurückgegeben wird, nicht verarbeiten. Um die Zeilenverarbeitung zu stoppen, führen Sie ein `ESCAPE BOTTOM`-Statement aus.

  Wenn Sie die durch `SELECT WITH RETURN` eingeleitete Verarbeitungsschleife nicht über `ESCAPE BOTTOM` verlassen, wird die erzeugte Ergebnismenge geschlossen und nichts an den Aufrufer zurückgegeben.
- **Achtung beim Zugriff auf andere Datenbanken!**
  Sie können innerhalb einer Natural Stored Procedure auf andere Datenbanken (z. B. Adabas) zugreifen. Beachten Sie jedoch, dass Ihr Zugriff auf andere Datenbanken weder mit den Aktualisierungen durch den Aufrufer der Stored Procedure noch mit den Aktualisierungen gegen Db2 innerhalb der Stored Procedure synchronisiert wird.
- **Natural for Db2-Behandlung von `COMMIT`- und `ROLLBACK`-Statemenst**
  Db2 lässt nicht zu, dass eine Stored Procedure Natural SQL `COMMIT`- oder `ROLLBACK`-Statements ausgibt (die Ausführung dieser Statements versetzt den Aufrufer in einen Must-Rollback-Status). Daher behandelt die Natural for Db2-Laufzeit diese Statements wie folgt, wenn sie von einer Stored Procedure ausgegeben werden:

ROLLBACK gegen Db2 wird übersprungen, wenn es von Natural selbst erzeugt wird.

ROLLBACK gegen Db2 wird ausgeführt, wenn es vom Benutzer programmiert wurde. So erhält der Aufrufer nach einem Natural-Fehler die Natural-Fehlerinformation und nicht den unqualifizierten Must-Rollback-Status. Außerdem stellt diese Funktion sicher, dass jede Datenbanktransaktion der Stored Procedure zurückgenommen wird, wenn das Benutzerprogramm die Transaktion zurücknimmt.

5 **Für DB2 UDB:** Geben Sie ein CREATE PROCEDURE Statement aus, das Ihre Stored Procedure definiert, z.B:

```
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>;
```

Die in spitzen Klammern (< >) angegebenen Daten entsprechen den in der obigen Tabelle aufgeführten Daten, PARM1 - PARM3 und FORMAT hängen von der Aufrufparameterliste der Stored Procedure ab. Siehe auch ***Beispiel Stored Procedure NDBPURGN***.

6 Codieren Sie Ihr Natural-Programm, das die Stored Procedure über das Natural-SQL-Statement CALLDBPROC aufruft.

Kodieren Sie die Parameter in dem CALLDBPROC-Statement in der gleichen Reihenfolge, wie sie in der Stored Procedure angegeben sind. Definieren Sie die Parameter im aufrufenden Programm in einem Format, das mit dem in der Stored Procedure definierten Format kompatibel ist.

Wenn Sie Ergebnismengen (Result Sets) verwenden, geben Sie eine RESULT SETS-Klausel in dem CALLDBPROC-Statement an, gefolgt von einer Anzahl von Ergebnismengen-Locator-Variablen in FORMAT (I4). Die Anzahl der Ergebnismengen-Locator-Variablen sollte mit der Anzahl der von der Stored Procedure erstellten Ergebnismengen übereinstimmen. Wenn Sie weniger angeben als erstellt werden, gehen einige Ergebnismengen verloren. Wenn Sie mehr angeben, als erstellt werden, gehen die restlichen Ergebnismengen-Locator-Variablen verloren. Die Reihenfolge der Locator-Variablen entspricht der Reihenfolge, in der die Ergebnismengen von der Stored Procedure erstellt werden.

Beachten Sie, dass die Felder, in die die Zeilen der Ergebnismenge gelesen werden, den Feldern entsprechen müssen, die in dem SELECT WITH RETURN-Statement verwendet werden, das die Ergebnismenge erzeugt hat.

Schreiben einer Natural UDF

Dieser Abschnitt enthält einen allgemeinen Leitfaden zum Schreiben einer benutzerdefinierten Funktion (Natural UDF) und zu den dabei zu beachtenden Aspekten.

Siehe auch den Abschnitt *Schreiben einer Natural Stored Procedure*.

Um eine Natural UDF zu schreiben:

1 Bestimmen Sie das Format und die Attribute der Parameter, die zwischen dem Aufrufer und der Stored Procedure übergeben werden.

2 Erstellen Sie einen Natural-Parameterdatenbereich (PDA).

3 Fügen Sie die Parameterdefinitionen der Natural UDF an, indem Sie `NULL`-Indikatoren (einen für jeden Parameter) definieren und den PDA `DB2SQL_P` inkludieren.

4 Kodieren Sie, falls erforderlich, einen `SCRATCHPAD`-Bereich in der Parameterliste.

5 Kodieren Sie, falls erforderlich, einen Call-Type-Parameter. Wenn Sie `DBINFO` angegeben haben, inkludieren Sie den PDA `DBINFO_P`. Siehe auch die entsprechende Db2-Literatur von IBM.

6 Kodieren Sie Ihre UDF als Natural-Subprogramm und beachten Sie Folgendes:

- **Achtung beim Zugriff auf andere Datenbanken!**
  Sie können innerhalb einer Natural UDF auf andere Datenbanken (z. B. Adabas) zugreifen. Beachten Sie jedoch, dass Ihr Zugriff auf andere Datenbanken weder mit den Aktualisierungen durch den Aufrufer der Stored Procedure noch mit den Aktualisierungen gegen Db2 innerhalb der Stored Procedure synchronisiert wird.
- **Behandlung von`COMMIT`- und `ROLLBACK`-Statements durch Natural für Db2**
  Db2 lässt nicht zu, dass eine Stored Procedure `COMMIT`- oder `ROLLBACK`-Statements ausgibt. Die Ausführung dieser Statements führt zu einem Must-Rollback-Status. Wenn eine Natural-Stored-Procedure ein `COMMIT` oder `ROLLBACK` ausgibt, verarbeitet die Natural for Db2-Laufzeitumgebung diese Anweisungen wie folgt:

  `COMMIT` gegen Db2 wird übersprungen. Dadurch kann die Stored Procedure Adabas-Updates festschreiben (Commit), ohne in einen Must-Rollback-Status von Db2 einzutreten.

  `ROLLBACK` gegen Db2 wird übersprungen, wenn es implizit von der Natural-Laufzeit ausgegeben wird.

  ROLLBACK gegen Db2 wird ausgeführt, wenn es vom Benutzer programmiert wurde. So erhält der Aufrufer nach einem Natural-Fehler einen entsprechenden Natural-Fehlermeldungstext, gerät aber nicht in einen unqualifizierten Must-Rollback-Zustand. Außerdem stellt diese Reaktion sicher, dass jede Datenbanktransaktion, die die Stored Procedure

ausführt, zurückgenommen wird, wenn das Benutzerprogramm die Transaktion zurücknimmt (Backout).

7 Geben Sie eine `CREATE FUNCTION`-Statement aus, das z.B. Ihre UDF definiert:

```
CREATE FUNCTION <FUNCTION>
  ([PARM1]    <FORMAT>,
   [PARM2]    <FORMAT>,
   [PARM3]    <FORMAT>

  )
  RETURNS <FORMAT>

  EXTERNAL NAME <LOADMOD>
  LANGUAGE ASSEMBLE
  PROGRAM TYPE <PGM TYPE>
  PARAMETER STYLE DB2SQL
.
.
.;
```

Im obigen Beispiel sind die variablen Daten in spitze Klammern (`< >`) eingeschlossen und beziehen sich auf die Schlüsselwörter, die den Klammern vorausgehen. Geben Sie einen gültigen Wert an, zum Beispiel:

`LOADMOD` bezeichnet das Natural for Db2 Server Stub-Modul, z. B. `NDB`*`vr`*`SRV`, wobei *`vr`* für die Natural-Versionsnummer steht.

`PARM1` - `PARM3` und `FORMAT` beziehen sich auf die Aufrufparameterliste der UDF. Siehe auch ***Beispiel für Natural User Defined Function***.

8 Codieren Sie ein Natural-Programm mit SQL-Statements, die die UDF aufrufen.

Geben Sie die Parameter des Natural-UDF-Aufrufs in der gleichen Reihenfolge an, wie sie in der Natural-UDF-Definition angegeben sind. Das Format der Parameter im aufrufenden Programm muss mit dem in der Natural UDF definierten Format kompatibel sein.

Beispiel einer Stored Procedure

In diesem Abschnitt wird die Beispiel-Stored-Procedure `NDBPURGN` beschrieben, ein Natural-Subprogramm, das Natural-Objekte aus dem vom Natural Stored-Procedures-Server verwendeten Buffer Pool löscht.

Die folgenden Themen werden in diesem Abschnitt behandelt:

- Objekte von NDBPURGN

Objekte von NDBPURGN

Die Beispiel-Stored-Procedure NDBPURGN besteht aus den folgenden Textobjekten (Members), die in der Natural System Library SYSDB2 gespeichert sind:

Objekt	Erläuterung
CR6PURGN	Eingabe-Member (Textobjekt) für SYSDB2 ISQL. Enthält SQL-Statements, die zur Deklaration von NDBPURGN in Db2 verwendet werden.
NDBPURGP	Das Client-(Natural-)Programm, das ■ den Namen des zu löschenden Programms und die Library, in der es sich befindet, abfragt, ■ die Stored Procedure NDBPURGN aufruft ■ und das Ergebnis der Anfrage meldet. Stellen Sie sicher, dass das spezielle Db2-Register PATH den Schema-Namen der von CR6PURGN erstellten Stored Procedure NDBPURGN enthält. Dies kann mit dem SQL-Statement SET PATH *schema-name* erreicht werden. Fehlt der Schema-Name, kann der SQLCODE-440 zurückgegeben werden.
NDBPURGN	Die Stored Procedure, die Objekte aus dem Buffer Pool löscht. NDBPURGN ruft die Anwendungsprogrammierschnittstelle USR0340N auf, die in der Natural System Library SYSEXT bereitgestellt wird.. Deshalb muss USR0340N in der Library verfügbar sein, die als Steplib für die Ausführungsumgebung definiert ist.

Definieren der Stored Procedure NDBPURGN

Um die Beispiel-Stored-Procedure NDBPURGN zu definieren:

1 Definieren Sie die Stored Procedure im Db2-Katalog mit Hilfe der SQL-Statements, die als Textobjekte CR5PURGN (für Db2 Version 5) und CR6PURGN (für Db2 Version 6) bereitgestellt werden.

2 Geben Sie den Namen des Natural Stored Procedure Stub (hier: NDB*vr*SRV, wobei *vr* für die Natural-Versionsnummer steht) als LOADMOD (V5) oder EXTERNAL NAME (V6) an. Der Natural Stored Procedure Stub wird während der Installation durch Assemblieren des Makros NDBSTUB erzeugt.

3 Übergeben Sie als ersten Parameter den internen Natural-Parameter STCB an die Stored Procedure. Der STCB-Parameter ist ein VARCHAR-Feld, das Informationen enthält, die zum Aufrufen der Stored Procedure in Natural erforderlich sind:

   ■ Der Programmname der Stored Procedure und die Bibliothek, in der sie sich befindet,

- die Beschreibung der an die Stored Procedure übergebenen Parameter und
- die von Natural erzeugte Fehlermeldung, wenn die Stored Procedure während der Ausführung fehlschlägt.

Der STCB-Parameter wird automatisch durch die CALLMODE=NATURAL-Klausel des Natural-SQL-Statement CALLDBPROC generiert und wird aus den Parametern entfernt, die der Server-Stub an die Natural Stored Procedure übergibt. Somit ist STCB für den Aufrufer und die Stored Procedure unsichtbar. Wenn jedoch ein Nicht-Natural-Client eine Natural Stored Procedure aufrufen will, muss der Client den STCB-Parameter explizit übergeben. Siehe auch *Stored Procedure Control Block* weiter unten.

Stored Procedure Control Block (STCB)

Nachfolgend sehen Sie den durch die CALLMODE=NATURAL-Klausel erzeugten Stored Procedure Control Block (STBC), wie er von der Stored Procedure NDBPURGN *vor* und *nach* der Ausführung erzeugt wurde. Geänderte Werte sind durch Fettdruck hervorgehoben:

STCB vor der Ausführung:

```
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 nach der Ausführung:

```
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
```

Beispiel einer Natural User Defined Function

Dieser Abschnitt beschreibt die User Defined Function (UDF) `NAT.DEM2UDFN`, ein Natural-Subprogramm zur Berechnung des Produkts zweier Zahlen.

Die Beispiel-UDF `NAT.DEM2UDF` besteht aus den folgenden Objekten, die in der Natural-System-Library SYSDB2 ausgeliefert werden:

Objekt	Erläuterung
`DEM2CUDF`	Enthält SQL-Statements, die zur Erstellung von `DEM2UDFN` verwendet werden (siehe unten).
`DEM2UDFP`	Das Client-(Natural-)Programm, das ■ Zeilen aus der UDF-Tabelle `NAT.DEMO` holt, ■ die `NAT.DEM2UDFN` (siehe unten) in der `WHERE`-Klausel aufruft und ■ die abgerufenen Zeilen anzeigt.
`DEM2UDFN`	Die UDF, die das Produkt von zwei Zahlen bildet. `DEM2UDFN` muss in die Natural Library NAT in der Natural Systemdatei FUSER in der Ausführungsumgebung kopiert werden.

18 Interface-Subprogramme

Es stehen mehrere Natural- und Nicht-Natural-Subprogramme zur Verfügung, die Ihnen interne Informationen aus Natural for Db2 liefern oder spezifische Funktionen bereitstellen, für die es kein entsprechendes Natural-Statement gibt.

In diesem Kapitel werden die folgenden Themen behandelt:

Natural-Subprogramme

Die folgenden Natural-Subprogramme sind verfügbar:

Subprogramm	Funktion
NDBCONV	Setzt den Conversational Mode 2 und setzt ihn zurück.
NDBDBRM	Prüft, ob ein Natural-Programm SQL-Zugriffe enthält und ob es für die statische Ausführung modifiziert wurde.
NDBDBR2	Prüft, ob ein Natural-Programm SQL-Zugriffe enthält und ob es für die statische Ausführung modifiziert wurde.
NDBDBR3	Prüft, ob ein Natural-Programm SQL-Zugriffe enthält, ob es für die statische Ausführung modifiziert wurde und ob es als statisch generiert werden kann.
NDBERR	Liefert Diagnoseinformationen über den zuletzt ausgeführten SQL-Aufruf.
NDBISQL	Führt SQL-Statements im dynamischen Modus aus.
NDBISQLD	Führt SQL-Statements im dynamischen Modus unter Verwendung dynamischer Variablen aus.
NDBNOERR	Unterdrückt die normale Natural-Fehlerbehandlung.
NDBNROW	Ermittelt die Anzahl der Zeilen, die von einem Natural SQL-Statement betroffen sind.
NDBSTMP	Stellt eine Db2-TIMESTAMP-Spalte als alphanumerisches Feld zur Verfügung und umgekehrt.

Die oben genannten Subprogramme sind alle in der Natural Library SYSDB2 und in der Natural Library SYSTEM in der Systemdatei FNAT enthalten.

Darüber hinaus enthält die Natural Library SYSTEM in der Systemdatei FNAT das Subprogramm DBTLIB2N und die Subroutine DBDL219S. Sie werden von NDBDBRM und NDBDBR2 verwendet. Die entsprechenden Parameter müssen in einem DEFINE DATA-Statement definiert werden.

Die Natural-Subprogramme NDBDBRM, NDBDBR2 und NDBDBR3 gestatten die optionale Angabe der Datenbankkennung, der Dateinummer, des Passworts und des Chiffriercodes der Library-Datei, die das zu untersuchende Programm enthält.

Werden diese Parameter nicht angegeben, wird entweder die aktuelle FNAT-Datei oder die FUSER-Datei verwendet, um das zu untersuchende Programm zu finden, je nachdem, ob der Library-Name mit „SYS“ beginnt oder nicht.

Programme, die NDBDBRM, NDBDBR2 oder NDBDBR3 ohne diese Parameter aufrufen, funktionieren ebenfalls wie vor dieser Änderung, da die hinzugefügten Parameter als optional deklariert sind.

Ausführliche Informationen zu diesen Subprogrammen finden Sie unter den in der obigen Tabelle enthaltenen Links und in der Beschreibung des Aufrufformats und der Parameter in dem mit dem Subprogramm gelieferten Textobjekt (*subprogram-name*T).

Aufrufen von Subprogrammen aus einem Natural-Programm heraus

- Natural-Subprogramme werden mit dem Natural-Statement CALLNAT aufgerufen.
- Nicht-Natural-Subprogramme werden mit dem Natural-Statement CALL aufgerufen.

Subprogramm NDBCONV

Das Natural-Subprogramm NDBCONV wird verwendet, um den Conversational Mode 2 in CICS-Umgebungen entweder zu setzen oder zurückzusetzen. Der Conversational Mode 2 bedeutet, dass Aktualisierungstransaktionen über Terminal-E/A erzeugt werden, bis entweder ein COMMIT oder ROLLBACK ausgegeben wurde (Achtung: Db2- und CICS-Ressourcen werden über Terminal-E/A gehalten!). Das bedeutet, dass der Conversational Mode 2 die gleiche Wirkung hat wie der Natural-Profilparameter PSEUDO=OFF, mit dem Unterschied, dass der Conversational Mode nach einem Db2-Aktualisierungs-Statement (UPDATE, DELETE, INSERT) aufgenommen und nach einem COMMIT oder ROLLBACK wieder verlassen wird, während PSEUDO=OFF den Conversational Mode für die gesamte Natural-Sitzung bewirkt.

Ein Beispielprogramm namens CALLCONV wird in der Library SYSDB2 zur Verfügung gestellt. Es demonstriert, wie man NDBCONV aufruft. Eine Beschreibung des Aufrufformats und der Parameter finden Sie in dem Textobjekt NDBCONVT.

Das aufrufende Natural-Programm muss die folgende Syntax verwenden:

```
CALLNAT 'NDBCONV' #CONVERS #RESPONSE
```

Die verschiedenen Parameter sind in der folgenden Tabelle beschrieben:

Parameter	Format/Länge	Erläuterung
#CONVERS	I1	Enthält den gewünschten Conversational Mode (Eingabe)
#RESPONSE	I4	Enthält die Rückmeldung von NDBCONV (Ausgabe)

Der Parameter #CONVERS kann die folgenden Werte enthalten:

Code	Erläuterung
0	Der Conversational Mode 2 muss zurückgesetzt werden.
1	Der Conversational Mode 2 muss gesetzt werden.

Der Parameter #RESPONSE kann die folgenden Werte enthalten:

Code	Erläuterung
0	Der Conversational Mode 2 wurde erfolgreich gesetzt oder zurückgesetzt.
-1	Der angegebene Wert von #CONVERS ist ungültig, der Conversational Mode wurde nicht geändert.
-2	NDBCONV wird in einer Umgebung aufgerufen, die keine CICS-Umgebung ist und in der der Conversational Mode 2 nicht unterstützt wird.

Subprogramm NDBDBRM

Das Natural-Subprogramm NDBDBRM wird verwendet, um zu prüfen, ob ein Natural-Programm SQL-Zugriffe enthält und ob es für die statische Ausführung modifiziert wurde. Es wird außerdem verwendet, um den entsprechenden DBRM-Namen (Datenbankanforderungsmodul) aus dem Header eines als statisch generierten Natural-Programms zu ermitteln (siehe auch **Programme für die statische Ausführung vorbereiten** vorbereiten).

Das Installationsmedium enthält ein Beispielprogramm namens CALLDBRM, das den Aufruf von NDBDBRM demonstriert. Eine Beschreibung des Aufrufformats und der Parameter finden Sie in dem Textobjekt NDBDBRMT.

Das aufrufende Natural-Programm muss die folgende Syntax verwenden:

```
CALLNAT 'NDBDBRM' #LIB #MEM #DBRM #RESP #DBID #FILENR #PASSWORD #CIPHER
```

Die verschiedenen Parameter sind in der folgenden Tabelle beschrieben:

Parameter	Format/Länge	Erläuterung
#LIB	A8	Enthält den Namen der Library des zu prüfenden Programms.
#MEM	A8	Enthält den Namen des zu prüfenden Programms (Member).
#DBRM	A8	Gibt den DBRM-Namen zurück.
#RESP	I2	Gibt einen Rückmeldecode zurück. Die möglichen Codes sind unten aufgeführt.
#DBID	N5	Optional. Datenbankkennung der Library-Datei.
#FILENR	N5	Optional. Dateinummer der Library-Datei.
#PASSWORD	A8	Optional. Passwort der Library-Datei.
#CIPHER	N8	Optional. Chiffrierschlüssel der Library-Datei.

Der Parameter #RESP kann die folgenden Werte enthalten:

Code	Erläuterung
0	Das Member #MEM in der Library #LIB hat SQL-Zugriff; es ist statisch, wenn #DBRM einen Wert enthält.
-1	Das Member #MEM in der Library #LIB hat keinen SQL-Zugriff.
-2	Das Member #MEM in der Bibliothek #LIB existiert nicht.
-3	Es wurde kein Library-Name angegeben.
-4	wurde kein Member-Name angegeben.
-5	Der Library-Name muss mit einem Buchstaben beginnen.
>-5	Weitere negative Rückmeldecodes entsprechen den Fehlernummern der Natural-Fehlermeldungen.
>0	Positive Rückmeldecodes entsprechen Fehlernummern von Natural Security-Meldungen.

Subprogramm NDBDBR2

Das Natural-Subprogramm NDBDBR2 dient dazu, zu prüfen, ob ein Natural-Programm SQL-Zugriffe enthält und ob es für die statische Ausführung modifiziert wurde. Außerdem ermittelt es aus dem Header eines statisch generierten Natural-Programms den entsprechenden DBRM-Namen (Datenbankanforderungsmodul) (siehe auch **Programme für die statische Ausführung vorbereiten**) und den vom Precompiler generierten Zeitstempel.

Das Installationsmedium enthält ein Beispielprogramm namens CALLDBR2, das den Aufruf von NDBDBR2 demonstriert. Eine Beschreibung des Aufrufformats und der Parameter finden Sie im Textobjekt NDBDBR2T.

Das aufrufende Natural-Programm muss die folgende Syntax verwenden:

