ADABAS NATIVE SQL STATEMENTS

This chapter covers the following topics:

The BEGIN Statement

```
EXEC ADABAS
    BEGIN DECLARE SECTION
END-EXEC
```

This statement must appear in every program that uses Adabas Native SQL statements. The only Adabas Native SQL statements allowed to precede this statement are COPY and GENERATE.

- In Ada programs, the BEGIN statement must be coded in the data declaration part of the program.
- In COBOL programs it must be coded in the DATA DIVISION.
- In FORTRAN programs it must be coded in the DATA DEFINITION area of the program.

Adabas Native SQL generates Adabas control blocks, format buffers, search buffers, value buffers and other miscellaneous information in response to the BEGIN DECLARE SECTION statement.

The CHECKPOINT Statement

```
EXEC ADABAS
    CHECKPOINT USER= value1
END-EXEC
```

The CHECKPOINT statement is used by update programs that issue checkpoints. It is only applicable to programs that run in exclusive file control mode. One option is available:

- USER

USER

For user checkpoints made in exclusive file control mode. An Adabas C1 command is generated.

value1 is a constant of 4 characters identifying the checkpoint code or the name of a variable containing the checkpoint code. If *value1* is a variable name, it must be preceded by a colon (’:’).

Examples:

```
EXEC ADABAS
  CHECKPOINT USER = ’CK01’
END-EXEC

EXEC ADABAS
  CHECKPOINT USER = :CURRENT-CKPT
END-EXEC
```

The CLOSE Statement

```
EXEC ADABAS
   CLOSE cursor-name
END-EXEC
```

This statement is part of the OPEN/FETCH/CLOSE sequence that is used for processing multiple records. See section 3 for further details.

The CLOSE statement must be used in conjunction with the COMPARE, FIND, FIND COUPLED and SORT statements, that is, with those statements that generate an ISN list. It may be used with the HISTOGRAM, READ LOGICAL and READ PHYSICAL SEQUENCE statements, but its use following these statements is not mandatory.

The CLOSE statement releases various Adabas resources, and it also releases the command-ID from the ISN list table. This makes it impossible to refer to the records after the CLOSE statement has been executed. No more FETCH statements can be executed after the CLOSE has taken place.

This statement generates an Adabas RC command.

The COMMIT WORK Statement

```
EXEC ADABAS
   COMMIT WORK
    [ USERDATA= value ]
END-EXEC
```

The COMMIT WORK statement is used to indicate the end of a logical transaction. It should be issued by ET mode users whenever the program has completed the processing of one logical transaction. Failure to do this may lead to an excessively large hold queue in the Adabas work file and eventually to hold queue overflow.

Should the application program ABEND, the status of the database at the time when the last COMMIT WORK was issued will automatically be restored when Adabas is restarted (autobackout).

An Adabas ET (end-of-transaction) command is generated.

USERDATA Clause

USERDATA= *value*

The user may write data to the Adabas system file using the ’USERDATA = value’ clause. The data can be retrieved using the CONNECT and READ USERDATA statements. value can be a constant enclosed in apostrophes or the name of a variable containing the user data. If value is a variable name, it must be immediately preceded by a colon (’:’). See the examples below.

If the USERDATA clause is used, a CONNECT statement with a valid user-ID must have been executed. The user-ID that was specified in the CONNECT statement is associated with the user data, and it must be quoted when recovering the user data with a subsequent CONNECT or READ USERDATA statement.

This facility can be used to record information required when performing a restart, for example the positions of files that are being processed sequentially.

The length of the user data, i.e., the number of characters to be written, must not exceed the limit specified in the USERDATA clause of the global OPTIONS parameter.

Examples:

```
EXEC ADABAS
  COMMIT WORK
    USERDATA = :USERVAR
END-EXEC

EXEC ADABAS
  COMMIT WORK
    USERDATA = ’TEST1234’
END-EXEC
```

The COMPARE Statement

```
EXEC ADABAS
    COMPARE [ ISN [ LISTS ] ]
  [ DECLARE cursor-name CURSOR [ FOR ] ]
  [ SELECT { field-name ,... } ]
           {      *          }
    FROM     file  [ alias ]
                                 {   AND   }
    WHERE CURSOR=   cursor-name  {   OR    }  CURSOR= cursor-name2
                                 { AND NOT }
    OPTIONS [ { AUTODBID                } ]
            [ { DBID= database-name     } ]
              [ CIPHER= value1 ]
              [ COND-NAME= { Y } ]
              [            { N } ]
              [ HOLD [ RETURN ] ]
              [ INDEXED=  { Y } ]
              [           { N } ]
              [ ISNSIZE= len ]
              [ PASSWORD= value2 ]
              [ PREFIX= prefix ]
              [ SAVE ]
              [ STATIC=  { Y } ]
              [          { N } ]
              [ SUFFIX= suffix ]
END-EXEC
```

The COMPARE statement performs logical processing on two ISN lists that were previously created using FIND, FIND COUPLED or COMPARE statements with the SAVE option. It can compute the intersection (logical AND) or union (logical OR) of two ISN lists, or the set of ISNs that are in one list but not in the other (logical AND NOT).

The two ISN lists to be compared must relate to the same file, and they must be in ascending ISN sequence. Therefore the ORDER BY option is not permitted in the FIND statement that created the ISN lists to be compared.

The ISN lists to be compared must have been created by Adabas Native SQL statements with the SAVE option. They should be released with the CLOSE statement when they are no longer required.

In general, the COMPARE statement will return a list containing the ISNs of many records.

If more than one record is to be processed, the COMPARE statement must contain the DECLARE *cursor-name* CURSOR FOR clause and it must be followed by an OPEN/FETCH/CLOSE sequence as described in chapter [2004-07-02 tbd]. The fields specified in the SELECT clause are available after execution of the FETCH statement.

If only the record whose ISN is at the head of the resulting ISN list is to be processed, the DECLARE clause may be omitted and the fields specified in the SELECT clause are available after execution of the COMPARE statement. In this case Adabas Native SQL generates executable code for the COMPARE statement, which must therefore appear in the procedure division in COBOL programs.

An Adabas S8 command is generated.

DECLARE Clause

```
DECLARE cursor-name CURSOR [ FOR ]
```

This clause specifies a cursor-name that identifies, or labels, the current statement. Subsequent statements can refer back to a statement that is labeled with a DECLARE clause by quoting the cursor-name. The cursor-name may be up to four characters long.

If multiple records are to be processed, the ’DECLARE *cursor-name* CURSOR FOR’ construction must be used. The keyword ’FOR’ indicates to Adabas Native SQL that the statement is used in conjunction with OPEN and FETCH statements that appear later in the program quoting the same cursor-name. If only a single record is to be processed, the DECLARE clause may be omitted.

SELECT Clause

```
SELECT { field-name ,... }
       {     *           }
```

The SELECT clause specifies which fields are to be retrieved from the database. All types of fields may be selected, with the exception of phonetic descriptors. The fields must be specified by their full names as defined in the data dictionary.

If an asterisk is specified following the keyword ’SELECT’, all the fields within the userview are read.

See also the previous discussion on the SELECT clause for more information.

FROM Clause

```
FROM { file [ alias ] } ,...
```

The FROM clause specifies the file from which data is to be retrieved. It is used together with the SELECT clause to generate the record buffer and to control the retrieval of information from the database.

file is the Adabas file name or view name as defined in the data dictionary. The *alias*, if present, is used as the name of the record buffer; otherwise, the name *