```
CALLNAT 'NDBDBR2' #LIB #MEM #DBRM #TIMESTAMP #PCUSER #PCRELLEV #ISOLLEVL #DATEFORM ↩
#TIMEFORM #RESP #DBID #FILENR #PASSWORD #CIPHER
```

Die verschiedenen Parameter sind in der folgenden Tabelle beschrieben:

Parameter	Format/Länge	Erläuterung
#LIB	A8	Enthält den Namen der Library des zu prüfenden Programms.
#MEM	A8	Enthält den Namen des zu prüfenden Programms (Member).
#DBRM	A8	Gibt den DBRM-Namen zurück.
#TIMESTAMP	B8	Vom Precompiler erzeugtes Konsistenz-Token.
#PCUSER	A8	Nicht unterstützter Parameter. Wird nur aus Kompatibilitätsgründen beibehalten.
#PCRELLEV	A1	Nicht unterstützter Parameter. Wird nur aus Kompatibilitätsgründen beibehalten.
#ISOLLEVL	A1	Nicht unterstützter Parameter. Wird nur aus Kompatibilitätsgründen beibehalten.

Parameter	Format/Länge	Erläuterung
#DATEFORM	A1	Nicht unterstützter Parameter. Wird nur aus Kompatibilitätsgründen beibehalten.
#TIMEFORM	A1	Nicht unterstützter Parameter. Wird nur aus Kompatibilitätsgründen beibehalten.
#RESP	I2	Gibt einen Rückmeldecode zurück. Die möglichen Codes sind unten aufgeführt.
#DBID	N5	Optional. Datenbankkennung der Library-Datei.
#FILENR	N5	Optional. Dateinummer der Library-Datei.
#PASSWORD	A8	Optional. Passwort der Library-Datei.
#CIPHER	N8	Optional. Chiffriercode der Library-Datei.

Der Parameter #RESP kann die folgenden Werte enthalten:

Code	Erläuterung
0	Das Member #MEM in der Library #LIB hat SQL-Zugriff. Es ist statisch, wenn #DBRM einen Wert enthält.
-1	Das Member #MEM in der Library #LIB hat keinen SQL-Zugriff.
-2	Das Member #MEM in der Library #LIB existiert nicht.
-3	Es wurde kein Library-Name angegeben.
-4	Es wurde kein Member-Name angegeben.
-5	Der Library-Name muss mit einem Buchstaben beginnen.
>-5	Weitere negative Rückmeldeecodes entsprechen den Fehlernummern der Natural-Fehlermeldungen.
>0	Positive Rückmeldeecodes entsprechen Fehlernummern von Natural Security-Meldungen.

Subprogramm NDBDBR3

Mit dem Natural-Subprogramm NDBDBR3 kann geprüft werden, ob ein Natural-Programm SQL-Zugriffe enthält (#RESP 0), ob das Natural-Programm ausschließlich SQL-Anweisungen enthält, die dynamisch ausführbar sind (#RESP 0, #DBRM '*DYNAMIC') und ob es für die statische Ausführung modifiziert wurde (#RESP 0, #DBRM *dbrmname*). Es wird außerdem verwendet, um den entsprechenden DBRM-Namen (Datenbankanforderungsmodul) aus dem Header eines als statisch generierten Natural-Programms (siehe auch Vorbereitung von Programmen für die statische Ausführung) und den vom Precompiler generierten Zeitstempel zu erhalten.

Auf dem Installationsmedium befindet sich ein Beispielprogramm namens CALLDBR3, das den Aufruf von NDBDBR3 demonstriert. Eine Beschreibung des Aufrufformats und der Parameter finden Sie im Textobjekt NDBDBR3T.

Das aufrufende Natural-Programm muss die folgende Syntax verwenden:

```
CALLNAT 'NDBDBR3' #LIB #MEM #DBRM #TIMESTAMP #PCUSER #PCRELLEV #ISOLLEVL #DATEFORM ↩
#TIMEFORM #RESP #DBID #FILENR #PASSWORD #CIPHER
```

Die verschiedenen Parameter sind in der folgenden Tabelle beschrieben:

Parameter	Format/Länge	Erläuterung
#LIB	A8	Enthält den Namen der Library des zu prüfenden Programms.
#MEM	A8	Enthält den Namen des zu prüfenden Programms (Member).
#DBRM	A8	Gibt den DBRM-Namen zurück. ■ Leerschritt (Space), wenn das Programm SQL-Zugriff hat, ■ *DYNAMIC, wenn das Programm nur dynamisch ausführbares SQL enthält, ■ DBRM-Name, wenn das Programm statisch generiert wurde.
#TIMESTAMP	B8	Vom Precompiler erzeugtes Konsistenz-Token.
#PCUSER	A8	Nicht unterstützter Parameter. Wird nur aus Kompatibilitätsgründen beibehalten.
#PCRELLEV	A1	Nicht unterstützter Parameter. Wird nur aus Kompatibilitätsgründen beibehalten.
#ISOLLEVL	A1	Nicht unterstützter Parameter. Wird nur aus Kompatibilitätsgründen beibehalten.
#DATEFORM	A1	Nicht unterstützter Parameter. Wird nur aus Kompatibilitätsgründen beibehalten.
#TIMEFORM	A1	Nicht unterstützter Parameter. Wird nur aus Kompatibilitätsgründen beibehalten.
#RESP	I2	Gibt einen Rückgabecode zurück. Die möglichen Codes sind unten aufgeführt.
#DBID	N5	Optional. Datenbankkennung der Library-Datei.
#FILENR	N5	Optional. Dateinummer der Library-Datei.
#PASSWORD	A8	Optional. Passwort der Library-Datei.
#CIPHER	N8	Optional. Chiffrierschlüssel der Library-Datei.

Der Parameter #RESP kann die folgenden Werte enthalten:

Code	Erläuterung
0	Das Member #MEM in der Library #LIB hat SQL-Zugriff. Es ist statisch, wenn #DBRM einen anderen Wert als Leerschritt (Space) und *DYNAMIC enthält.
-1	Das Member #MEM in der Library #LIB hat keinen SQL-Zugriff.
-2	Das Member #MEM in der Library #LIB existiert nicht.
-3	Es wurde kein Library-Name angegeben.
-4	Es wurde kein Member-Name angegeben.
-5	Der Library-Name muss mit einem Buchstaben beginnen.

Code	Erläuterung
>-5	Weitere negative Rückmeldecodes entsprechen den Fehlernummern der Natural-Fehlermeldungen.
>0	Positive Rückmeldecodes entsprechen Fehlernummern von Natural Security-Meldungen.

Subprogramm NDBERR

Das Natural-Subprogramm NDBERR ersetzt die Funktion E der DB2SERV-Schnittstelle, die zwar noch vorhanden, aber nicht mehr dokumentiert ist. Es liefert Diagnoseinformationen über den letzten SQL-Aufruf. Es gibt auch den Datenbanktyp zurück, bei dem der Fehler aufgetreten ist. NDBERR wird typischerweise aufgerufen, wenn ein Datenbankaufruf einen SQLCODE ungleich Null zurückgibt (was einen Fehler NAT3700 bedeutet).

Ein Beispielprogramm namens CALLERR wird auf dem Installationsmedium mitgeliefert. Es demonstriert, wie NDBERR aufgerufen werden kann. Eine Beschreibung des Aufrufformats und der Parameter finden Sie im Textobjekt NDBERRT.

Das aufrufende Natural-Programm muss die folgende Syntax verwenden:

```
CALLNAT 'NDBERR' #SQLCODE #SQLSTATE #SQLCA #DBTYPE
```

Die verschiedenen Parameter sind in der folgenden Tabelle beschrieben:

Parameter	Format/Länge	Erläuterung
#SQLCODE	I4	Gibt den SQL-Rückgabecode zurück.
#SQLSTATE	A5	Gibt einen Rückgabecode für die Ausgabe des zuletzt ausgeführten SQL-Statements zurück.
#SQLCA	A136	Gibt den SQL-Kommunikationsbereich des letzten Db2-Zugriffs zurück.
#DBTYPE	B1	Gibt die Kennung (im Hexadezimalformat) für die aktuell verwendete Datenbank zurück (wobei X'02' für Db2 steht).

Subprogramm NDBISQL

Das Natural-Subprogramm NDBISQL dient dazu, SQL-Statements im dynamischen Modus auszuführen. Das SELECT-Statement und alle SQL-Statements, die dynamisch vorbereitet werden können (gemäß der Db2-Literatur von IBM), können an NDBISQL übergeben werden.

Auf dem Installationsmedium befindet sich ein Beispielprogramm namens CALLISQL, das den Aufruf von NDBISQL demonstriert. Eine Beschreibung des Aufrufformats und der Parameter ist im Textobjekt NDBISQLT enthalten.

Das aufrufende Natural-Programm muss die folgende Syntax verwenden:

```
CALLNAT 'NDBISQL'#FUNCTION #TEXT-LEN #TEXT (*) #SQLCA #RESPONSE #WORK-LEN #WORK (*)
```

Die verschiedenen Parameter sind in der folgenden Tabelle beschrieben:

Parameter	Format/Länge	Erläuterung	
#FUNCTION	A8	Gültige Funktionen, siehe unten.	
#TEXT-LEN	I2	Länge des SQL-Statements oder des Puffers für den Rückgabebereich.	
#TEXT	A1(1:V)	Enthält das SQL-Statement (EXECUTE) oder empfängt eine Datenzeile (FETCH).	
#SQLCA	A136	Enthält die SQLCA.	
#RESPONSE	I4	Gibt einen Rückmeldecode zurück.	
#WORK-LEN	I2	Länge des durch #WORK angegebenen Arbeitsbereiches (optional).	
#WORK	A1(1:V)	Arbeitsbereich zur Aufnahme von SQLDA/SQLVAR und Hilfsfeldern bei Aufrufen (optional).	
#DBTYPE	I2	Datenbanktyp (optional).	
		0	Voreinstellung
		2	DB2
		4	CNX

Gültige Funktionen für den Parameter #FUNCTION sind:

Function	Parameter	Erläuterung
CLOSE		Schließt den Cursor für das SELECT-Statement.
EXECUTE	#TEXT-LEN #TEXT (*)	Führt das SQL-Statement aus. Enthält die Länge des Statements. Enthält das SQL-Statement. Die ersten beiden Zeichen müssen leer sein.
FETCH	#TEXT-LEN #TEXT (*)	Gibt einen Datensatz aus dem SELECT-Statement zurück. Größe von #TEXT (in Bytes). Puffer für die Kopfzeile.
TITLE	#TEXT-LEN #TEXT (*)	Gibt die Kopfzeile für das SELECT-Statement zurück. Größe von #TEXT (in Bytes), erhält die Länge der Kopfzeile (= Länge des Datensatzes). Puffer für die Kopfzeile.

Der Parameter #RESPONSE kann die folgenden Rückmeldecodes enthalten:

Code	Funktion	Erläuterung
5	EXECUTE	Das Statement ist ein SELECT-Statement.
6	TITLE, FETCH	Die Daten werden abgeschnitten; nur beim ersten Aufruf von TITLE oder FETCH gesetzt.
100	FETCH	Kein Datensatz / Ende der Daten.
-2		Nicht unterstützter Datentyp (z.B. GRAPHIC).
-3	TITLE, FETCH	Kein Cursor geöffnet; wahrscheinlich ungültige Aufrufsequenz oder anderes Statement als SELECT.
-4		Zu viele Spalten in der Ergebnistabelle.
-5		SQLCODE vom Aufruf.
-6		Keine Versionsübereinstimmung.
-7		Ungültige Funktion.
-8		Fehler beim SQL-Aufruf.
-9		Arbeitsbereich ungültig (möglicherweise Relokation).
-10		Schnittstelle nicht verfügbar.
-11	EXECUTE	Die ersten beiden Bytes des Statements sind nicht leer.

Aufruf-Reihenfolge

Der erste Aufruf muss ein EXECUTE-Aufruf sein. NDBISQL hat einen festen SQLDA AREA, der Platz für 50 Spalten bietet. Wenn dieser Bereich für ein bestimmtes SELECT zu klein ist, ist es möglich, einen optionalen Arbeitsbereich bei den Aufrufen von NDBISQL durch Angabe von #WORK-LEN (I2) und #WORK(A1/1:V) bereitzustellen.

Dieser Arbeitsbereich wird verwendet, um die SQLDA und temporäre Arbeitsfelder wie Null-Indikatoren und Hilfsfelder für numerische Spalten aufzunehmen. Kalkulieren Sie 16 Bytes für den SQLDA-Kopf und 44 Bytes für jede Ergebnisspalte sowie 2 Bytes Null-Indikator für jede Spalte und Platz für jede numerische Spalte, wenn Sie #WORK-LEN und #WORK(*) bei NDBISQL-Aufrufen angeben. Wenn diese optionalen Parameter bei einem EXECUTE-Aufruf angegeben werden, müssen sie auch bei jedem folgenden Aufruf angegeben werden.

Handelt es sich bei der Anweisung um ein SELECT-Statement (d.h. es wird der Rückmeldecode 5 zurückgegeben), kann eine beliebige Folge von TITLE- und FETCH-Aufrufen verwendet werden, um die Daten abzurufen. Ein Rückmeldecode von 100 zeigt das Ende der Daten an.

Der Cursor muss mit einem CLOSE-Aufruf geschlossen werden.

Der Funktionscode EXECUTE schließt implizit einen Cursor, der durch einen vorherigen EXECUTE-Aufruf für ein SELECT-Statement geöffnet wurde.

In TP-Umgebungen kann zwischen einem EXECUTE-Aufruf und einem TITLE-, FETCH- oder CLOSE-Aufruf, der sich auf dasselbe Statement bezieht, keine Terminal-E/A durchgeführt werden.

Subprogramm NDBISQLD

Das Natural-Subprogramm NDBISQLD dient dazu, SQL-Statements im dynamischen Modus auszuführen. Das SELECT-Statement und alle SQL-Statements, die dynamisch vorbereitet werden können (gemäß der Db2-Literatur von IBM), können an NDBISQLD übergeben werden.

Ein Beispielprogramm namens CALISQLD ist im Lieferumfang des Installationsmediums enthalten. Es demonstriert, wie man NDBISQLD aufruft. Eine Beschreibung des Aufrufformats und der Parameter ist im Textobjekt ISQLDT enthalten.

Das aufrufende Natural-Programm muss die folgende Syntax verwenden:

```
CALLNAT 'NDBISQLD'#FUNCTION #TEXT #SQLCA #RESPONSE #WORK #DBTYPE
```

Die verschiedenen Parameter sind in der folgenden Tabelle beschrieben:

Parameter	Format/Länge	Erläuterung	
#FUNCTION	A8	Gültige Funktionen, siehe unten.	
#TEXT	A DYNAMIC	Enthält das SQL-Statement (EXECUTE) oder empfängt die Datenzeile (FETCH).	
#SQLCA	A136	Enthält die SQLCA.	
#RESPONSE	I4	Gibt einen Rückmeldecode zurück.	
#WORK	A DYNAMIC	Arbeitsbereich, um SQLDA/SQLVAR und Hilfsfelder über Aufrufe hinweg zu halten (optional). Wenn angegeben, muss #WORK groß genug sein, um alle Hilfsfelder (SQLDA) für die SQL-Anfrage zu speichern.	
#DBTYPE	I2	Datenbanktyp (optional).	
		0	Voreinstellung
		2	DB2
		4	CNX

Gültige Funktionen für den Parameter #FUNCTION sind:

Funktion	Parameter	Erläuterung
CLOSE	-	Schließt den Cursor für das SELECT-Statement.
EXECUTE	#TEXT	Schließt den Cursor für das SELECT-Statement. Enthält das SQL-Statement. Die ersten vier Zeichen müssen leer sein.
FETCH	#TEXT	Gibt eine Zeile aus dem SELECT-Statement zurück. #TEXT muss groß genug sein, um die Zeile der durch das SELECT-Statement erzeugten Ergebnismenge (Result Set) aufzunehmen.

Funktion	Parameter	Erläuterung
		After FETCH, the *LENGTH(#TEXT) is reduced to the exact size of the row. Nach FETCH wird die *LENGTH(#TEXT) auf die genaue Größe der Zeile reduziert.
TITLE	#TEXT	Gibt die Header-Literale für das SELECT-Statement zurück. #TEXT muss groß genug sein, um die Zeile der durch das SELECT-Statement erzeugten Ergebnismenge (result set) aufzunehmen.

Der Parameter #RESPONSE kann die folgenden Rückmeldecodes enthalten:

Code	Funktion	Erläuterung
5	EXECUTE	Das Statement ist ein SELECT-Statement.
6	TITLE, FETCH	Die Daten werden abgeschnitten; nur beim ersten Aufruf von TITLE oder FETCH gesetzt.
100	FETCH	Kein Datensatz/Ende der Daten.
-2	-	Nicht unterstützter Datentyp (z.B. GRAPHIC).
-3	TITLE, FETCH	Kein Cursor geöffnet. Wahrscheinlich ungültige Aufrufsequenz oder anderes Statement als SELECT.
-4	-	Zu viele Spalten in der Ergebnistabelle.
-5	-	SQLCODE aus Aufruf.
-6	-	Keine Versionsübereinstimmung.
-7	-	Ungültige Funktion.
-8	-	Fehler vom SQL-Aufruf.
-9	-	Arbeitsbereich ungültig (möglicherweise Relokation).
-10	-	Schnittstelle nicht verfügbar.
-11	EXECUTE	Die ersten beiden Bytes des Statements sind nicht leer.

Aufruf-Reihenfolge

Der erste Aufruf muss ein EXECUTE-Aufruf sein. NDBISQLD hat eine feste SQLDA AREA, die Platz für 50 Spalten bietet. Wenn dieser Bereich für ein bestimmtes SELECT zu klein ist, ist es möglich, bei den Aufrufen von NDBISQLD mit #WORK(A)DYNAMIC einen optionalen Arbeitsbereich anzugeben.

Dieser Arbeitsbereich wird verwendet, um die SQLDA und temporäre Arbeitsfelder wie Null-Indikatoren und Hilfsfelder für numerische Spalten aufzunehmen. Kalkulieren Sie 16 Bytes für den SQLDA-Header und 44 Bytes für jede Ergebnisspalte sowie 2 Bytes Null-Indikator für jede Spalte und Platz für jede numerische Spalte, wenn Sie #WORK(A)DYNAMIC bei NDBISQLD-Aufrufen angeben. Wenn diese optionalen Parameter bei einem EXECUTE-Aufruf angegeben werden, müssen sie auch bei jedem folgenden Aufruf angegeben werden.

Handelt es sich bei dem Statement um ein SELECT-Statement (d.h. es wird der Rückmeldecode 5 zurückgegeben), kann eine beliebige Folge von TITLE- und FETCH-Aufrufen verwendet werden, um die Daten abzurufen. Ein Rückmeldecode von 100 zeigt das Ende der Daten an.

Der Cursor muss mit einem CLOSE-Aufruf geschlossen werden.

Der Funktionscode EXECUTE schließt implizit einen Cursor, der durch einen vorherigen EXECUTE-Aufruf für ein SELECT-Statement geöffnet wurde.

In TP-Umgebungen kann zwischen einem EXECUTE-Aufruf und einem TITLE-, FETCH- oder CLOSE-Aufruf, der sich auf dasselbe Statement bezieht, keine Terminal-E/A durchgeführt werden.

Subprogramm NDBNOERR

Das Natural-Subprogramm NDBNOERR dient zur Unterdrückung von Natural-NAT3700-Fehlern, die durch den nächsten SQL-Aufruf verursacht werden. Dies ermöglicht eine kontrollierte Fortsetzung des Programms, wenn ein SQL-Statement einen SQLCODE ungleich Null erzeugt. Nachdem der SQL-Aufruf ausgeführt wurde, wird NDBERR zur Untersuchung des SQLCODE verwendet.

Ein Beispielprogramm namens CALLNOER wird auf dem Installationsmedium mitgeliefert; es demonstriert, wie NDBNOERR aufgerufen werden kann. Eine Beschreibung des Aufrufformats und der Parameter finden Sie im Textobjekt NDBNOERT.

Das aufrufende Natural-Programm muss die folgende Syntax verwenden:

```
CALLNAT 'NDBNOERR'
```

Für dieses Subprogramm sind keine Parameter vorgesehen.

Anmerkung: Es werden nur NAT3700-Fehler (d.h. SQL-Rückmeldecodes ungleich Null) unterdrückt, und außerdem nur Fehler, die durch den nächstfolgenden SQL-Aufruf verursacht werden.

Einschränkungen bei Datenbankschleifen

- Wenn NDBNOERR vor einem Statement aufgerufen wird, das eine Datenbankschleife einleitet, und ein Initialisierungsfehler auftritt, wird keine Verarbeitungsschleife eingeleitet, es sei denn, es wurde eine IF NO RECORDS FOUND-Klausel angegeben.
- Wenn NDBNOERR innerhalb einer Datenbankschleife aufgerufen wird, gilt dies nicht für die Verarbeitungsschleife selbst, sondern nur für das SQL-Statement, das anschließend innerhalb dieser Schleife ausgeführt wird.

Subprogramm NDBNROW

Das Natural-Subprogramm NDBNROW wird verwendet, um die Anzahl der von den Natural-SQL-Statements Searched UPDATE, Searched DELETE und INSERT betroffenen Zeilen zu ermitteln.

Die Anzahl der betroffenen Zeilen wird aus dem SQL-Kommunikationsbereich (SQLCA) gelesen. Ein positiver Wert steht für die Anzahl der betroffenen Zeilen, während ein Wert von minus eins (-1) anzeigt, dass alle Zeilen einer Tabelle in einem segmentierten Tablespace gelöscht wurden, siehe auch die Natural-Systemvariable *NUMBER in der Natural-*Systemvariablen*-Dokumentation.

Auf dem Installationsmedium befindet sich ein Beispielprogramm namens CALLNROW, das den Aufruf von NDBNROW demonstriert. Eine Beschreibung des Aufrufformats und der Parameter finden Sie in dem Textobjekt NDBNROWT.

Das aufrufende Natural-Programm muss die folgende Syntax verwenden:

```
CALLNAT 'NDBNROW' #NUMBER
```

Der Parameter #NUMBER (I4) enthält die Anzahl der betroffenen Zeilen.

Subprogramm NDBSTMP

Für Db2 bietet Natural eine TIMESTAMP-Spalte als alphanumerisches Feld (A26) im Format *YYYY-MM-DD-HH.MM.SS.MMMMMM*.

Da Natural noch keine Berechnungen mit solchen Feldern unterstützt, wird das Natural-Subprogramm NDBSTMP bereitgestellt, um diese Art von Funktionalität zu ermöglichen. Es konvertiert Natural-Zeitvariablen in Db2-Zeitstempel und umgekehrt und führt arithmetische Berechnungen mit Db2-Zeitstempeln durch.

Ein Beispielprogramm namens CALLSTMP wird auf dem Installationsmedium mitgeliefert; es demonstriert, wie NDBSTMP aufgerufen werden kann. Eine Beschreibung des Aufrufformats und der Parameter ist im Textobjekt NDBSTMPT enthalten.

Die verfügbaren Funktionen sind:

Code	Erläuterung
ADD	Addiert Zeiteinheiten (Labeled Durations, s. Anmerkung weiter unten) zu einem gegebenen Db2-Zeitstempel und gibt eine Natural-Zeitvariable und einen neuen Db2-Zeitstempel zurück.
CNT2	Konvertiert eine Natural-Zeitvariable (Format T) in einen Db2-Zeitstempel (Spaltentyp TIMESTAMP) und Labeled Durations.
C2TN	Konvertiert einen Db2-Zeitstempel (Spaltentyp TIMESTAMP) in eine Natural-Zeitvariable (Format T) und Labeled Durations.
DIFF	Ermittelt die Differenz zwischen zwei gegebenen Db2-Zeitstempeln und gibt Labeled Durations zurück.
GEN	Generiert einen Db2-Zeitstempel aus den aktuellen Datums- und Zeitwerten der Natural-Systemvariablen *TIMX und gibt einen neuen Db2-Zeitstempel zurück.
SUB	Subtrahiert Labeled Durations von einem gegebenen Db2-Zeitstempel und gibt eine Natural-Zeitvariable und einen neuen Db2-Zeitstempel zurück.
TEST	Prüft einen gegebenen Db2-Zeitstempel auf gültiges Format und gibt TRUE oder FALSE zurück.

Anmerkung: Labeled Durations sind Einheiten von Jahr, Monat, Tag, Stunde, Minute, Sekunde und Mikrosekunde.

DB2SERV-Schnittstelle

DB2SERV ist ein Assembler-Programm-Einstiegspunkt, der aus einem Natural-Programm heraus aufgerufen werden kann.

DB2SERV führt eine der folgenden Funktionen aus:

- **Funktion D**, die das SQL-Statement EXECUTE IMMEDIATE ausführt.
- **Funktion P**, die ein Assembler-Modul namens NDBPLAN aufruft.

Die Parameter- oder Variablenwerte, die von jeder dieser Funktionen zurückgegeben werden, werden auf ihr Format, ihre Länge und ihre Anzahl überprüft.

Funktion D

Die Funktion D führt das SQL-Statement EXECUTE IMMEDIATE aus. Damit können SQL-Statements innerhalb eines Natural-Programms ausgegeben werden.

Der SQL-Statement-String, der auf das EXECUTE IMMEDIATE-Statement folgt, muss der Natural-Programmvariablen STMT zugewiesen werden. Er muss gültige SQL-Statements enthalten, die mit dem EXECUTE IMMEDIATE-Statement zulässig sind, wie in der entsprechenden IBM-Literatur beschrieben. Beispiele finden Sie unten und in den Demonstrationsprogrammen DEM2* in der Natural System Library SYSDB2.

Anmerkung: Die Bedingungen, die für die Ausgabe der Natural-Statements `END TRANSACTION` oder `BACKOUT TRANSACTION` gelten, gelten auch für die Ausgabe der SQL-Statements `COMMIT` oder `ROLLBACK`.

Kommandosyntax

```
CALL 'DB2SERV' 'D' STMT STMTL SQLCA RETCODE
```

Die in diesem Kommando verwendeten Variablen sind in der folgenden Tabelle beschrieben:

Variable	Format/Länge	Erläuterung
`STMT`	A*nnn*	Enthält einen Kommando-String, der aus der oben beschriebenen SQL-Syntax besteht.
`STMTL`	I2	Enthält die Länge des in `STMT` definierten Strings.
`SQLCA`	A136	Gibt den aktuellen Inhalt des SQL-Kommunikationsbereichs zurück.
`RETCODE`	I2	Gibt einen Schnittstellen-Rückgabecode zurück. Die folgenden Codes sind möglich: 0 Keine Warnung bzw. kein Fehler aufgetreten. 4 SQL-Statement verursachte eine SQL-Warnung. 8 SQL-Statement führte zu einem SQL-Fehler. 12 Interner Fehler aufgetreten; die entsprechende Natural-Fehlernummer kann mit dem Kommando `SQLERR` angezeigt werden.

Der aktuelle Inhalt des SQLCA und ein Schnittstellen-Rückgabecode (`RETCODE`) werden zurückgegeben. Der SQLCA ist eine Sammlung von Variablen, die von Db2 verwendet werden, um einem Anwendungsprogramm Informationen über die Ausführung seiner SQL-Statements zu liefern.

Die folgenden Beispiele zeigen, wie Sie `DB2SERV` mit der Funktion `D` verwenden können.

Beispiel für Funktion 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
************************************************************************
```

Anmerkung: Die Funktionalität der `DB2SERV`-Funktion `D` wird auch mit dem `PROCESS SQL`-Statement bereitgestellt.

Beispiel für Funktion 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
************************************************************************
```

Bei Verwendung von `SET CURRENT SQLID` kann der Ersteller-Name (Creator Name) einer Tabelle durch die aktuelle SQLID ersetzt werden. Damit ist es möglich, auf identische Tabellen mit demselben Tabellennamen, aber mit unterschiedlichen Ersteller-Namen zuzugreifen. Tabellennamen dürfen also nicht durch einen Ersteller-Namen qualifiziert werden, wenn dieser durch die SQLID ersetzt werden soll.

In allen unterstützten TP-Monitor-Umgebungen kann die `SQLID` dann über Terminal-Ein-/Ausgaben hinweg beibehalten werden, bis entweder die Sitzung beendet oder sie über `DB2SERV` zurückgesetzt wird.

Funktion P

Die Funktion `P` ruft ein Assembler-Modul namens `NDBPLAN` auf, das zum Aufbau und/oder Abbau der Db2-Verbindung unter TSO und im Batch-Modus verwendet wird. Damit kann eine Natural-Anwendung eine Planumschaltung unter TSO und im Batch-Modus durchführen.

Das Programm `DEM2PLAN` ist ein Anwendungsbeispiel für den Einsatz von `DB2SERV` mit der Funktion `P`.

Der Name des aktuellen Db2-Subsystems (`#SSM`) und der Name des neuen Anwendungsplans (`#PLAN`) müssen angegeben werden. Außerdem werden ein Schnittstellen-Rückgabecode (`#RETCODE`) und der Db2-Ursachencode (`#REASON`) zurückgegeben.

Kommando-Syntax

```
CALL 'DB2SERV' 'P' #SSM #PLAN #RETCODE #REASON
```

Variable	Format/Länge	Erläuterung
#SSM	A4	Enthält den Namen des aktuellen Db2-Subsystems.
#PLAN	A8	Enthält den Namen des neuen Plans.
#RETCODE		Gibt einen Schnittstellen-Rückgabecode zurück. Die folgenden Codes sind möglich: 0 Keine Warnung bzw. kein Fehler aufgetreten. 12 Der angegebene neue Anwendungsplan ist nicht eingeplant. 99 Die aktuelle Umgebung ist keine Call Attachment Facility-Umgebung (CAF). *nnn* Rückgabecode aus der CAF-Schnittstelle (siehe auch die entsprechende Db2-Literatur von IBM).
#REASON	I4	Gibt den Ursachencode der CAF-Schnittstelle zurück (siehe auch die entsprechende Db2-Literatur von IBM).

Beispiel für Funktion 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
************************************************************************
```

Wichtig: Die Planumschaltung unter TSO und im Batch-Modus ist nur mit der CAF-Schnittstelle möglich; siehe auch den Abschnitt *Planumschaltung unter TSO und im Batch-Modus*.

19 Natural File Server für DB2

In allen unterstützten TP-Monitor-Umgebungen (CICS, IMS TM und TSO) stellt Natural for Db2 eine Zwischenarbeitsdatei bereit, die als der File Server bezeichnet wird, um zu verhindern, dass die Ergebnisse der Datenbankauswahl bei jeder Terminal-E/A verloren gehen. Ausnahme: Complete.

File Server-Konzept

Um zu vermeiden, dass das verwendete Auswahl-Statement erneut ausgegeben und die Cursor neu positioniert werden müssen, schreibt Natural die Ergebnisse einer Datenbankauswahl in eine Zwischendatei. Die gespeicherten ausgewählten Zeilen, die möglicherweise später benötigt werden, werden dann von Natural so verwaltet, als ob die Möglichkeiten der Dialogverarbeitung vorhanden wären. Dies wird dadurch erreicht, dass die Zwischendatei für nachfolgende Bildschirme automatisch durchgeblättert wird, wobei die Position in der Arbeitsdatei und nicht in Db2 beibehalten wird.

Alle Zeilen aller geöffneten Cursor werden vor der ersten Terminal-Eingabe-/Ausgabe-Operation zum File Server ausgelagert. Anschließend werden alle Daten aus dieser Datei abgerufen, wenn sich Natural auf einen der zuvor ausgelagerten Cursor bezieht (siehe die Beschreibung des Auslagerns im Abschnitt *Logischer Aufbau des File Servers* weiter unten).

Wenn eine Zeile aktualisiert oder gelöscht werden soll, wird zunächst geprüft, ob sie in der Zwischenzeit durch einen anderen Vorgang aktualisiert worden ist. Dazu wird die Zeile erneut ausgewählt und aus der Db2-Datenbank geholt und dann mit der ursprünglichen Version, die vom File Server abgerufen wurde, verglichen. Wenn die Zeile noch unverändert ist, kann der Aktualisierungs- oder Löschvorgang ausgeführt werden. Wenn nicht, wird eine entsprechende Fehlermeldung zurückgegeben. Die beim Aktualisieren oder Löschen einer Zeile erforderliche Neuauswahl ist sowohl im dynamischen als auch im statischen Modus möglich.

Nur die Felder, die im File Server gespeichert sind, werden auf Konsistenz mit dem aus Db2 abgerufenen Datensatz geprüft.

Da die Zeile eindeutig identifiziert werden muss, muss die Natural-View ein Feld enthalten, für das eine eindeutige Zeile erstellt wurde. Dieses Feld muss in Db2 als eindeutiger Schlüssel definiert sein. In einem Natural-Datendefinitionsmodul (DDM) wird es dann als eindeutiger Schlüssel über den entsprechenden Natural-spezifischen Kurznamen angezeigt.

Vorbereitungen für die Verwendung des File Servers

Die Größe einer Zeile, die auf den File-Server geschrieben werden kann, ist auf 32 KB oder 32767 Byte begrenzt. Wenn eine Zeile größer ist, wird eine entsprechende Fehlermeldung ausgegeben.

Der File Server kann entweder eine VSAM RRDS-Datei oder den Software AG Editor Buffer Pool als Speichermedium verwenden, um ausgewählte Zeilen von Db2-Tabellen zu speichern.

In diesem Abschnitt werden die folgenden Themen behandelt:

- File Server – VSAM
- File Server – Editor Buffer Pool
- File Server – Shared-Memory-Objekt

File Server – VSAM

Die Installation des File-Servers erfolgt über einen Batch Job, der die Zwischendatei definiert und formatiert. Beispiele für diesen Batch Job werden auf dem Installationsmedium mitgeliefert, wie im entsprechenden Abschnitt beschrieben.

Definition der Größe des File Servers

Der File Server wird durch die Definition einer RRDS-VSAM-Datei mit Hilfe der Access Method Services (AMS) erstellt. Seine physische Größe und sein Name müssen angegeben werden.

Formatierung des File-Servers

Die Formatierung des File Servers erfolgt durch einen Batch Job, der fünf vom Benutzer angegebene Eingabeparameter benötigt und den File Server entsprechend diesen Parametern formatiert. Die Parameter geben an:

1. Die Anzahl der zu formatierenden Blöcke (logische Größe der VSAM-Datei). Dieser Wert wird aus dem ersten Parameter des Unterkommandos `RECORD` des AMS-Kommandos `DEFINE CLUSTER` übernommen.
2. Die Anzahl der Benutzer, die sich gleichzeitig bei Natural anmelden können.
3. Die Anzahl der formatierten Blöcke, die als primäre Zuordnung pro Benutzer definiert werden.
4. Die Anzahl der formatierten Blöcke, die pro Benutzer als sekundäre Zuordnung verwendet werden sollen.
5. Die maximale Anzahl der File Server-Blöcke, die jedem Benutzer zugewiesen werden. Wird diese Zahl überschritten, wird eine entsprechende Natural-Fehlermeldung zurückgegeben.

Unmittelbar vor dem ersten Zugriff auf den File Server wird der Natural-Sitzung ein File Server-Verzeichniseintrag zugeordnet und der Natural-Sitzung wird die als Primärzuordnung angegebene Anzahl von Blöcken zugeordnet.

Die primäre Zuordnung wird als Zwischenspeicher für das Ergebnis einer Datenbankauswahl verwendet und sollte groß genug sein, um alle Zeilen einer üblichen Datenbankauswahl aufnehmen zu können. Sollte für eine große Datenbankauswahl mehr Platz auf dem File Server benötigt werden, weisen die File Server-Module einen sekundären Bereich zu, der der Menge entspricht, die beim Formatieren des File Servers für die sekundäre Zuordnung angegeben wurde.

Ein sekundärer Bereich wird also nur dann zugewiesen, wenn die aktuelle primäre Zuordnung nicht ausreicht, um alle Daten zu aufzunehmen, die in die Zwischendatei geschrieben werden müssen. Die Anzahl der zulässigen sekundären Zuordnungen hängt von der maximalen Anzahl der Blöcke ab, die Sie zuordnen dürfen. Dieser Parameter wird auch bei der Formatierung des File-Servers angegeben.

Die als Sekundärzuordnung definierte Anzahl von Blöcken wird so lange wiederholt zugeordnet, bis entweder alle ausgewählten Daten in die Datei geschrieben wurden oder die maximal zulässige Anzahl von Blöcken überschritten wird. Ist dies der Fall, wird eine entsprechende Natural-Fehlermeldung ausgegeben. Wenn die als sekundäre Zuordnung erhaltenen Blöcke nicht mehr benötigt werden (d. h. wenn die mit dieser Zuordnung verbundene Natural-Schleife geschlossen ist), werden sie in den Pool freier Blöcke des File Servers zurückgegeben.

Die primäre Zuordnung von Blöcken ist Ihnen jedoch immer zugeteilt, und zwar bis zum Ende Ihrer Natural-Sitzung.

Erforderliche Änderungen für einen Multiple-Volume File Server

Um Kanalkonflikte oder Engpässe zu minimieren, die durch die Platzierung eines großen und stark genutzten File Servers auf einem einzigen DASD-Volume verursacht werden können, können Sie einen File Server anlegen, der sich über mehrere DASD-Volumes erstreckt.

Um einen solchen File Server zu erstellen und zu formatieren, sind zwei Änderungen in dem Job erforderlich, der zur Definition des VSAM-Clusters verwendet wird:

1. Ändern Sie `VOLUME ()` in `VOLUMES (`*`vol1`*`,`*`vol2`*`,...)`.
2. Teilen Sie die Gesamtzahl der für die Datei erforderlichen Datensätze (wie im ersten Job-Formatierungsparameter angegeben) durch die Anzahl der oben angegebenen Volumes. Das Ergebnis der Berechnung wird für den Parameter `RECORDS` des `DEFINE CLUSTER`-Kommandos verwendet.

Das bedeutet, dass im File Server-Formatierungsjob der Wert des ersten Parameters das Ergebnis der Multiplikation zweier Parameter aus dem Kommando `DEFINE CLUSTER` ist: `RECORDS` und `VOLUMES`.

File Server – Editor Buffer Pool

Der Buffer Pool des Software AG-Editors wird als Speichermedium verwendet, wenn im NTDB2-Makro EBPFSRV=ON eingestellt ist. In diesem Fall werden die primären, sekundären und maximalen Zuordnungsmengen für den File Server durch die Subparameter EBPPRAL, EBPSEC und EBPMAX des NTDB2-Makros festgelegt. Bevor Natural for Db2 zum ersten Mal versucht, Daten aus einer Natural-Benutzersitzung auf den File Server zu schreiben, wird eine logische Datei des Software AG Editor Buffer Pool mit der Natural-Terminalkennung als Benutzername und der Nummer 2240 als Sitzungsnummer zugewiesen.

Der Betrieb des File Servers hängt in diesem Fall von der Definition des Software AG Editor Buffer Pool ab. Siehe *Editor Buffer Pool* in der Natural *Operations*-Dokumentation.

Die Anzahl der logischen Dateien für den Buffer Pool begrenzt die Anzahl der Benutzer, die gleichzeitig auf den File Server zugreifen. Die Anzahl der Arbeitsdateiblöcke begrenzt die Menge der Daten, die zu einem bestimmten Zeitpunkt gespeichert werden. (Sie müssen auch berücksichtigen, dass es außer Natural for Db2 noch andere Benutzer des Software AG Editors gibt).

Die Verwendung des Software AG Editor Buffer Pool als Speichermedium für den File Server ermöglicht es Natural for Db2 jedoch, in einer Sysplex-Umgebung zu laufen.

Wenn Sie den File Server in einer Sysplex-Umgebung einsetzen möchten, empfiehlt es sich, den Buffer Pool des Software AG Editors als Speichermedium zu verwenden.

File Server – Shared-Memory-Objekt

Als Speichermedium für den File Server wird ein z/OS Shared-Memory-Objekt oberhalb der Speichergrenze verwendet. In diesem Fall hat das Shared-Memory-Objekt die gleiche Struktur wie die im vorigen Abschnitt erwähnte VSAM-RRDS-Datei, aber alle Daten werden im virtuellen Speicher oberhalb der Grenze verwaltet. Ein File Server, der ein Shared-Memory-Objekt verwendet, wird als Shared Memory Objects File Server (FSSM) bezeichnet.

Um einen Shared Memory Objects File Server zu verwenden:

1. Definieren Sie das Shared-Memory-Objekt in der Parameterdatei ASMPARM parameter file (siehe Natural-*Operations*-Dokumentation) eines Natural Authorized Services Manager, der das Modul NATFSSM enthält.

   Die Meldungen, die vom Authorized Services Manager zurückgegeben werden können, finden Sie unter *Messages from the Shared Memory Objects File Server under NDB* in der *Messages and Codes*-Dokumentation.
2. Setzen Sie SMFSRV=ON als Schlüsselwort-Subparameter des Profilparameters DB2 oder des NTDB2-Makros.
3. Geben Sie den Namen des zu verwendenden Shared-Memory-Objekts mit dem Schlüsselwort-Subparameter DDFSERV des Profilparameters DB2 (bzw. des Makros NTDB2) an.

4 Starten Sie den in Schritt 1 konfigurierten Authorized Services Manager.

Wenn Natural for zIIP in Ihrer Natural-Umgebung aktiviert ist, ist ein Wechsel zwischen den Modi SRB und TCB für den Zugriff auf den File Server nicht erforderlich, da keine Ein-/Ausgaben auf der Platte durchgeführt werden müssen.

Definition von Größe und Format eines FSSM

Die Größe und das Format des FSSM werden im Dataset `ASMPARM` des Authorized Services Manager definiert, und zwar ähnlich wie der VSAM File Server definiert wird. Jede Definition hat das folgende Format:

```
FSSMxxxx=(name,number-of-blocks,number-of-users,primary-blocks,secondary-blocks,maximum-blocks,block-size)
```

Erklärungen:

`FSSM`*xxxx*	Das für die Definition eines File Servers erforderliche Initialisierungsschlüsselwort. `FSSM` ist ein Pflicht-Präfix, das von 1 bis 4 Zeichen *xxxx* gefolgt werden kann.
name	Der logische Name für den File-Server. Der Name kann bis zu 8 Zeichen enthalten und muss mit dem Namen übereinstimmen, der mit dem Schlüsselwort-Subparameter `DDFSERV` des Profilparameters `DB2` (bzw. des Makros `NTDB2`) in der Natural-Sitzung angegeben wurde.
number-of-blocks	Die Anzahl der vom File-Server verwendeten Blöcke. Gültiger Bereich: 3 - 2147483647. Die Zahl muss ein Vielfaches von 8 sein.
number-of-users	Die Anzahl der Benutzer, die den File-Server gleichzeitig benutzen können. Gültiger Bereich: 1 - 32767
primary-blocks	Die primäre Blockzuordnung für jeden Benutzer. Gültiger Bereich: 1 - 32767
secondary-blocks	Die sekundäre Blockzuordnung für jeden Benutzer. Gültiger Bereich: 1 - 32767
maximum-blocks	Die maximale Blockzuordnung für jeden Benutzer. Gültiger Bereich: 1 - 32767
block-size	Die Größe der einzelnen Blöcke auf dem File Server. Gültiger Bereich: 1 - 32767

Beispiele:

```
FSSMPRM1=(CMFSERV,1000,500,50,10,100,31744)

FSSMPRM2=(CMFSERV2,1000,203,50,10,200,4080)
```

Formatierung eines FSSM

Die FSSM wird implizit formatiert, wenn der erste Benutzer eine Natural-Sitzung mit `SMFSRV=ON` mit dem Namen des in `DDFSERV` angegebenen Shared-Memory-Objekts startet.

Wenn ein anderer Benutzer aus einem anderen Adressraum das gleiche Shared-Memory-Objekt verwendet, stellt Natural for Db2 implizit den Zugriff auf das Shared-Memory-Objekt für diesen Adressraum her. Der Zugriff auf das Shared-Memory-Objekt für einen Adressraum scheitert erst, wenn der Adressraum beendigt ist.

Unmittelbar vor dem ersten Zugriff auf den File Server wird der Natural-Sitzung ein File Server-Verzeichniseintrag zugeordnet und die als primäre Zuordnung angegebene Anzahl von Blöcken zugeordnet.

Die primäre Zuordnung wird als Zwischenspeicher für das Ergebnis einer Datenbankauswahl verwendet und sollte groß genug sein, um alle Zeilen einer üblichen Datenbankauswahl zu enthalten. Benötigt der File Server mehr Speicherplatz, führen die Module des File Servers eine sekundäre Zuordnung durch, wobei die Anzahl der Blöcke verwendet wird, die bei der Formatierung des File-Servers für die sekundäre Zuordnung angegeben wurde.

Ein sekundärer Bereich wird also nur dann zugeordnet, wenn Ihre aktuelle primäre Zuordnung nicht ausreicht, um alle Daten aufzunehmen, die in die Zwischendatei geschrieben werden müssen. Die Anzahl der Blöcke, die für die sekundären Zuordnungen zulässig sind, hängt von der maximalen Anzahl der Blöcke ab, die Sie zuordnen dürfen.

Die für die sekundäre Zuordnung festgelegte Anzahl an Blöcken wird so lange wiederholt, bis entweder alle ausgewählten Daten in die Datei geschrieben wurden oder bis die maximale Anzahl an Blöcken, die Sie zuordnen dürfen, überschritten wird. Ist dies der Fall, wird eine entsprechende Natural-Fehlermeldung ausgegeben. Wenn die als sekundäre Zuordnung erhaltenen Blöcke nicht mehr benötigt werden (d. h. wenn die mit dieser Zuordnung verbundene Natural-Schleife endet), werden sie in den Pool freier Blöcke des File-Servers zurückgegeben. Die primäre Zuordnung von Blöcken ist Ihnen jedoch immer bis zum Ende Ihrer Natural-Sitzung zugewiesen.

Logischer Aufbau des File Servers

Unmittelbar bevor eine Natural-Benutzersitzung auf den File Server zugreift, wird der Natural-Benutzersitzung ein File Server-Verzeichniseintrag (VSAM) oder eine logische Datei (Software AG Editor Buffer Pool) zugeordnet und die als primäre Zuordnung angegebene Anzahl an Blöcken wird bis zum Ende der Sitzung reserviert.

Generell wird der File-Server nur verwendet, wenn eine Terminal-Ein-/Ausgabe innerhalb einer aktiven `READ`-, `FIND`- oder `SELECT`-Schleife erfolgt, bei der die Ergebnisse der Datenbankselektion verloren gehen würden. Vor jeder Terminal-E/A-Operation prüft Natural, ob offene Cursor vorhanden sind. Für jeden gefundenen nicht scrollbaren Cursor werden alle verbleibenden Zeilen aus Db2 abgerufen und in eine Zwischendatei geschrieben. In der Dokumentation von Natural for Db2 wird dieser Vorgang als Cursor-Rollout (Auslagerung) bezeichnet.

Für jede Auslagerung (Rollout) eines Cursors (scrollbar und nicht scrollbar) wird eine logische Datei geöffnet, die alle von diesem Cursor abgerufenen Zeilen aufnimmt. Der Platz für die Zwischendatei wird innerhalb des Ihrer Sitzung zugeordneten Speicherplatzes verwaltet. Die logische Datei wird dann auf der Zeile positioniert, die zum Zeitpunkt der Terminal-Eingabe/Ausgabe `CURRENT OF CURSOR` war.

Nachfolgende Datenanforderungen werden dann durch direktes Lesen der Zeilen aus der Zwischendatei erfüllt. Die Datenbank ist nicht mehr beteiligt, und Db2 wird nur noch für Aktualisierungs-, Lösch- oder Speicheroperationen benutzt.

Positioned `UPDATE`- und/oder Positioned `DELETE`-Statements gegen ausgelagerte scrollbare Cursor werden gegen die Db2-Basistabelle und gegen die logische Datei auf dem File Server ausgeführt.

Sobald die entsprechende Verarbeitungsschleife in der Anwendung geschlossen ist, wird die Datei nicht mehr benötigt und die von ihr belegten Blöcke werden in Ihren Pool freier Blöcke zurückgegeben. Von hier aus werden die Blöcke in den Pool freier Blöcke des File Servers zurückgeführt, so dass Ihnen nur noch Ihre primäre Zuordnung zur Verfügung steht.

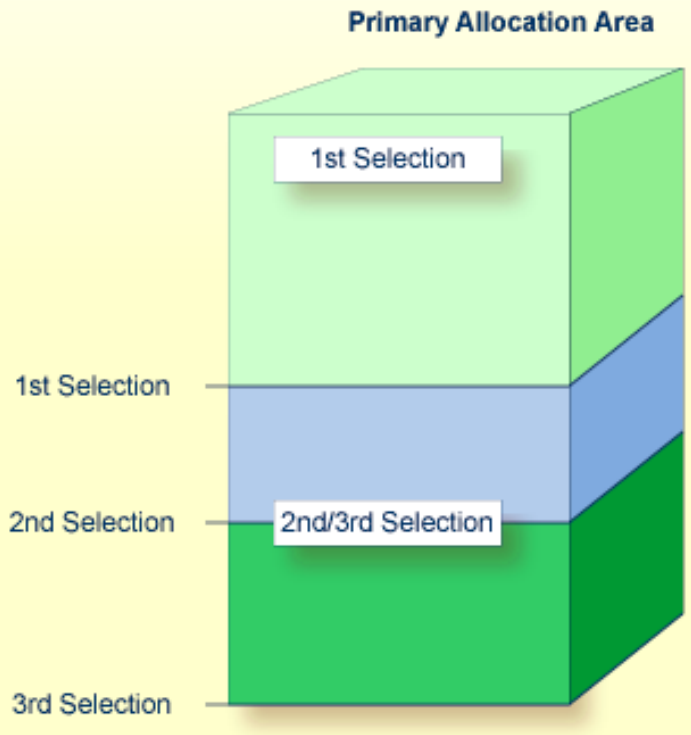
Im folgenden Beispiel wird der für die erste Auswahl zugeordnete Speicherplatz erst dann freigegeben, wenn alle bei der dritten Auswahl ausgewählten Zeilen abgerufen worden sind. Dasselbe gilt für den der dritten Auswahl zugeordneten Speicherplatz.

Der der zweiten Auswahl zugeordnete Speicherplatz kann also für die Auswahlergebnisse der dritten Auswahl verwendet werden.

Der der zweiten Auswahl zugeordnete Speicherplatz kann also für die Auswahlergebnisse der dritten Auswahl verwendet werden.

Example:

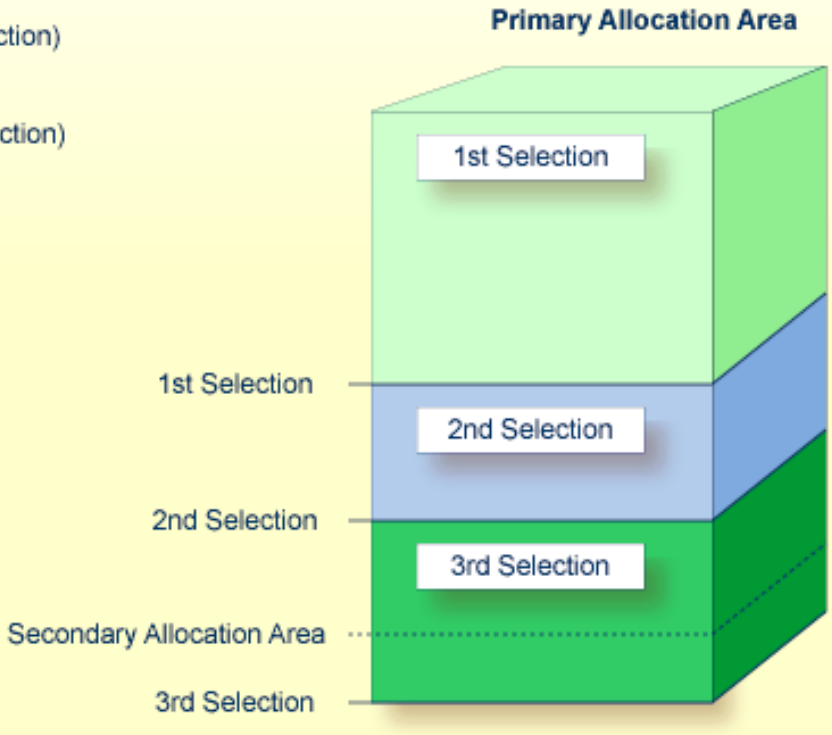
```
:
:
FIND ... (1st selection)
  :
  :
  FIND ... (2nd selection)
    :
    INPUT ...
    :
  END-FIND
  :
  :
  FIND ... (3rd selection)
    :
    INPUT ...
    :
  END-FIND
  :
  :
END-FIND
:
:
```



Wenn der primäre Zuordnungsbereich nicht groß genug ist, z.B. wenn die dritte Auswahl in der zweiten Auswahl verschachtelt ist, wird der sekundäre Zuordnungsbereich verwendet.

Example:

```
:
:
FIND ... (1st selection)
  :
  :
  FIND ... (2nd selection)
    :
    :
    FIND ... (3rd selection)
      :
      INPUT ...
      :
    END-FIND
    :
    :
  END-FIND
  :
  :
END-FIND
:
:
```



Wenn eine Sitzung beendet wird, werden alle Blöcke eines Benutzers in den Pool freier Blöcke zurückgegeben. Wenn eine Sitzung abnormal beendet wird, prüft Natural, sofern möglich, ob ein File Server-Verzeichniseintrag für den entsprechenden Benutzer existiert. Ist dies der Fall, werden alle von diesem Benutzer gehaltenen Ressourcen freigegeben.

Ist Natural nicht in der Lage, die Ressourcen einer abnormal beendeten Benutzersitzung freizugeben, werden diese Ressourcen erst dann freigegeben, wenn sich dieselbe Benutzerkennung von demselben logischen Terminal aus erneut anmeldet.

Wenn dieselbe Benutzerkennung und/oder dasselbe logische Terminal nicht mehr für Natural verwendet werden, bleiben der vorhandene Verzeichniseintrag und der zugewiesene Speicherplatz erhalten, bis der File-Server erneut formatiert wird. Ein erneuter Durchlauf des Formatierungs-Jobs löscht alle vorhandenen Daten und legt das Verzeichnis neu an.

II Natural for VSAM

Diese Dokumentation beschreibt die verschiedenen Aspekte von Natural beim Einsatz in einer VSAM-Umgebung.

Allgemeine Informationen	Besondere Überlegungen zu den von Natural for VSAM unterstützten Umgebungen, zu bekannten Inkompatibilitäten und Einschränkungen beim Einsatz von Natural for VSAM, zu den in dieser Dokumentation verwendeten Begriffen und zu Fehlermeldungen im Zusammenhang mit Natural for VSAM.
Einführung in Natural for VSAM	Komponenten von Natural for VSAM, Struktur der Natural-Schnittstelle zu VSAM.
Anpassung von VSAM	Beschreibung der Parameter, Makros und Eingabe-/Ausgabe-Module von Natural for VSAM.
Betrieb	Informationen zu betrieblichen Aspekten wie dem Aufruf von Natural for VSAM, der `OPEN/CLOSE`-Verarbeitung, dem Natural-Dateizugriff, den Puffern für die Speicherverwaltung und den Anwendungsprogrammierschnittstellen.
Natural-Statements und Natural-Transaktionslogik mit VSAM	Dieses Kapitel beschreibt Besonderheiten, die hinsichtlich der Natural-Statements und Systemvariablen bei der Verwendung mit VSAM zu berücksichtigen sind. Darüber hinaus wird die Natural-Transaktionslogik im Zusammenhang mit VSAM behandelt.

Verwandte Dokumentation

Installationsanweisungen finden Sie unter *Installing Natural for VSAM* in *Installation for z/OS*-Dokumentation.

Zu verschiedenen Aspekten des Zugriffs auf Daten in einer Datenbank mit Natural siehe auch *Database Access* im Natural-*Leitfaden zur Programmierung*.

Eine Liste der Abend-Codes von Natural for VSAM finden Sie unter *Natural for VSAM Abend Codes* (in der Natural-*Messages and Codes*-Dokumentation).

20 Allgemeine Informationen

Verwendungszweck

Mit der Natural-Schnittstelle zu VSAM kann ein Natural-Benutzer auf Daten zugreifen, die in VSAM-Dateien gespeichert sind. Als Voraussetzung muss die aktuelle Version von Natural for Mainframes installiert sein.

Generell gibt es keinen Unterschied zwischen der Verwendung von Natural mit VSAM und der Verwendung von Natural mit Adabas oder einem anderen unterstützten Datenbankmanagementsystem. Die Natural-Schnittstelle zu VSAM ermöglicht Natural-Programmen den Zugriff auf VSAM-Daten mit denselben Natural DML-Statements, die auch für Adabas verfügbar sind. Daher können Programme, die für VSAM geschrieben wurden, auch für den Zugriff auf Adabas-Datenbanken verwendet werden.

Alle Operationen, die eine Interaktion mit VSAM erfordern, werden über die Natural-Schnittstelle zu VSAM ausgeführt.

Umgebungsspezifische Überlegungen

Natural for VSAM ist vollständig ESA- und z/OS Parallel Sysplex-kompatibel. Es läuft im Batch-Modus oder unter den Online-Umgebungen CICS, Com-plete und TSO. Unter CICS läuft es auch im Conversational oder Pseudo-Conversational Mode.

Natural for VSAM unterstützt die folgenden Arten von VSAM-Dateien:

- KSDS
- ESDS
- RRDS
- VRDS

Unter z/OS unterstützt Natural for VSAM die Dataset-Zugriffsmodi Record-Level-Sharing (RLS) und DFSMS Transactional VSAM Services (DFSMStvs).

Die Natural-Systemdateien FNAT, FUSER, FDIC, FSPOOL und FSEC können auch in VSAM-Systemdateien liegen. Für VSAM-Systemdateien verwendet Natural for VSAM die Multi-Fetch-Option, um das Laden von Objekten in den Buffer Pool zu beschleunigen.

Natural for VSAM unterstützt Local Shared Resources (LSR) unter TSO- und in z/OS-Batch-Modi. Für CICS und Com-plete müssen die entsprechenden Dateidefinitionswerkzeuge verwendet werden. Die LSR-Option für VSAM-Dateien verbessert die Geschwindigkeit des wahlfreien Zugriffs.

Natural for VSAM unterstützt den Create/Loading Mode für leere Dateien sowohl unter TSO als auch im Batch-Modus.

Natural for VSAM unterstützt die folgenden Arten von Datentabellen unter CICS z/OS:

- User-Maintained Data Tables (UMT),
- CICS-Maintained Data Tables (CMT),
- Coupling Facility Data Tables (CFDT).

Es unterstützt auch Data Set Name Sharing (DSN) unter TSO und Batch-Modus-Verarbeitung in z/OS, insbesondere für den Zugriff auf Datensätze über einen definierten Pfad.

Natural for VSAM unterstützt Datasets im erweiterten Format für alle Typen der VSAM-Dataset-Organisation. Es gibt jedoch Einschränkungen hinsichtlich ESDS, RRDS and VRDS, die sich aus der Verwendung der Natural-Systemvariablen `*ISN` und deren interner Größenbeschränkung auf 4 Bytes ergeben.

Natural for VSAM mit Natural Security

Da Natural Security die FSEC-Systemdatei als VSAM-Systemdatei unterstützt, müssen die folgenden Einschränkungen beachtet werden:

- Die Generierung von ETIDs ist deaktiviert.
- Die Protokollierung von Verwaltungsaktionen ist deaktiviert.
- Die Passwort-Historienführung ist deaktiviert.
- Die Definition von Utility-Profilen ist deaktiviert.

Integration mit Predict

Predict, das Datendiktionär der Software AG für die Entwicklung mit Sprachen der vierten Generation, insbesondere Natural, ist ein zentrales Repository für Anwendungsmetadaten. Predict bietet Dokumentations- und Cross-Referenz-Funktionen und ermöglicht es Ihnen, automatisch Code aus Definitionen zu generieren und so die Produktivität bei Entwicklung und Wartung zu steigern.

Da Predict VSAM unterstützt, ist ein direkter Zugriff auf VSAM-Dateien über Predict möglich, und Informationen aus VSAM können in das Predict-Datendiktionär übertragen werden, um sie in Datendefinitionen für andere Umgebungen zu integrieren.

Physische und logische VSAM-Datensichten (Views) können eingebunden und verglichen werden, neue VSAM-Views können generiert werden, und Natural-Views können generiert und verglichen

werden. Es werden alle VSAM-spezifischen Datentypen und die referenzielle Integrität von VSAM unterstützt. Einzelheiten finden Sie in der *Predict*-Dokumentation.

In dieser Dokumentation verwendete Begriffe und Akronyme

Begriff	Erläuterung
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	Der in dieser Dokumentation verwendete Begriff „Front-End“ bezieht sich auf den Treiber in Verbindung mit dem Natural-Parameter-Modul.
LSR	Local Shared Resources
NVS	Dies ist der Produktcode von Natural for VSAM. In dieser Dokumentation wird der Produktcode häufig als Präfix in den Namen von Datensätzen, Modulen usw. verwendet.
UMT	User-Maintained Data Tables

Meldungen im Zusammenhang mit VSAM

Die Meldungsnummernbereiche von Natural-Systemmeldungen im Zusammenhang mit VSAM sind 3500-3599.

Eine Liste der Abend-Codes, die von der Natural-Schnittstelle zu VSAM ausgegeben werden können, finden Sie unter *Natural for VSAM Interface Abend Codes* in der Natural-*Messages and Codes*-Dokumentation.

21 Einführung in Natural für VSAM

Dieses Kapitel beschreibt die Bestandteile und die Struktur der Natural-Schnittstelle zu VSAM. Folgende Themen werden behandelt:

Bestandteile von Natural für VSAM

Die Natural-Schnittstelle zu VSAM umfasst die folgenden Bestandteile:

- Das Modul `NVSNUC`, das obligatorisch und umgebungsunabhängig ist und nur als Lademodul geliefert wird.
- Die VSAM-spezifischen Natural-Parameter, die im Natural-Parametermodul definiert sind.
- Das Eingabe-/Ausgabe-Modul, das obligatorisch ist, sich je nach Umgebung unterscheidet und nur als Quellcode geliefert wird.
- Die Module, die für den Betrieb mit VSAM-Systemdateien erforderlich sind. Sie sind optional und werden nur als Lademodule geliefert.
- Die User-Exits.
- Aufrufbare Systemdienste.

Natural für VSAM ist vollständig (E)LPA- oder SVA-kompatibel für mehrere Umgebungen (z. B. CICS, Com-plete und Batch). Das Natural-Parametermodul und das entsprechende E/A-Modul müssen mit dem Front-End-Modul verlinkt werden.

Struktur der Natural-Schnittstelle zu 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).

22 Natural für VSAM anpassen

Die Natural-Parameter in einer VSAM-Umgebung werden an einer einzigen Stelle definiert:

- die Natural-Standardparameter, die im Natural-Parametermodul enthalten sind; siehe *Building a Natural Parameter Module* in der *Operations*-Dokumentation,
- die VSAM-spezifischen Natural-Parameter, die ebenfalls im Natural-Parametermodul enthalten sind; siehe Parametermakro `NTVSAM` in der *Parameter-Referenz*-Dokumentation.

Das Natural-Parametermodul kann editiert werden, damit es den Standards Ihres Standorts entspricht, und dann mit den entsprechenden Jobs assembliert und verlinkt werden (siehe *Installing Natural for VSAM Installation for z/OS*-Dokumentation).

Anpassung des Natural-Parametermoduls

Um Natural in einer VSAM-Umgebung ausführen zu können, müssen Sie den Profilparameter `VSIZE`, das `NTDB`-Makro und das `NTVSAM`-Makro in Ihr Natural-Parametermodul aufnehmen.

Siehe Abschnitt *Installing Natural for VSAM* in der *Installation for z/OS*-Dokumentation).

Für eine Adabas-Systemdatei:

```
VSIZE=72,
NTDB VSAM,vsam-dbid
NTVSAM
```

Für eine VSAM-Systemdatei:

```
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 ist der logische Name (DD oder DLBL) der Systemdatei; siehe auch *Installing Natural for VSAM* in der *Installation for z/OS-* bzw. *Installation for z/VSE*-Dokumentation

Anmerkung: Wenn Sie VSAM-Systemdateien mit Natural ISPF verwenden, lesen Sie auch die *Natural ISPF*-Dokumentation.

Nachfolgend finden Sie Informationen zu:

- Profilparameter VSIZE

- NTDB-Makro
- NTVSAM-Makro

Profilparameter VSIZE

`VSIZE` ist ein Natural-Profilparameter, der auch dynamisch angegeben werden kann. Er gibt die Größe des Natural-Pufferbereichs für VSAM an und definiert den maximalen Speicherplatzbedarf für die internen Tabellen der Natural-Schnittstelle zu VSAM. Die tatsächliche Größe dieser Tabellen hängt von den im Natural-Parametermodul angegebenen Werten ab (siehe *Assemblierung des VSAM-spezifischen Natural-Parametermoduls*).

Mögliche Werte `0`, `1 - 32767` KB.

Wenn Sie die im Natural-Parametermodul angegebenen Standardwerte verwenden, muss der Wert des Parameters `VSIZE` mindestens 72 KB betragen

Wenn `VSIZE` auf `0` gesetzt ist, steht Natural for VSAM nicht zur Verfügung und beim Versuch, auf VSAM-Dateien zuzugreifen, wird eine entsprechende Fehlermeldung ausgegeben. Die Deaktivierung von Natural für VSAM führt zu leichten Leistungsverbesserungen, da die Initialisierung, die Verlagerung und der Aus-/Einlagerungsaufwand der Natural-Schnittstelle zu VSAM entfällt.

NTDB-Makro

Mit dem `NTDB`-Makro werden die Datenbanknummern angegeben, die sich auf VSAM-Dateien beziehen, d.h. die für Natural verfügbaren logischen Zuordnungen.

Die möglichen Werte der Optionen des NTDB-Makros sind in der Natural *Parameter-Referenz*-Dokumentation beschrieben.

Anmerkung: Vergewissern Sie sich, dass die im `NTDB`-Makro für VSAM ausgewählten Datenbankkennungen (DBIDs) nicht mit DBIDs kollidieren, die für andere Datenbankmanagementsysteme ausgewählt wurden.

NTVSAM-Makro

Das `NTVSAM`-Makro wird zur Angabe der VSAM-spezifischen Parameter verwendet.

Der Wertebereich der Schlüsselwort-Subparameter des `NTVSAM`-Makros wird in der Natural *Parameter-Referenz*-Dokumentation beschrieben.

Assemblierung des VSAM-spezifischen Natural-Parametermoduls

Wenn die im Natural-Parametermodul ausgelieferten Standardwerte Ihren Anforderungen nicht genügen, können Sie die Parameterwerte an Ihre Umgebung anpassen. Die einzelnen VSAM-spezifischen Parameter, die im Natural-Parametermodul enthalten sind, werden im folgenden Abschnitt beschrieben.

Das VSAM-spezifische Natural-Parametermodul wird durch Assemblieren des Makros erstellt:

- `NTVSAM`

und optional eines oder mehrere der folgenden Makros:

- `NTVEXIT`
- `NTVLSR`
- `NTVTVSD`

Wenn mehr als ein Makro angegeben wird, muss das `NTVSAM`-Makro zuerst angegeben werden. Weitere Makros nach dem Makro `NTVSAM` können in beliebiger Reihenfolge angegeben werden.

Natural-Eingabe-/Ausgabe-Modul für VSAM

Das Natural E/A-Modul für VSAM hängt von der jeweiligen Umgebung ab, in der Sie arbeiten.

Alle verfügbaren E/A-Module werden im Quellcode geliefert, so dass Sie standortspezifische Änderungen vornehmen und umgebungsspezifische Makros und/oder Precompiler verwenden können. Das E/A-Modul muss mit dem Natural-Parameter-Modul verlinkt werden.

Folgende E/A-Module sind verfügbar:

- NVSCICS-Modul

- NVSMISC-Modul

NVSCICS-Modul

Das NVSCICS-Modul ist für CICS unter z/OS erforderlich. Das Modul enthält die folgenden Parameter:

" bzw. z/VSE" entfällt ab NAT921.

&FCTRELI - Indikator für Reliable Remote FCT Entries

Der Parameter &FCTRELI zeigt an, ob die Schlüssellänge und Satzgröße einer entfernten Datei im FCT-Eintrag der Application Owning Region (AOR) korrekt definiert sind.

Mögliche Werte:

Wert	Erklärung
0	NVSCICS gibt Dummy-Kommandos aus, um das Öffnen der Datei in der File Owning Region (FOR) Region zu erzwingen, und wiederholt dann die Abfrage der tatsächlichen Werte. Dies ist die Standardeinstellung.
1	NVSCICS geht von einem korrekten FCT-Eintrag aus.

Wenn der FCT-Eintrag keine Definition der Schlüssellänge enthält, verwendet NVSCICS die Schlüssellänge des entsprechenden VSAM-DDM.

NVSMISC-Modul

Das NVSMISC-Modul ist in allen Umgebungen mit Ausnahme von CICS erforderlich. Das Modul besteht hauptsächlich aus dem NVMMISC-Makro, mit dem die NVSMISC-E/A-Schnittstelle je nach Betriebssystem und/oder TP-Monitor-Umgebung generiert wird.

NVSMISC wird wie folgt angegeben:

```
name NVMMISC NONRLS=value TIMEOUT=value DSECTS=value DEFER=value COMMIT=value ERROR=value
HFACTOR=value READINT=value SMARTS=value TVS=value
```

Der Name *name* des verschiebbaren Moduls muss 8 Zeichen lang sein.

Der Standardname ist NVSMISCD (nur bei z/VSE).

Die einzelnen Parameter werden im folgenden Abschnitt beschrieben. Geben Sie diese Parameter entsprechend Ihren Erfordernissen an.

NONRLS - Wechsel vom RLS- zum Nicht-RLS-Modus

Nur bei NAT828/912: Dieser Parameter wird unter z/VSE ignoriert.

Wenn Natural for VSAM ein RLS-OPEN für eine RLS-Datei ausgibt und diese Datei in dieser z/OS-Sitzung bereits im Nicht-RLS-Modus geöffnet wurde, gibt dieser Parameter an, ob Natural for VSAM einen erneuten Öffnungsversuch in einem Nicht-RLS-Modus ausgibt oder ob ein Öffnungsfehler auftritt.

Mögliche Werte	Standardwert
YES/NO	YES

TIMEOUT - Timeout in Sekunden für eine RLS-Anfrage

Dieser Parameter gibt die Zeit in Sekunden an, die Natural for VSAM wartet, um eine Sperre für einen Natural for VSAM-Datensatz zu erhalten, wenn eine Sperre für den Datensatz bereits von einem anderen Benutzer gehalten wird. Weitere Einzelheiten finden Sie im IBM-Handbuch *z/OS DFSMS* Version 1.6 oder höher, *Macro Instructions for Data Sets*.

Mögliche Werte	Standardwert
0 - 10	0

DEFER - Schreibvorgänge in LSR-Pools aufschieben

Dieser Parameter gilt nur im Batch-Modus und unter TSO.

Dieser Parameter gibt an, ob Schreibvorgänge auf die Platte im LSR-Pool aufgeschoben werden sollen. Wenn dies der Fall ist und der LSR-Pool voll wird, schreibt Natural die 5% des Poolbereichs auf die Platte, die am längsten nicht benutzt wurden.

Mögliche Werte	Standardwert
YES/NO	NO

DSECTS - List VSAM System DSECTs

Der Parameter DSECTS gibt an, ob die VSAM-System-DSECTs aufgelistet werden sollen oder nicht.

Mögliche Werte	Standardwert
YES/NO	NO

COMMIT - Unterstützung von Buffer Flush für LSR Pools

Dieser Parameter gilt nur im Batch-Modus und unter TSO.

Der COMMIT-Parameter gibt an, ob bei jedem END TRANSACTION-Statement eines Anwenderprogramms alle nicht festgeschriebenen Aktualisierungen in einem beliebigen LSR-Pool auf Platte geschrieben werden sollen.

Mögliche Werte	Standardwert
YES/NO	NO

Anmerkung: Die Angabe von COMMIT=YES bewirkt eine erhebliche Erhöhung der E/A-Rate.

ERROR - Initialisierungsfehler ausgeben

Dieser Parameter gibt einen Natural-Initialisierungsfehler aus, wenn eine DD- oder DLBL-Karte in der Laufzeit-JCL ausgelassen wird (siehe auch das Makro NTVLSR).

Mögliche Werte	Standardwert
YES/NO	YES

Wenn der Wert NO gesetzt ist, wird die Verarbeitung fortgesetzt und Natural for VSAM wird initialisiert.

HFACTOR - Faktor für Hiperspace-Puffer

Der Parameter HFACTOR gibt einen Faktor für die Erstellung von ESO-Hiperspace-Puffern an. Wenn ein solcher Hiperspace initialisiert wird, kann die entsprechende BLDVRP-Anforderung zu einer Natural-Fehlermeldung führen. In diesem Fall muss der Wert von HFACTOR verringert werden.

Mögliche Werte	Standardwert
0 - ein Wert, bei dem eine entsprechende Natural-Fehlermeldung zurückgegeben wird.	100

READINT - Leseintegrität für Upgrade-Set

Der Parameter READINT gibt an, ob die Leseintegrität für ein Upgrade-Set gewährt werden soll oder nicht.

Mögliche Werte	Standardwert
YES/NO	NO

SMARTS - Unterstützung von SMARTS und Com-plete

Der Parameter SMARTS ist erforderlich, wenn Natural for VSAM unter SMARTS und/oder in einer Com-plete-Umgebung installiert wird.

Mögliche Werte	Standardwert
YES/NO	NO

TVS - Unterstützung von DFSMS Transactional VSAM Services (DFSMStvs)

Der Parameter TVS gibt die Unterstützung von DFSMStvs in einer z/OS-Umgebung an.

Mögliche Werte	Standardwert
YES/NO	NO

23 Betrieb

In diesem Kapitel finden Sie Informationen zu verschiedenen Aspekten, die beim Betrieb von Natural for VSAM relevant sind:

Aufrufen von Natural for VSAM

Wenn die Natural-Schnittstelle zu VSAM verfügbar ist, wird sie initialisiert, wenn Sie eine Natural-Sitzung starten. Sie kann ausgeschaltet werden, indem Sie den Parameter `VSIZE` auf `0` setzen (siehe auch die entsprechende Beschreibung im Abschnitt *Natural für VSAM anpassen*).

OPEN/CLOSE-Verarbeitung

In diesem Abschnitt sind mit VSAM-Dateien sowohl VSAM-Benutzerdateien als auch VSAM-Natural-Systemdateien gemeint.

Die Datenbank-`OPEN/CLOSE`-Verarbeitung wird durch den Natural-Parameter `OPRB` gesteuert (siehe Natural-*Parameter-Referenz*-Dokumentation).

Anstelle des `OPRB`-Parameters können Sie auch das Makro `NTOPRB` des Natural-Parametermoduls verwenden.

Auf einen `OPEN/CLOSE`-Fehler muss die Fehlermeldung NAT3539 folgen. In einer TP-Umgebung kann die Fehlermeldung NAT3516 auch während einer aktiven Natural-Sitzung auftreten, wenn die Datei geschlossen wird.

Anmerkung: Für die dynamische `OPEN`-Behandlung innerhalb einer Sitzung können Sie die Anwendungsprogrammierschnittstelle **USR2008N** verwenden.

OPRB-Parameter für VSAM-Datenbanken

Der Parameter `OPRB` ist unter CICS oder Com-plete nicht anwendbar, da in diesen Umgebungen der TP-Monitor die `OPEN/CLOSE`-Verarbeitung von VSAM-Dateien steuert.

Standardmäßig, d.h. ohne Angabe des `OPRB`-Parameters, werden VSAM-Dateien für die Ein-/Ausgabe geöffnet, so dass sie gelesen und/oder aktualisiert werden können.

Wenn Sie möchten, dass alle verwendeten VSAM-Dateien nur für die Eingabe geöffnet werden, geben Sie den `OPRB`-Parameter mit der folgenden Syntax an:

```
OPRB = (.ALL)
```

Mit dieser Syntax geben Sie eine OPEN-Anforderung für *alle* zu adressierenden VSAM-Dateien an. Alle Dateien werden nur zur Eingabe geöffnet. Einzelne Dateien werden jedoch nur geöffnet, wenn sie tatsächlich von einem bestimmten Programm angesprochen werden.

Anmerkung: Wenn Sie wollen, dass alle VSAM-Systemdateien zur Eingabe geöffnet werden, müssen Sie den Natural-Profilparameter ROSY=ON setzen (siehe Natural-*Parameter-Referenz*-Dokumentation).

Wenn Sie VSAM-Dateien zur Eingabe (I) oder Ausgabe (O) pro DBID öffnen wollen, verwenden Sie die folgende Syntax:

```
OPRB = (DBID = nnn, { MODE = { I } [, string ; ...] } [, ...] )
                    { MODE = { O }                   }
                    { string ; ...                   }
```

Mit MODE geben Sie eine globale Standardbehandlung für DBID *nnn* an.

Wenn Sie keine Standardbehandlung pro DBID angeben wollen oder wenn Sie für einige VSAM-Dateien eine andere Ein-/Ausgabebehandlung als die Standardbehandlung wünschen, können Sie den String-Parameter auf geeignete Weise angeben.

Die DBID muss mit dem NTDB-Makro als VSAM DBID definiert werden, und *string* ist vom Betriebssystem abhängig (siehe unten).

Wichtig: Wenn mehrere Strings definiert werden sollen, muss ein Semikolon (;) als Trennzeichen angegeben werden. Wenn nicht, muss das Semikolon weggelassen werden.

Unter z/OS

Unter z/OS geben Sie den String *string* wie folgt an:

```
{ FNR = nnn                                                  }
{                                                            }
{ DD = dd-name , TYP = { K } } { , O } { , B } [,R]
{                      { E } }     I       A
{                      { R } }
{                      { P } }
```

Die angegebenen VSAM-Dateien müssen als DDMs definiert sein. Anstatt jedoch die Dateinummer des Natural-DDM anzugeben, welches der zu adressierenden VSAM-Datei entspricht, können der *dd-name* und der Typ (KSDS, ESDS, RRDS, oder PATH) dieser Datei direkt angegeben werden, so dass Sie nicht erst in das DDM schauen müssen.

Einzelne Dateien können zur Ausgabe (Option `O`), zur Eingabe (Option `I`), vor dem eigentlichen Zugriff (Option `B`) oder beim ersten Zugriff (Option `A`) geöffnet werden, als wiederverwendbare Datei (Option `R`) geöffnet werden.

Aus Leistungsgründen ist es manchmal wünschenswert, den VSAM-Parameter `STRNO` (Stringnummer) zu ändern, um mehr Index- und Datenpuffer bereitzustellen. Standardmäßig verwendet Natural die Stringnummer 3 für die Eingabeverarbeitung und Stringnummer 5 für die Ausgabeverarbeitung. Da `STRNO` in der JCL angegeben wird, können beide Werte mit dem Parameter `AMP` in der entsprechenden DD-Karte geändert werden.

Der nächste Abschnitt "Unter z/VSE" ist nur bei NAT828 und 912 vorhanden!

Beispiel für eine OPRB-Angabe

Das folgende `OPRB`-Beispiel öffnet die angegebenen Dateien zur Eingabe, während nicht angegebene Dateien standardmäßig zur Ausgabe geöffnet werden:

```
OPRB=(DBID=254,MODE=I)
```

oder

```
OPRB=(DBID=254,FNR=21,I,A;FNR=22,I,A)
```

Die VSAM-Datenbankkennung ((DBID) und Dateinummer (FNR), wie im DDM angegeben, sind erforderlich. Option `I` gibt an, dass der entsprechende FNR für die Eingabe geöffnet wird. Option `A` gibt an, dass die entsprechende FNR nur geöffnet wird, wenn die Datei von einem Natural-Programm aufgerufen wird.

Das entsprechende `NTOPRB`-Makro-Beispiel lautet:

```
NTOPRB 254,'MODE=I'
```

oder

```
NTOPRB 254,'FNR=21,I,A';'FNR=22,I,A'
```

Natural-Dateizugriff

Die Natural-Schnittstelle zu VSAM unterstützt VSAM Entry-Sequenced Data Sets (ESDS), Key-Sequenced Data Sets (KSDS), Relative Record Data Sets (RRDS), Variable Relative Record Data Sets (VRDS) und Pfade für alternative Indexe.

Damit Natural auf VSAM-Dateien zugreifen kann, ist für jede VSAM-Datei, die für Natural-Programme zugänglich gemacht werden soll, ein Natural-DDM erforderlich.

In diesem Abschnitt werden die folgenden Themen behandelt:

Natural-Datendefinitionsmodule (DDMs)

Für jede Datei muss ein Datendefinitionsmodul (DDM) eingerichtet werden. DDMs werden mit Predict (Einzelheiten finden Sie in der *Predict*-Dokumentation) oder mit dem Natural-Dienstprogramm SYSDDM erstellt und gepflegt; sie werden in der Natural-Datendiktionär-Systemdatei (FDIC) gespeichert.

Mit VSAM können zusätzlich zu den logischen Natural-DDMs auch VSAM-Benutzer-DDMs aus einem physischen DDM erstellt werden.

Wenn Sie Predict nicht installiert haben, benutzen Sie das Dienstprogramm SYSDDM, um DDMs aus VSAM-Dateien zu erzeugen. Das Dienstprogramm SYSDDM ist in der *Editoren*-Dokumentation beschrieben. Die für VSAM relevanten Teile davon sind in den folgenden Abschnitten beschrieben.

Alle DDMs, die innerhalb einer Sitzung verwendet werden, befinden sich im Natural-Buffer Pool. Dies steigert die Performance und ermöglicht die Synchronisation der DDM-Verwendung über mehrere Sitzungen hinweg.

SYSDDM-Hauptmenü

Die folgenden Funktionen im Hauptmenü des Dienstprogramms `SYSDDM` sind für Natural for VSAM relevant:

Funktion	Erläuterung
Catalog DDM	**DDM katalogisieren** Das DDM, das sich momentan im Arbeitsbereich befindet, wird katalogisiert und steht damit für die Verwendung in Natural-Anwendungen zur Verfügung. Das DDM muss zuvor mit einem `READ`-Kommando (siehe auch *Editor- und Systemkommandos* in der Dokumentation des SYSDDM-Dienstprogramms) in den Arbeitsbereich gestellt oder mit der unten beschriebenen Funktion **Edit DDM** eingegeben worden sein. Weiter unten finden Sie ausführlichere Informationen zu ***Catalog DDM***.
Edit DDM	**DDM bearbeiten** Liest ein DDM aus der Systemdatei FDIC in den SYSDDM-Arbeitsbereich, wo es bearbeitet werden kann.
List DDMs	**Datendefinitionsmodule auflisten** Zeigt einen einzelnen DDM-Quellcode (DDM-Editor wird nicht aufgerufen) oder eine Liste von DDMs an. Das Anzeigeformat und die Optionen sind identisch mit denen

Funktion	Erläuterung
	des Kommandos `LIST DDM` (siehe auch *Editor- und Systemkommandos* in der Dokumentation des SYSDDM-Dienstprogramms).
Copy DDM to Another FDIC File	**DDM in eine andere FDIC-Datei kopieren** Ein oder alle DDMs können in eine andere Natural-Systemdatei (FDIC) und/oder in eine andere Datenbank kopiert werden. Dies ist z. B. bei der Umstellung einer Natural-Anwendung vom Test- auf den Produktionsstatus erforderlich. Zusätzlich zu DDM-Name, DBID und FNR muss bei Natural for VSAM der Dateityp `V` angegeben werden sowie der DD/FCT-Name der Natural-Systemdatei FDIC, wenn die FDIC-Datei eine VSAM-Datei ist.
List DDMs with Additional Information	**DDMs mit Zusatzinformationen auflisten** Zeigt eine Liste der DDMs an, die in der angegebenen FDIC-Systemdatei gespeichert sind. Aus der Liste können Sie einzelne DDMs zur weiteren Bearbeitung auswählen. Diese Funktion unterscheidet sich von der Funktion **List DDMs** dadurch, dass sie zusätzliche Informationen zu den einzelnen DDMs anzeigt. Zu den angezeigten Informationen gehören Dateiname, DBID, Dateinummer, DDM-Länge, Security-Typ (nur in Verbindung mit Natural Security), Dateityp (d.h. `LOG.DDM`, `PHY.FILE`, `LOG.FILE` oder `USERDDM` für VSAM-DDMs) und Bemerkungen wie z.B. die VSAM-Dateiorganisation (KSDS, VRDS, RRDS, ESDS). Einzelheiten finden Sie unter SYSDDM Utility in der Dokumentation der Natural-Editoren.
Delete DDM	**DDM löschen** Löscht ein zuvor katalogisiertes DDM aus der Natural-Systemdatei FDIC. Das DDM bleibt im Arbeitsbereich erhalten. **Wichtig:** Wenn ein DDM mit SYSDDM gelöscht wird, wird auch das zugehörige Natural Security-Dateiprofil automatisch gelöscht.

Die folgenden für Natural for VSAM relevanten Parameter können für die verschiedenen Funktionen angegeben werden:

Parameter	Erläuterung
`DDM Name`	Der Name des zu bearbeitenden DDM.
`FNR`	Die Dateinummer des zu bearbeitenden DDM.
`DBID`	Die Kennung der Die Datenbank, in der sich das zu bearbeitende DDM befindet.
`Replace`	Wenn `Y` eingegeben wird, werden DDMs, die gerade kopiert oder katalogisiert werden und die bereits vorhanden sind, ersetzt. Wenn `N` eingegeben wird, werden solche DDMs nicht ersetzt.
`FDIC Type`	Der Typ der Systemdatei FDIC.
`DDM Type`	Der Typ des DDM. Für VSAM muss der Typ `V` sein.

Parameter	Erläuterung
DBID Type	Der Typ des DDM. Für VSAM muss der Typ V sein.

DDM katalogisieren – Funktion: Catalog DDM

Ein DDM kann entweder durch Eingabe des Funktionscodes C im SYSDDM-Hauptmenü oder durch Eingabe des Kommandos CATALOG in der Kommandozeile des DDM-Editors katalogisiert werden.

Für diese Funktion sind Dateiname und Dateinummer erforderlich. Bei Natural for VSAM muss eine dem VSAM zugeordnete Datenbankkennung (DBID) angegeben werden. Wenn keine DBID angegeben wird, wird sie als 0 angenommen und zur Ausführungszeit dynamisch auf der Basis der DBID der verwendeten Natural-Systemdatei FUSER erzeugt (siehe auch die Beschreibung des Profilparameters UDB in der Natural-*Parameter-Referenz*-Dokumentation).

Wenn eine DBID angegeben wird, die VSAM zugeordnet ist (und V für VSAM im Feld **Type of this DDM**), fordert SYSDDM Sie zur Eingabe zusätzlicher Informationen auf.

Anmerkung: Die eigentliche DBID-Zuweisung für VSAM erfolgt mit NTDB-Makros beim Assemblieren des Natural-Parametermoduls. Siehe *Installing Natural for VSAM* in der Installations-Dokumentation.

Zusätzliche Optionen für VSAM-Dateien

Wenn das DDM auf eine VSAM-Datei zugreifen soll, wird ein zusätzlicher Bildschirm angezeigt, der die Eingabe zusätzlicher VSAM-Optionen anfordert:

```
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
```

Die zusätzlichen Optionen für VSAM-Dateien bestehen aus zwei Teilen: **VSAM File Information** and **VSAM File Organization**.

Optionen für VSAM File Information

Option	Erläuterung	
VSAM file name	Der DDNAME/FCT-Eintrag, wie er im TP-Monitor oder bei Verwendung des Batch-Modus definiert ist, z. B.: `//PERSON DD ...` wobei PERSON unter **VSAM file name** eingegeben wird.	
VSAM View (DDM)	Gibt an, ob dieses DDM ein logisches Benutzer-DDM oder ein physisches DDM darstellt.	
	`Y`	Gibt an, dass es sich bei dem DDM um ein logisches DDM handelt, was bedeutet, dass es nicht unbedingt dem physischen Layout der VSAM-Datei entspricht. Ein logisches DDM muss dieselbe Dateinummer verwenden wie das physische DDM, von dem es abgeleitet ist, und das entsprechende physische DDM muss zum Zeitpunkt des Aufrufs des Benutzer-DDMs während der Ausführung existieren. Die Kurznamen des logischen DDM müssen mit den im physischen DDM definierten identisch sein. Die Reihenfolge der Felder innerhalb des DDM kann von der physischen Reihenfolge abweichen. Das Primärschlüsselfeld darf nicht aus dem DDM gelöscht werden. Da das logische DDM eine Teilmenge des physischen DDM ist, erscheinen die entsprechenden Teilmengen der zugrunde liegenden VSAM-Datei dem Benutzer als unabhängige Dateien mit unterschiedlichem Datensatz-Layout. Bei der Verarbeitung eines logischen DDM erhält der Benutzer nur Datensätze aus der entsprechenden Teilmenge und nicht aus einer anderen Teilmenge, die in derselben physischen VSAM-Datei enthalten ist.
	`N`	Zeigt an, dass das DDM ein physisches DDM darstellt. Es kann nur ein DDM mit einer bestimmten Dateinummer als physisches DDM für eine VSAM-Datei verwendet werden. Dieses physische DDM wird von Natural intern zur Berechnung von Feld-Offsets verwendet.

Logische DDMs werden verwendet, um verschiedene Datensatztypen in einer physischen VSAM-Datei zu definieren. Bei der DDM-Generierung werden diese Satztypen durch Angabe eines Präfixes für den Primärschlüssel identifiziert.

Wenn ein logisches DDM gelesen wird, werden nur Sätze aus der VSAM-Datei zurückgegeben, deren Schlüssel mit dem angegebenen Präfix beginnt. Datensätze, die mit einem anderen Präfix beginnen, werden ignoriert. Wenn nicht anders angegeben, entspricht das Präfix der logischen Dateinummer.

Jedem logischen DDM muss ein anderes Präfix zugewiesen werden. Natural verlinkt das Präfix automatisch mit dem logischen Schlüssel. Das Feld-Layout im logischen DDM muss nicht dasselbe sein wie im physischen DDM.

Die folgenden beiden Optionen werden nur verwendet, wenn das DDM eine logische Datei darstellt, die von einer physischen VSAM-Datei abgeleitet werden soll.

Option	Erläuterung
Logical related to FNR	Mit dieser Option wird die Dateinummer des physischen DDM angegeben, von dem die logische Datei oder der DDM abgeleitet ist. Ein logisches DDM entspricht einem Satztyp, der durch ein Präfix gesteuert wird. In einer physischen VSAM-Datei können mehrere logische Satztypen enthalten sein. Die Satztypen werden durch ein Präfix unterschieden, das festlegt, welche Sätze verarbeitet werden sollen. Siehe das folgende **Beispiel**.
User defined prefix	Der Präfixwert, der für die logische Datei zugewiesen werden soll. Der Standardpräfixwert ist die logische Dateinummer (Länge 3).

Beispiel für 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-Dateiorganisationsoptionen

Option	Erläuterung	
KSDS, ESDS, RRDS, VRDS	Der Typ der VSAM-Datei:	
	K	KSDS-Datei (Standard)
	E	ESDS-Datei
	R	RRDS-Datei
	V	VRDS-Datei
Compress file	Gibt an, ob die erweiterte VSAM-Datei (also die größere ISN) verwendet werden soll oder nicht.	
	N	Gibt an, dass die Datei nicht komprimiert werden soll. Die Datei wird in der maximalen Länge (d. h. der Länge aller Felder in dieser Datei) geschrieben, wie sie in SYSDDM oder Predict definiert ist. N ist der Standardwert.
	Y	Gibt an, dass die Datei in variabler Satzlänge geschrieben werden soll. Während der Komprimierung wird der Datensatz rückwärts nach Standardwerten durchsucht, die bei alphanumerischen Feldern leer sind, bei binären Feldern niedrige Werte, bei gepackten Feldern niedrige Werte

Option	Erläuterung	
		mit einer Zone und bei numerischen Feldern `X'F0'`. Die Komprimierung wird beendet, sobald der erste Nicht-Standardwert erkannt oder der erste Deskriptor gefunden wird. Die neu berechnete Länge wird verwendet, um den Datensatz in die Datei zu schreiben. Dies gilt nur für KSDS- und ESDS-Dateien. Durch die Komprimierung der nachgestellten Nullwerte in VSAM-Datensätzen wird der Platzbedarf für VSAM-Datensätze minimiert. Die Anwendungsprogrammierschnittstelle `USR0100N` in der Bibliothek SYSEXT wird zur Verfügung gestellt, um die logische Satzlänge durch ein Natural-Programm verwalten zu können.
Zones X'0C' / X'0F'	In Adabas haben alle positiven gepackten Werte `X'0F'` als Zone. Dieser Wert kann in VSAM ein anderer sein. `F` Gibt an, dass alle gepackten Daten mit der Zone `X'0F'` in die VSAM-Datei geschrieben werden. Dies ist der Standardwert. `C` Gibt an, dass alle gepackten Werte mit der Zone `X'0C'` in die VSAM-Datei geschrieben werden.	

DDM bearbeiten

Um das momentan im Arbeitsbereich geladene DDM zu bearbeiten, können Sie den DDM-Editor des SYSDDM-Dienstprogramms verwenden. Wenn noch kein DDM in den Arbeitsbereich eingelesen wurde, wird ein leerer Bildschirm angezeigt, der die manuelle Eingabe einer DDM-Definition ermöglicht.

Anstatt eine komplette DDM-Definition manuell einzugeben, können Sie auch eine bestehende DDM-Definition in den Arbeitsbereich einlesen, indem Sie `EDIT` *`ddm-name`* in die Kommandozeile des DDM-Editors eingeben. Dieses DDM kann geändert und unter einem anderen Namen katalogisiert werden.

Anmerkung: Wenn Sie ein DDM ändern, müssen alle Objekte, die auf dieses DDM verweisen, neu katalogisiert werden.

DDM-Editor

Beispiel:

```
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
```

Wenn Sie das Kommando HELP oder ein Fragezeichen (?) in die Kommandozeile eingeben, werden die Hilfeinformationen zum Editor angezeigt.

Informationen im Kopfbereich des DDM-Editors:

DDM Name	Der Name, der verwendet wird, um in einem Natural-Programm auf das DDM zu verweisen. Der Name muss innerhalb der angegebenen Natural-Systemdatei eindeutig sein.
Def. Seq.	Die Standardreihenfolge, in der die Datei gelesen wird, wenn in einem Natural-Programm mit einem `READ LOGICAL`-Statement auf sie zugegriffen wird.
DBID	Die Datenbank, in der die Datei, auf die mit dem DDM zugegriffen werden soll, enthalten ist. Bei Natural for VSAM muss eine dem VSAM zugeordnete DBID angegeben werden. Wenn `0` angegeben wird, wird die Standard-DBID für die Natural-Systemdatei FUSER, wie im Natural-Parametermodul definiert, verwendet. **Anmerkung:** Die tatsächlichen DBID-Zuweisungen für VSAM werden mit `NTDB`-Makros bei der Assemblierung des Natural-Parametermoduls vorgenommen; siehe Installation von Natural for VSAM in der Installations-Dokumentation.
FNR	Die Nummer der Datei, auf die verwiesen wird. Die angegebene Dateinummer wird von Natural for VSAM intern verwendet.

Das DDM selbst umfasst die folgenden Felddefinitionsattribute, die eingegeben oder geändert werden können:

Attribute	Erläuterung
I	Zeilenkennzeichen. Dieses Feld wird vom DDM-Editor zur Markierung von Zeilen verwendet. E Zeilen, in denen bei der Ausführung des CHECK-Kommandos ein Fehler festgestellt wurde. S Zeilen, die einen gescannten Wert enthalten. X/Y Zeilen, die für den Kopier-/Verschiebevorgang ausgewählt wurden.
T	Feldtyp: G Gruppenüberschrift M Multiples Feld. P Feldüberschrift für Perioden-Gruppe. * Kommentarzeile. *leer* Elementares Feld.
L	Ebene: Dem Feld zugewiesene Level-Nummer. Gültige Level-Nummern sind 1 - 7. Die Level-Nummern müssen in fortlaufender aufsteigender Reihenfolge angegeben werden.
DB	Der zweistellige Code für VSAM-Dateien, der in VSAM verwendet wird.
Name	Ein 3- bis 32-stelliger externer Feldname. Dies ist der Feldname, der in Natural-Programmen verwendet wird, um auf das Feld zu verweisen.
F	Feldformat. Gültige Formate finden Sie unter Benutzervariablen, Format und Länge von Benutzervariablen (im Natural-*Leitfaden zur Programmierung*).
Leng	Standardfeldlänge. Diese Länge kann in einem Natural-Programm überschrieben werden. Für numerische Felder (Format N) wird die Länge als *nn*.*m* angegeben, wobei *nn* für die Anzahl der Vorkommastellen und *m* für die Anzahl der Nachkommastellen steht.
S	Dieses Attribut gilt nicht für Natural for VSAM.
D	Deskriptor-Option. A Zeigt an, dass das Feld ein alternativer Index für eine VSAM-Datei ist. P Zeigt an, dass das Feld ein Primärschlüssel ist. S Gibt an, dass das Feld ein primärer Subdeskriptor oder Superdeskriptor ist, d. h. ein Primärschlüssel für eine VSAM-Datei. X Gibt an, dass das Feld ein alternativer Subdeskriptor oder Superdeskriptor ist, d. h. ein alternativer Index für eine VSAM-Datei. Bemerkung Ein Kommentar, der sich auf ein Feld und/oder das DDM bezieht.
Remark	Ein Kommentar, der sich auf ein Feld und/oder das DDM bezieht.

Die meisten der im Natural-Programmeditor verfügbaren Editor- und Zeilenkommandos gelten auch für den Natural-DDM-Editor. Spezielle Kommandos, wie z.B. PROFILE, RENUMBER, SET, SHIFT usw. und einige Zeilenkommandos sind nicht verfügbar. Weitere Informationen zu den Editorkommandos finden Sie unter *DDM-Editor (SYSDDM Utility)* und *Programm-Editor* in der Dokumentation zu den Natural-Editoren.

Erweiterte Bearbeitung auf Feldebene – Extended Field Editing

Der DDM-Editor kann auch verwendet werden, um DDM-Definitionen auf Feldebene einzugeben oder zu ändern.

Der erweiterte Editiermodus dient zur Angabe von Feldüberschriften und Editiermasken, die bei Verwendung des Feldes in einem `DISPLAY`- oder `INPUT`-Statement angewendet werden sollen, sowie zur Angabe weiterer Spezifikationen für VSAM-DDM-Definitionen. Alle anderen feldspezifischen Informationen (Feldtyp, Länge, Name, Format, Bemerkungen) können an dieser Stelle ebenfalls geändert werden.

Der erweiterte Bearbeitungsmodus wird durch die Eingabe des Zeilenkommandos `.E` an den ersten Stellen der Zeile, die das Feld enthält, aufgerufen.

Ein Bereich von Felddefinitionen kann zur Bearbeitung ausgewählt werden, indem `.E`*nnn* eingegeben wird, wobei *nnn* die Anzahl der auszuwählenden Felder ist.

Der Bearbeitungsmodus auf Feldebene wird beendet, wenn Sie Enter drücken, egal ob Sie Änderungen vorgenommen haben oder nicht.

Auf dem Bildschirm **Extended Field Editing** werden spezielle Attribute der Felddefinition angezeigt, wenn es sich bei dem in Bearbeitung befindlichen DDM um ein VSAM-DDM handelt:

```
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
```

Die folgenden Attribute können angegeben werden:

Attribut	Erläuterung
Alternate Index Name	Verweist das Feld auf einen alternativen VSAM-Index oder einen Pfad (gekennzeichnet durch ein `A` in Spalte **D**), muss der Index- oder Pfadname hier eingegeben werden.
Maximum Occurrence	Die Anzahl der Ausprägungen für ein multiples Feld oder eine Periodengruppe (gekennzeichnet durch ein `M` oder `P` in Spalte **T**).
Die folgenden Flags gelten nur für alternative Indizes und nicht für Pfade:	
Upgrade Flag	Da Natural keine VSAM-Pfade verwendet, kann das Upgrade entweder von Natural oder von VSAM durchgeführt werden, wenn eine KSDS- oder ESDS-Datei mit definierten alternativen Indexen verwendet wird. Ein leerer Wert zeigt an, dass die Aktualisierung des alternativen Indexes von VSAM durchgeführt werden soll, was der Standard ist. Wenn VSAM die Aktualisierung durchführen soll, definieren Sie die VSAM-Datei unter Einsatz von IDCAMS mit `UPGRADE`. Wenn Sie ein `X` eingeben, wird das Upgrade des alternativen Indexes von Natural durchgeführt. In diesem Fall muss AIX mit der Option `NONUPGRADE` definiert werden. **Anmerkung:** Für die LSR-Behandlung ist es empfehlenswert, diese Option anzugeben. Unter CICS muss der FCT-Eintrag auch die Option `VARIABLE` enthalten.
Sort Flag	Ist diese Option mit einem `X` markiert, soll der alternative Index in auf- oder absteigender Wertreihenfolge gelesen werden. Diese Option wird nur wirksam, wenn auch die Option **Upgrade Flag** angegeben ist.
Unique Key Flag	Wenn diese Option mit einem `X` markiert ist, stellt Natural sicher, dass die Werte des alternativen Indexfeldes eindeutig sind. Der Versuch einer Aktualisierung mit einem nicht eindeutigen Wert führt zu einer Fehlermeldung. Der Standardwert ist ein Leerzeichen. Diese Option wird nur wirksam, wenn auch die Option **Upgrade Flag** angegeben ist.
Null Flag	Der Wert `S` gibt an, dass Nullwerte für das alternative Indexfeld unterdrückt werden. Der Standardwert ist ein Leerzeichen. Diese Option wird nur wirksam, wenn auch die Option **Upgrade Flag** angegeben ist.

Anmerkung: Bei allen DDMs, die mit Natural katalogisiert sind und alternative Indexe sowie Angaben zu den oben genannten Flags enthalten, werden alle Flags zur Laufzeit aufgehoben, sobald die Pfadverarbeitung für diese DDMs aktiviert wird.

Die letzten beiden Felder auf dem Bildschirm dienen der Definition von Sub-/Superdeskriptoren für eine VSAM-Datei. Um z.B. das Feld `S1` als Superdeskriptor zu definieren, der im Feld `BA` beginnt und im Feld `BB` endet, muss folgendes eingegeben werden:

```
S1 redefines BA with offset 0
```

Das Feld S1 muss in VSAM als Primär- oder Sekundärschlüssel definiert worden sein.

VSAM-Superdeskriptoren können nur aus zusammenhängenden Feldern gebildet werden. Um das Feld S2 als Superdeskriptor zu definieren, der an der 11. Position des Feldes BA beginnt und mit den ersten beiden Positionen des Feldes BB endet, muss Folgendes eingegeben werden:

```
S2 redefines BA with offset 10
```

Außerdem muss die Länge von S2 auf 7 gesetzt werden. Wie bereits erwähnt, muss S2 als Primär- oder Sekundärindex für VSAM definiert worden sein.

Einschränkungen bei der DDM-Generierung im Vergleich zu Adabas

- Innerhalb von Periodengruppen können keine Schlüssel definiert werden.
- Deskriptoren, die multiple Felder enthalten, sind bei VSAM nicht zulässig.
- Natural-DDMs für VSAM können keine multiplen Felder oder Periodengruppen *innerhalb* von Periodengruppen enthalten.
- Ein und dasselbe Feld kann nicht mehrmals im selben DDM definiert werden. Eine Datendefinition wie im folgenden Beispiel würde bei der Verwendung mit VSAM zu unvorhersehbaren Ergebnissen führen:

Beispiel:

```
...
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 würde das Feld MIDDLE-I nicht als Neudefinition von MIDDLE-NAME behandeln, sondern als ein separates Feld.

Puffer für die Speicherverwaltung

Der Parameter VSIZE ist in zehn verschiedene Bereiche suballokiert, deren Größe durch den Aufbau des Natural-Parametermoduls bestimmt wird. Die verschiedenen VSAM-Bereiche sind in feste und variable Puffertypen unterteilt. Wenn im VSIZE-Puffer nicht genügend Platz für alle Bereiche des Natural-Parametermoduls vorhanden ist, erhalten Sie bei der Initialisierung die Fehlermeldung NAT3592. Zur Laufzeit kann bei festen Puffertypen die Fehlermeldung NAT3513 auftreten. In diesem Fall müssen Sie den entsprechenden Wert des Natural-Parametermoduls anpassen. Variable Puffer werden zur Laufzeit vergrößert, NAT3513 tritt nicht auf. Einige Puffergrößen hängen von der Verwendung von VSAM-Systemdateien ab. Die relevanten Puffer sind FCT, FWA, TSA und UPD.

Der VSIZE-Parameter wird wie folgt suballokiert:

- FCT - Dateikontrolltabelle
- FWA - Dateiarbeitsbereich
- OPV - Open-Anforderungstabelle
- SFT - Systemdatei-Tabelle
- SWT - Umschalttabelle
- TAF - Tabelle der zugegriffenen Dateien
- ROLL - Tabelle der Sitzungsstatusinformationen
- DFB - Tabelle der dekodierten Formatpuffer
- TSA - Tabelle für sequenziellen Zugriff
- UPD - Tabelle der Aktualisierungssätze
- VCA - Natural-Kontrollbereich für VSAM

FCT - Dateikontrolltabelle

FCT (File Control Table) enthält dateispezifische Informationen und ist ein fester Puffertyp.

FCT enthält auch den vollständigen VSAM-Zugriffskontrollblock (ACB), Informationen über vorhandene User-Exits und Informationen über die Anwendungsprogrammierschnittstelle USR0100N.

Die Größe der Tabelle wird anhand der folgenden Formel ermittelt und anschließend auf eine Doppelwortgrenze aufgerundet:

(72 + *ACB-length*) (TAFE * 2) + 80

Ohne VSAM-Systemdateien ist die Standardeinstellung wie folgt:

(72 + 76) (10 * 2) + 80 = 3040

Mit VSAM-Systemdateien ist die Standardeinstellung:

(72 + 76) (26 * 2) + 80 = 7776

FCT und **SWT** (siehe unten) teilen sich einen gemeinsamen Pufferbereich.

FWA - Dateiarbeitsbereich

FWA (File Work Area) enthält Informationen über eine VSAM-Anforderung und ist ein fester Puffertyp.

FWA enthält auch Informationen über die VSAM-Anforderungsparameterliste (Request Parameter List, RPL).

Die Größe der Tabelle wird anhand der folgenden Formel ermittelt und dann auf eine Doppelwortgrenze aufgerundet.

(40 + *RPL-length*) (TAFE * 2) + 80

Ohne VSAM-Systemdateien ist die Standardeinstellung wie folgt:

(40 + 76) (10 * 2) + 80 = 2400

Mit VSAM-Systemdateien ist die Standardeinstellung:

(40 + (76*4)) (26 * 2) + 80 = 17968

FWA und OPV (siehe unten) teilen sich einen gemeinsamen Pufferbereich.

OPV - Open-Anforderungstabelle

OPV (Open Table) enthält Informationen über einen OPRB-String und ist ein fester Puffertyp.

Die Größe der Tabelle wird anhand der folgenden Formel ermittelt und dann auf eine Doppelwortgrenze aufgerundet:

24 * (TAFE * 2) + 48

Die Standardeinstellung ist:

24 * (10 * 2) + 48 = 528

OPV und **FWA** teilen sich einen gemeinsamen Pufferbereich.

SFT - Systemdatei-Tabelle

Die Tabelle SFT (System File Table) ist nur aktiv, wenn VSAM-Systemdateien definiert sind. Der Puffertyp ist fest vorgegeben.

Dieser Bereich enthält die Beschreibung der VSAM-Systemdateien FNAT, FUSER, FDIC, FSEC und FSPOOL sowie die von Natural ISPF verwendete Systemdatei, falls vorhanden.

Die Größe des Bereichs beträgt `8192` bei `SFILE=ON`. Die Standardeinstellung ist `0`.

SWT - Umschalttabelle

SWT (Switch Table) enthält Informationen, die für die Anwendungsprogrammierschnittstelle `USR1047N` zur dynamischen DD/DLBL-Änderung benötigt werden. SWT wird nur zugeordnet, wenn der für den Parameter `DDSWITE` in `NTVSAM` angegebene Wert größer als `0` ist.

Der SWT-Puffertyp ist fest vorgegeben.

Die Größe der Tabelle wird anhand der folgenden Formel ermittelt und dann auf eine Doppelwortgrenze aufgerundet:

24 * DDSWITE + 48

Die Standardeinstellung ist `0`.

SWT und **FCT** (siehe oben) teilen sich einen gemeinsamen Pufferbereich.

TAF - Tabelle der zugegriffenen Dateien

Der -Bereich der TAF (Table of Accessed Files) beschreibt das Datensatzlayout für jede Datei, auf die Natural zugreift. Er wird durch Lesen des physischen oder logischen DDM für die Datei erstellt. Jeder TAF-Eintrag besteht aus einem Header-Eintrag und einem Eintrag für jedes Feld im DDM. Der Header-Eintrag beschreibt den Typ der Datei, den Dateinamen, den Primärschlüssel usw. Die Feldeinträge beschreiben das Format, den Offset und die Länge jedes Feldes in der Datei. Die Layouts für die Header- und Feldeinträge werden in den Makros `NVMTAF` bzw. `NVMFLD` beschrieben.

Der TAF-Puffertyp ist fest vorgegeben.

Die Größe der Tabelle wird mit Hilfe der folgenden Formel ermittelt, wobei auf eine Doppelwortgrenze aufgerundet wird:

(((32 * TAFN) + 112) * TAFE) + 80

Die Standardeinstellung ist:

(((32 * 50) + 112) * 10) + 80 = 17200

ROLL - Tabelle der Sitzungsstatusinformationen

Die ROLL-Tabelle (Table of Session Status Information) wird verwendet, um die Position innerhalb einer Datei für jedes aktive `FIND`- oder `READ`-Statement zu ermitteln. Sie wird durch die CID identifiziert. Dadurch kann Natural alle VSAM-Ressourcen während einer ROLLOUT-Operation (Auslagerung) freigeben und sich dann nach einer ROLLIN-Operation (Einlagerung) wieder korrekt positionieren.

Der Typ des ROLL-Puffers ist fest vorgegeben.

Die Größe dieses Bereichs wird durch den Subparameter `ROLLSIZ` des Makros `NTVSAM` im Natural-Parametermodul bestimmt, wobei auf eine Doppelwortgrenze aufgerundet wird:

TAXSIZE + 80

Die Standardeinstellung ist:

550 + 80 = 632

DFB - Tabelle der dekodierten Formatpuffer

Die DFB (Table of Decoded Format Buffers) ist in zwei Bereiche gegliedert, einen für globale Formatkennungen (GFIDs) und einen für Kommandokennungen (CIDs).

In diesem Bereich wird für jede gegebene E/A-Anforderung beschrieben, welche Felder aus dem VSAM-Datensatzbereich in den Natural Record Buffer zurückgegeben werden. Jeder DFB-Eintrag (decodierter Formatpuffer) besteht aus einem Header, der durch die CID oder die GFID der E/A-Anforderung identifiziert wird, sowie einem Eintrag für jedes Feld, das an Natural zurückgegeben werden soll. Jeder Feldeintrag im DFB enthält das Format, den Offset und die Länge des Feldes, wie sie sich aus dem zugehörigen TAF-Eintrag für die Datei ergeben. Die Layouts der Header- und Feldeinträge sind in den Makros `NVMDFB` bzw. `NVMDFF` beschrieben.

Der Typ des DFB-Puffers ist fest vorgegeben. Tritt die No-Space-Bedingung für GFID-orientierte Einträge auf, werden die ältesten Einträge gelöscht.

Die Größe des TDFB-Bereichs wird anhand der folgenden Formel ermittelt und anschließend auf eine Doppelwortgrenze aufgerundet:

(16 * DFBN * 2 + 36) * DFBE * 2 + 128

Die Standardeinstellung ist:

(16 * 50 * 2 + 36) * 10 * 2 + 128 = 32848

TSA - Tabelle für sequenziellen Zugriff

Die TSA (Table of Sequential Access) wird verwendet, um wichtige Zeiger und Informationen zu jedem READ- oder FIND-Statement zu speichern. Es gibt einen TSA-Eintrag für jedes aktive READ- und FIND-Statement, und jeder Eintrag wird durch die CID identifiziert. Das Layout des TSA ist im Makro NVMTSA beschrieben.

Der Typ des TSA-Puffers ist variabel.

Die Größe des Bereichs wird anhand der folgenden Formel ermittelt und dann auf eine Doppelwortgrenze aufgerundet:

(104 + KEYLGH) * TSAE + 80

Dabei ist TSAE = TSA-Eintrag.

Die Standardeinstellung ist:

(104 + 32) * 10 + 80 = 1440

UPD - Tabelle der Aktualisierungssätze

Der Bereich UPD (Table of Update Records) enthält einen Eintrag für jede READ- oder FIND-Schleife, die ein UPDATE- oder DELETE-Statement enthält. Diese Einträge werden freigegeben, wenn ein END TRANSACTION- oder BACKOUT TRANSACTION-Statement ausgeführt wird. Jeder Eintrag enthält Kontrollinformationen über den Datensatz und die Werte aller Felder, die innerhalb der Schleife aktualisiert werden können. Das Layout jedes UPD-Eintrags ist im Makro NVMUPD beschrieben.

Der Typ des UPD-Puffers ist variabel.

Die Größe des UPD-Bereichs wird durch den Subparameter UPDL des Makros NTVSAM im Natural-Parametermodul ermittelt, wobei auf eine Doppelwort-Grenze aufgerundet wird.

Die Standardeinstellung ist 8272 ohne VSAM-Systemdateien und 32848 mit VSAM-Systemdateien.

VCA - Natural-Kontrollbereich für VSAM

VCA (Natural Control Area for VSAM) ist ein Bereich fester Länge, der Zeiger, Adressen, Flags und Arbeitsbereiche enthält, die für eine Natural-Umgebung für VSAM wichtig sind. Das Layout für diesen Bereich ist im Makro NVMCA beschrieben. In einer Natural-Umgebung für VSAM zeigt R3 immer auf diesen Bereich.

Die Größe dieses Bereichs beträgt 6744 Bytes.

Anwendungsprogrammierschnittstellen

Für Natural for VSAM stehen die folgenden Anwendungsprogrammierschnittstellen (APIs) in der Natural Library SYSEXT zur Verfügung:

API	Funktion
USR0100N	Steuert die variable VSAM-Satzlänge (LRECL).
USR1047N	Unterstützt die dynamische Umschaltung von DD/DLBL-Namen, die in einem DDM definiert sind.
USR2008N	Unterstützt dynamische OPEN-Aufrufe für VSAM-Datasets.

Eine Kurzbeschreibung der APIs finden Sie im folgenden Abschnitt. Ausführlichere Informationen erhalten Sie, wenn Sie sich bei der System-Library SYSEXT anmelden und das Textobjekt (USR*xxxx*T) anzeigen, das der gewünschten API entspricht, oder dazu das Natural-Dienstprogramm SYSEXT benutzen.

Der folgende Abschnitt enthält Informationen zu den folgenden APIs:

- USR0100N
- USR1047N
- USR2008N

USR0100N

Die API USR0100N steuert die Satzlänge von variablen VSAM-Dateien.

Die API wird wie folgt aufgerufen (ein Beispielprogramm namens USR0100P ist in der Bibliothek SYSEXT enthalten):

```
CALLNAT 'USR0100N' parm1 parm2 parm3 parm4 parm5
```

Die Parameter sind in der folgenden Tabelle beschrieben:

Parameter	Format/Länge	Erläuterung
parm1	A1	Gibt einen der folgenden Funktionscodes an: G Für Abfrage-Statements. Die aktuelle Datensatzlänge wird für *parm5* bestimmt. P Für Aktualisierungs-/Speicher-Statements. Die in *parm5* angegebene Länge wird zur aktuellen Datensatzlänge.
parm2	A8	Gibt den DD/DLBL-Namen für die aktuelle Datei an (optional). Falls angegeben, ist *parm5* nur für diese Datei gültig.

Parameter	Format/Länge	Erläuterung
parm3	N5	Gibt die DBID aus dem DDM an (optional). Wird anstelle des DD/DLBL-Namens und nur in Verbindung mit *parm4* verwendet.
parm4	N5	Gibt die FNR aus dem DDM an (optional).
parm5	N5	Gibt die Datensatzlänge an oder gibt sie zurück, abhängig von der Einstellung von *parm1*.

Anmerkung: Wenn weder *parm2* noch *parm3* und *parm4* angegeben sind, gilt *parm5* für alle Dateien.

USR1047N

Die Anwendungsprogrammierschnittstelle USR1047N ermöglicht die dynamische Änderung von DD/DLBL-Namen innerhalb eines Natural-Programms, wenn der Subparameter DDSWITE im Makro NTVSAM angegeben ist. Sie kann verwendet werden, wenn Daten über mehrere VSAM-Dateien verteilt sind, die unterschiedliche DD/DLBL-Namen, aber die gleiche Satzstruktur haben.

Die API wird wie folgt aufgerufen (ein Beispielprogramm mit dem Namen USR1047P ist in der Library SYSEXT enthalten):

```
CALLNAT 'USR1047N' parm1 parm2 parm3 parm4
```

Die verschiedenen Parameter sind in der folgenden Tabelle beschrieben:

Parameter	Format/Länge	Erläuterung
parm1	A1	Gibt einen der folgenden Funktionscodes an: S Zum Umschalten von DD-Namen bei den nächstfolgenden Datenbankaufrufen. R Zum Zurücksetzen der DD-Namen; der Switch-Tabelleneintrag der Funktion S wurde gelöscht (siehe ***SWT - Umschalttabelle***).
parm2	A8	Gibt den alten DD-Namen aus dem DDM an.
parm3	A8	Gibt den neuen DD-Namen für die nächsten Datenbankaufrufe an.
parm4	P4	Rückgabecode von Natural für VSAM.

Der Parameter *parm4* kann die folgenden Rückgabecodes enthalten:

Code	Erläuterung
0	Normale Rückgabe.
4	Die Umschalttabelle (SWT) ist zu klein. Erhöhen Sie den Subparameter DDSWITE im Makro NTVSAM.
8	Der Umschalttabelleneintrag wurde nicht gefunden; Programmfehler.
12	Ungültiger Funktionscode.
16	Die Umschalttabelle ist nicht zugeordnet, d.h. der Parameter DDSWITE ist auf 0 gesetzt.

USR2008N

Diese Anwendungsprogrammierschnittstelle (API) ist unter Com-plete und CICS nicht anwendbar.

USR2008N unterstützt dynamische OPEN-Aufrufe während einer Natural-Sitzung, wenn OPSUPP=ON im Makro NTVSAM angegeben ist.

Die API wird wie folgt aufgerufen (ein Beispielprogramm namens USR2008P ist in der Bibliothek SYSEXT enthalten):

```
CALLNAT 'USR2008N' parm1 parm2 parm3 parm4 parm5 parm6
```

Die Parameter sind in der folgenden Tabelle beschrieben:

Parameter	Format/Länge	Erläuterung
parm1	N5	Gibt die DBID aus der NTDB-Makro-Definition an; siehe ***NTDB-Makro*** im Abschnitt *Natural für VSAM anpassen*.
parm2	A1	Gibt den globalen OPEN MODE an. Siehe ***OPEN/CLOSE-Verarbeitung***.
parm3	A4	Gibt den Datenverwaltungstyp an, z.B. VSAM.
parm4	A40/16	Gibt die gültige OPRB-Syntax und/oder den DDM-Langnamen anstelle der Definitionen DD= oder FNR= an.
parm5	P4	Gibt die Natural for VSAM-Fehlernummer zurück.
parm6	A50	Gibt den Natural for VSAM-Fehlertext zurück.

24 Natural-Statements und Natural-Transaktionslogik mit VSAM

Dieses Kapitel beschreibt Besonderheiten, die hinsichtlich der Natural-Statements und Systemvariablen bei der Verwendung mit VSAM zu berücksichtigen sind. Darüber hinaus wird die Natural-Transaktionslogik im Zusammenhang mit VSAM behandelt.

Die Natural-Statements, die für den Zugriff auf VSAM-Dateien verwendet werden, sind eine Untermenge der in der Programmiersprache Natural enthaltenen Statements. Für den Zugriff auf eine VSAM-Datei sind keine neuen Statements erforderlich, da jedes Natural-Statement unabhängig vom Datenbankmanagementsystem oder der verwendeten Zugriffsmethode dieselbe Funktion erfüllt. Daher können Programme, die für VSAM-Dateien geschrieben wurden, auch für den Zugriff auf Adabas-Datenbanken verwendet werden.

Die Natural-Schnittstelle zu VSAM hat keine eingebaute Transaktionslogik und verwendet die Transaktionslogik der Umgebung, in der sie läuft. Dies führt je nach Umgebung zu unterschiedlichen Resultaten.

Natural-Statements mit VSAM

Dieser Abschnitt umfasst in der Hauptsache Informationen, die auch in der Dokumentation zu Natural-Statements enthalten sind. Dort wird jede Natural-Statement im Detail beschrieben, gegebenenfalls mit Hinweisen zur Verwendung bei VSAM. Im Folgenden sind die besonderen Punkte zusammengefasst, die ein Programmierer bei der Verwendung von Natural-Statements mit VSAM beachten muss.

Anmerkung: Da der Natural-Compiler nicht prüft, ob ein Programm die durch die Natural-Schnittstelle zu VSAM auferlegten Beschränkungen einhält, treten VSAM-spezifische Programmierfehler bei der Verwendung von Natural-Statements erst bei der Ausführung des Programms auf.

Natural-Statements, die in diesem Abschnitt nicht erwähnt wird, können ohne Einschränkungen mit VSAM verwendet werden.

- BACKOUT TRANSACTION
- DELETE
- END TRANSACTION
- FIND
- GET
- GET SAME
- GET TRANSACTION DATA
- HISTOGRAM
- READ
- STORE

- UPDATE

BACKOUT TRANSACTION

Das BACKOUT TRANSACTION-Statement wird verwendet, um alle Datenbankaktualisierungen, die während der aktuellen logischen Benutzertransaktion durchgeführt wurden, rückgängig zu machen. Dieses Statement gibt außerdem alle während der Transaktion gehaltenen Datensätze frei.

Bei Verwendung mit Natural for VSAM gibt das BACKOUT TRANSACTION-Statement Datensätze in der UPD-Tabelle frei. Es werden keine Transaktionen rückgängig gemacht, es sei denn, Natural läuft unter einem TP-Monitor oder DFSMStvs, der das dynamische Rückgängigmachen von Transaktionen unterstützt (z. B. CICS). In diesem Fall wird ein ROLLBACK zum letzten SYNCPOINT ausgegeben.

DELETE

Das DELETE-Statement wird verwendet, um einen Datensatz aus einer VSAM-Datei zu löschen.

Die Verwendung des DELETE-Statements setzt jeden in dem entsprechenden FIND- oder READ-Statement ausgewählten Datensatz in den Hold-Status.

Das DELETE-Statement ist nicht gültig bei VSAM Entry-Sequenced Data Sets (ESDS).

END TRANSACTION

Das END TRANSACTION-Statement wird verwendet, um das Ende einer logischen Transaktion anzuzeigen. Eine logische Transaktion ist die kleinste logische Arbeitseinheit (wie vom Benutzer definiert), die in ihrer Gesamtheit ausgeführt werden muss, um die logische Konsistenz der in der VSAM-Datei enthaltenen Informationen zu gewährleisten.

Das END TRANSACTION-Statement gibt auch alle Sätze frei, die während der Transaktion in den Hold-Status gesetzt wurden.

Ein END TRANSACTION-Statement gibt nur die in der UPD-Tabelle gehaltenen Datensätze frei, es sei denn, Natural läuft unter einem TP-Monitor oder DFSMStvs, der ein dynamisches Rückgängigmachen (Backout) von Transaktionen unterstützt (z. B. CICS). In diesem Fall bewirkt ein END TRANSACTION-Statement, dass ein SYNCPOINT ausgegeben wird.

FIND

Das FIND-Statement wird verwendet, um einen Set von Datensätzen aus der VSAM-Datei anhand eines Suchkriteriums auszuwählen, das aus Feldern besteht, die als Deskriptoren (Schlüssel) definiert sind.

Die WITH-Klausel wird verwendet, um das Suchkriterium anzugeben, das aus in der VSAM-Datei definierten Schlüsselfeldern (Deskriptoren) besteht.

Es können nur VSAM-Schlüsselfelder verwendet werden.

Die Anzahl der als Ergebnis einer WITH-Klausel auszuwählenden Datensätze kann durch die Angabe des Schlüsselworts LIMIT zusammen mit einem Grenzwert (*operand1*), der als numerische Konstante oder als benutzerdefinierte Variable ausgedrückt wird, begrenzt werden. Der Limit-Wert wird in Klammern gesetzt. Wenn die Anzahl der ausgewählten Datensätze den Limit-Wert überschreitet, wird das Programm mit einer Fehlermeldung abgebrochen.

Der Deskriptor muss in einer VSAM-Datei als VSAM-Schlüsselfeld definiert sein. In einem DDM wird er mit P für Primärschlüssel, S für primärer Sub-/Superdeskriptor, X für alternativer Sub-/Superdeskriptor oder A für alternativer Schlüssel gekennzeichnet (siehe ***DDM bearbeiten*** im Abschnitt *Betrieb* und die Beschreibung *DDM-Editor (SYSDDM Utility)* in der Natural-*Editoren*-Dokumentation).

Die Formate des Deskriptors und des Suchwertes müssen kompatibel sein.

Die folgenden Natural-Systemvariablen sind beim FIND-Statement verfügbar:

Variable	Inhalt
*ISN	Die Systemvariable *ISN enthält die relative Byte-Adresse des zurzeit in Bearbeitung befindlichen Datensatzes (nur ESDS-Dateien). Diese Variable steht bei den Statements FIND NUMBER und FIND FIRST nicht zur Verfügung.
*NUMBER	Die Systemvariable *NUMBER enthält die Anzahl der Datensätze, die das in der WITH-Klausel angegebene Basissuchkriterium erfüllen, und zwar vor der Auswertung eines WHERE-Kriteriums. *NUMBER enthält nur dann einen aussagefähigen Wert, wenn im Suchkriterium der Operator EQUAL TO verwendet wird. Bei jedem anderen Operator ist *NUMBER 0, wenn keine Datensätze gefunden wurden. Jeder andere Wert zeigt an, dass Datensätze gefunden wurden, aber der Wert hat keinen Bezug zur Anzahl der tatsächlich gefundenen Datensätze. Dasselbe gilt für *NUMBER beim Statement FIND NUMBER.
*COUNTER	Die Systemvariable *COUNTER enthält die Anzahl der Aufrufe der Verarbeitungsschleife. Diese Systemvariable ist beim Statement FIND NUMBER nicht verfügbar.

Das FIND-Statement ist nur für Key-Sequenced (KSDS) und Entry-Sequenced (ESDS) VSAM-Datasets gültig. Für ESDS muss ein alternativer Index oder ein Pfad für einen alternativen Index

definiert werden. Relative Record Datasets (RRDS) sind nicht zulässig, da sie keine Schlüsselfelder (Deskriptoren) enthalten.

GET

Das GET-Statement wird verwendet, um einen Datensatz mit einer bestimmten VSAM-Datensatznummer zu lesen. Bei einer ESDS-Datei ist die Datensatznummer (ISN) die relative Byte-Adresse (RBA). Bei RRDS- und VRDS-Dateien ist es die relative Datensatznummer (RRN). Das GET-Statement initiiert keine Verarbeitungsschleife. Folglich wird ein nachfolgendes UPDATE- oder DELETE-Statement nicht verarbeitet und Natural gibt eine entsprechende Fehlermeldung zurück.

Für ESDS muss die RBA in einer benutzerdefinierten Variablen (numerisches Format) enthalten sein oder als Integer-Konstante angegeben werden. Für RRDS und VRDS gelten dieselben Regeln, mit der Ausnahme, dass die RRN anstelle des RBA angegeben werden muss.

GET SAME

Das GET SAME-Statement gilt nur für VSAM ESDS, RRDS und VRDS (siehe auch das GET-Statement weiter oben).

GET TRANSACTION DATA

Das GET TRANSACTION DATA-Statement ist nicht bei der Natural-Schnittstelle zu VSAM anwendbar.

HISTOGRAM

Das HISTOGRAM-Statement wird verwendet, um die Werte eines Feldes zu lesen, das als Deskriptor, Subdeskriptor oder Superdeskriptor definiert ist.

Das HISTOGRAM-Statement initiiert eine Verarbeitungsschleife, bietet aber keinen Zugriff auf andere Felder als das in dem Statement angegebene Feld.

Als Deskriptoren können nur VSAM-Schlüsselfelder verwendet werden.

Die folgende Natural-Systemvariable ist mit dem HISTOGRAM-Statement verfügbar:

Variable	Inhalt
*NUMBER	Wenn die Systemvariable *NUMBER in Verbindung mit einem KSDS-Primärschlüssel oder einem eindeutigen alternativen Index verwendet wird, ist ihr Inhalt immer 1.

Anmerkung: Die Systemvariable *ISN ist bei der Natural-Schnittstelle zu VSAM nicht verfügbar.

Bei Verwendung mit VSAM ist das HISTOGRAM-Statement nur für KSDS- und ESDS-Datasets gültig. Für ESDS muss ein alternativer Index oder ein Pfad für einen alternativen Index definiert werden.

Die Werte werden direkt aus dem VSAM-Index gelesen und in auf- oder absteigender Reihenfolge zurückgegeben.

READ

Das READ-Statement wird verwendet, um Datensätze aus einer VSAM-Datei zu lesen. Die Sätze können in der Reihenfolge (auf- oder absteigend) der Werte eines Deskriptorfeldes (Schlüssel) abgerufen werden. Die READ-Sequenz initiiert eine Verarbeitungsschleife.

IN LOGICAL SEQUENCE wird verwendet, um Datensätze in der Reihenfolge der Werte eines Deskriptors (Schlüssel) zu lesen. Wenn LOGICAL mit einem Deskriptor angegeben wird, werden die Datensätze in der Reihenfolge der Werte des Deskriptors gelesen. Ein Deskriptor kann zur Reihenfolgesteuerung verwendet werden. Ein Deskriptor innerhalb einer Periodengruppe kann nicht verwendet werden. Wird LOGICAL ohne Deskriptor angegeben, werden die Datensätze in der Standard-Deskriptorreihenfolge gelesen, wie sie im DDM definiert ist.

WITH REPOSITION kann zum Übergehen der sequenziellen Verarbeitung innerhalb der aktiven Schleife verwendet werden. Die neue Position muss als neuer Startwert für die Schleife definiert werden und die Systemvariable *COUNTER zurücksetzen.

IN LOGICAL SEQUENCE ist nur für KSDS mit definierten Primärschlüsseln und Alternativschlüsseln und ESDS mit definierten Alternativschlüsseln gültig. Auch ein Subdeskriptor oder Superdeskriptor kann zur Ablaufsteuerung verwendet werden.

Die folgenden Natural-Systemvariablen stehen beim READ-Statement zur Verfügung:

Variable	Inhalt
*ISN	Die Systemvariable *ISN enthält entweder die RRN (für RRDS oder VRDS) oder die RBA (für ESDS) des aktuellen Datensatzes.
*COUNTER	Die Systemvariable *COUNTER enthält die Anzahl der Einsprünge in die Verarbeitungsschleife.

Datensätze können auch IN PHYSICAL SEQUENCE abgerufen werden, was dazu dient, Datensätze in der Reihenfolge zu lesen, in der sie physisch in einer Datenbank gespeichert sind. Sie ist nur bei VSAM ESDS, RRDS und VRDS gültig. Dies ist die Standardreihenfolge.

STARTING WITH ISN kann als Startwert für die Schleife in aufsteigender oder absteigender physischer Reihenfolge verwendet werden.

BY ISN wird verwendet, um Datensätze in RBA- und RRN-Reihenfolge für ESDS-, RRDS- bzw. VRDS-Dateien zu lesen.

STORE

Das STORE-Statement wird verwendet, um einen Datensatz in einer Datenbank hinzuzufügen.

Ein eindeutiger Wert für das Primärschlüsselfeld oder das Alternate-Index-Feld muss angegeben werden, wenn der Datensatz mit einem Primärschlüssel oder einem eindeutigen Alternate-Index definiert ist.

Die Klausel USING/GIVING NUMBER ist nur bei RRDS oder VRDS gültig. In diesem Fall entspricht die ISN der relativen Datensatznummer.

USING/GIVING NUMBER wird verwendet, um einen Datensatz mit einer vom Benutzer vergebenen RRN zu speichern. Wenn ein Datensatz mit der angegeben RRN bereits existiert, wird eine Fehlermeldung zurückgegeben und die Ausführung des Programms abgebrochen, sofern nicht ON ERROR-Verarbeitung angegeben wurde.

Die Natural-Systemvariable *ISN enthält die RRN, die dem neuen Datensatz als Ergebnis der Ausführung des STORE-Statements zugewiesen wurde.

Eine nachfolgende Referenz auf *ISN muss die Statement-Nummer des entsprechenden STORE-Statement enthalten. *ISN ist nur bei RRDS- oder VRDS-Dateien verfügbar.

UPDATE

Das UPDATE-Statement wird verwendet, um ein oder mehrere Felder eines Datensatzes in einer Datenbank zu aktualisieren. Der zu aktualisierende Datensatz muss zuvor mit einem FIND- oder READ-Statement ausgewählt worden sein.

Der Primärschlüssel kann nicht aktualisiert werden.

Natural-Transaktionslogik im Zusammenhang mit VSAM

Natural for VSAM verwendet die Transaktionslogik der Umgebung, in der es läuft. Daher unterscheiden sich die Ergebnisse der Natural-Statements END TRANSACTION und BACKOUT TRANSACTION (siehe auch die entsprechenden Abschnitte in *Natural-Statements mit VSAM*) je nach der tatsächlichen Umgebung:

- Bei nativem VSAM
- Unter CICS

- Unter DFSMStvs

Bei nativem VSAM

Da VSAM selbst keine Transaktionslogik besitzt, ist auch keine Transaktionslogik verfügbar, wenn Natural in einer nativen VSAM-Umgebung arbeitet. Dies ist der Fall unter Com-plete, TSO und im Batch-Modus, d.h. wenn `NVSMISC` das verwendete E/A-Modul ist.

Bei `NVSMISC` geben die Statements `END TRANSACTION` und `BACKOUT TRANSACTION` keine Fehlermeldungen zurück, sondern werden von der Natural-Schnittstelle zu VSAM ignoriert.

Unter CICS

Unter CICS können VSAM-Dateien als „wiederherstellbare Ressourcen“ oder für RLS als „wiederherstellbare Sphäre“ definiert werden, die alle von CICS unter Verwendung des Konzepts der „logischen Arbeitseinheiten“ (Logical Units of Work, LUW) synchronisiert werden. Eine LUW endet, wenn ein `SYNCPOINT`-Kommando ausgegeben wird oder wenn die CICS-Task beendet wird. Einzelheiten finden Sie in der entsprechenden IBM-Literatur über CICS.

Nachfolgend finden Sie Informationen über:

- Modul NVSCICS
- Tasks im Conversational Mode
- Pseudo-Conversational Tasks

Modul NVSCICS

Für CICS ist das E/A-Modul `NVSCICS` ein normales Anwendungsprogramm auf Kommandoebene. Es übergibt die Statements `END TRANSACTION` und `BACKOUT TRANSACTION` an den `NATCICS`-Treiber, der die Kommandos `EXEC CICS SYNCPOINT` und `EXEC CICS ROLLBACK` ausgibt. Wenn in einer Natural-Sitzung mit nicht festgeschriebenen Aktualisierungen (Uncommitted Updates) ein Fehler auftritt und keine Fehlertransaktion übergeben wird, veranlasst Natural selbst die Schnittstelle zu VSAM zur Ausgabe eines `ROLLBACK`-Kommandos.

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.

Tasks im Conversational Mode

Wenn die Natural-Sitzung im CICS Conversational Mode läuft, wird die LUW nicht durch eine Terminal-E/A beendet. Natural läuft im Conversational Mode, wenn entweder der Natural-Parameter `PSEUDO=OFF` angegeben wurde oder Natural selbst festgestellt hat, dass eine pseudo-konversationelle Verarbeitung nicht möglich ist.

Da Terminal-Ein-/Ausgaben die Transaktionslogik einer Anwendung nicht stören, solange Natural im Conversational Mode läuft, funktioniert ein Programm wie das folgende ohne Probleme:

Example:

```
READ vsam-file
 UPDATE
 INPUT ...
END-READ
BACKOUT TRANSACTION
```

Pseudo-Conversational Tasks

Wenn die Natural-Sitzung im Pseudo-Conversational Mode läuft, beendet jede Terminal-Ein-/Ausgabe die CICS-Task und führt damit implizit ein `SYNCPOINT`-Kommando aus. Die Auswirkungen eines `BACKOUT TRANSACTION`-Statement, d.h. eines `EXEC CICS SYNCPOINT ROLLBACK`-Kommandos, reichen daher nur bis zur letzten Terminal-Ein-/Ausgabe zurück. Das obige Beispielprogramm würde daher mit der Fehlermeldung NAT3548 enden, weil es nicht möglich ist, alle Aktualisierungen zurückzunehmen.

Anmerkung: Beachten Sie, dass alle Meldungen der Natural-Schnittstelle zu VSAM nur zur Laufzeit ausgegeben werden, da der Natural-Compiler nicht in der Lage ist, diese Art von logischen Fehlern zu erkennen.

Unter DFSMStvs

DFSMS Transactional VSAM Services (DFSMStvs) bietet die gleichen Funktionen wie CICS: Vorwärts- und Rückwärts-Wiederherstellungsprotokollierung, Backout-Verarbeitung und einen zweistufigen Commit-Prozess. Eine LUW endet, wenn der RRS-Aufruf (Resource Recovery Service) `SRRCMIT` oder `SRRBACK` ausgegeben wird (`END TRANSACTION` oder `BACKOUT TRANSACTION`). Einzelheiten finden Sie in der entsprechenden IBM-Literatur zu DFSMStvs und RRS.

III 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.

25 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.

26 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.

27 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

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.

Anmerkung: 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.

28 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 `NDPBNSBO`. 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 `NDPBNSBO`, which uses the linked PSB module from the previous step to create NSB control blocks which are stored on the `FDIC` system file. `NDPBNSBO` 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:

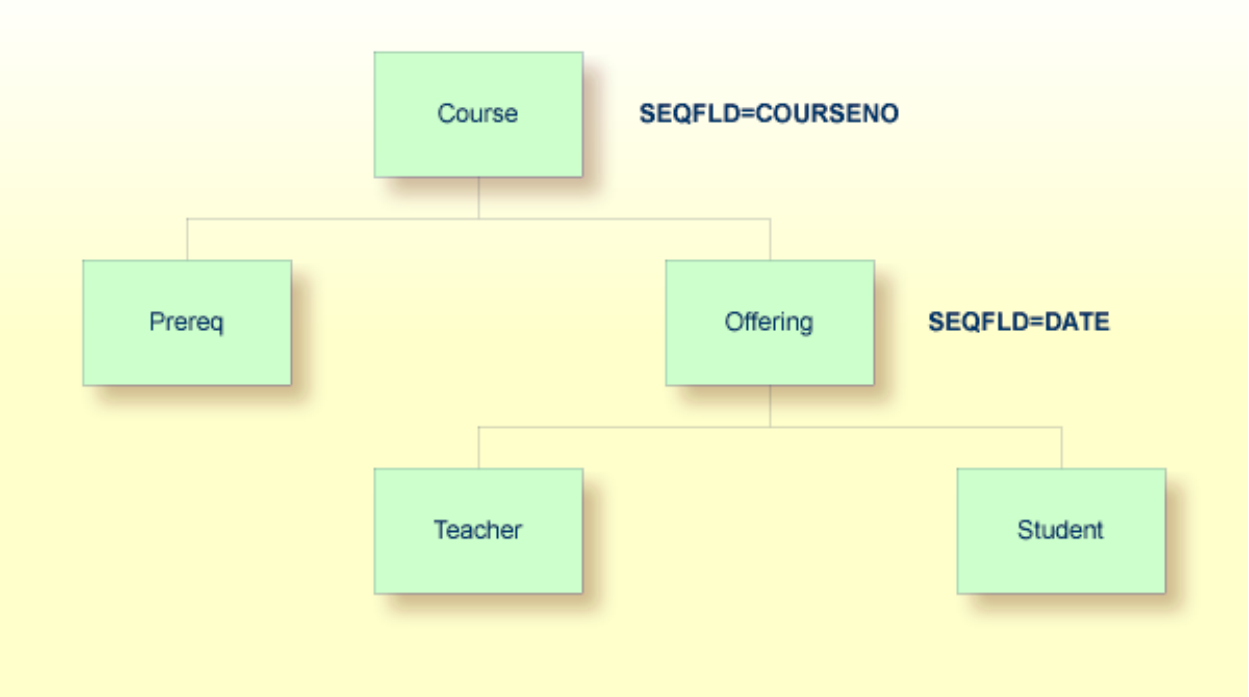
```
NAT3789 Active PSB contains too few PCBs for program execution
```

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.

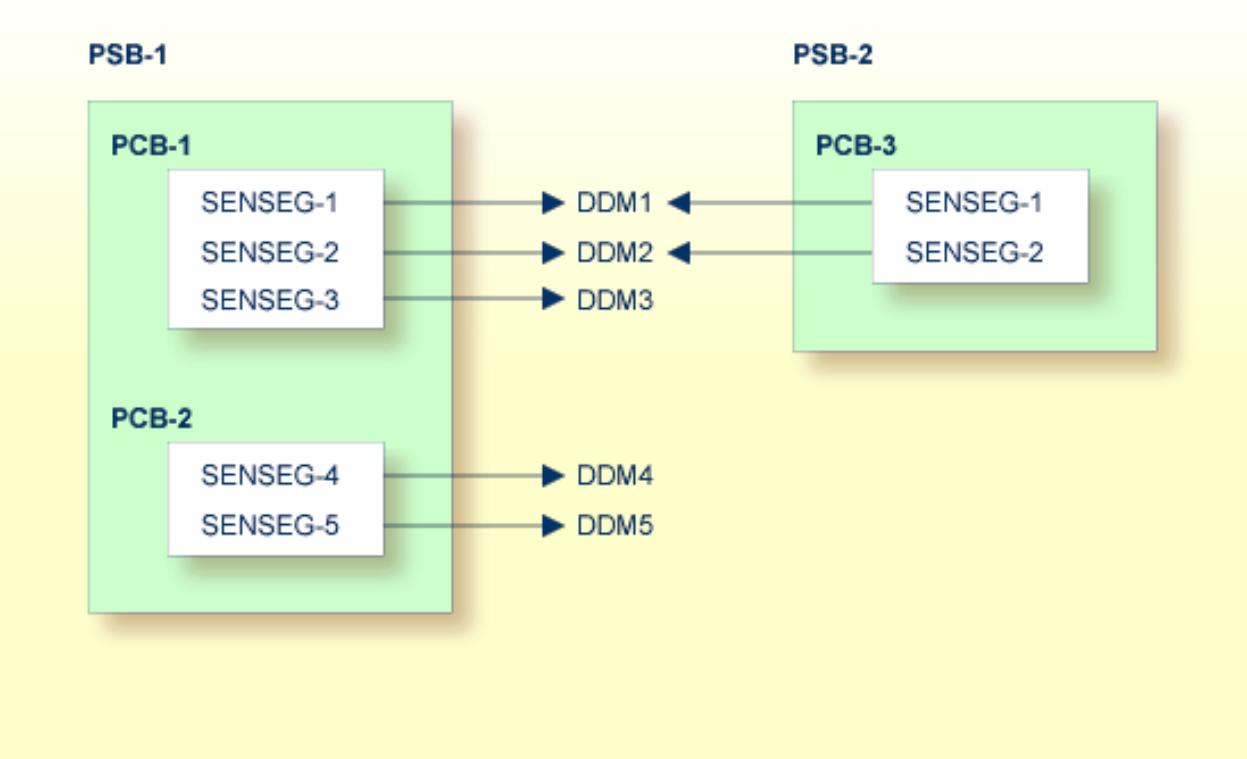
The following example PSB and Natural program demonstrate that the sequence of the PCBs, referring to the same DBD, may affect Natural programs if the PROCOPT parameters are different:

```
PCB    TYPE=DB,DBDNAME=EDOODBD,PROCOPT=GO,...
SENSEG NAME=COURSE
SENSEG NAME=OFFERING,PARENT=COURSE
PCB    TYPE=DB,DBDNAME=EDOODBD,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 `NDPBNDBO`.

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 NDPBNDBO, which uses the linked DBD module from the previous step to create NDB control blocks which are stored on the FDIC system file. NDPBNDBO dynamically loads the Natural module NDLBO001, 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.

Wichtig: 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

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=EDO2DBD,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=$$$$
FUNC=REP,DBD=EDO2DBD,SEGM=COURSE
FUNC=FLD,NAME=DUM41,TYPE=B,LEVEL=1,LENGTH=9
FUNC=STR,BEGIN=DESCRIPN
FUNC=FLD,NAME=DUN2,LEVEL=1,MAXOCC=2
FUNC=STR,BEGIN=TITLE
FUNC=FLD,NAME=GRU21,TYPE=N,LEVEL=2,LENGTH=1
FUNC=STR
FUNC=FLD,NAME=GRU22,TYPE=A,LEVEL=2,LENGTH=2
FUNC=STR
FUNC=FLD,NAME=GRU23,TYPE=N,LEVEL=2,LENGTH=3
FUNC=STR
FUNC=FLD,NAME=$$$$
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=EDO2DBD,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`.

29 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	U*dddfff*	255	253

Anmerkungen:

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 U*dddfff* 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.

30 Natural Batch Utilities

This section covers the following topics:

Transfer of NDBs/NSBs/UDFs from one System File to Another

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.

Anmerkung: 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.

Anmerkung: 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

Anmerkungen:

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              000A0000N00000002000200000
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
```

31 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

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.

Anmerkung: The NATPSB INQ command is valid in an IMS TM environment, too.

The following command, for example, issues a PSB schedule of EDOOPSB:

```
NATPSB ON EDOOPSB
```

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 NAT*vrs* 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.

32 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.

Anmerkungen:

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

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.

Anmerkung: 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.

Anmerkung: 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.

Anmerkung: 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.

33 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** **Anmerkung:** 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**

34 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.

35 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

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 = EDOODBD
   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  EDOODBD  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 `EDOODBD` 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

Anmerkungen:

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:	EDOODBD
Name of segment:	STUDENT
Name of generated DDM:	EDOODBD-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**.

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        EDOODBD
                                     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
```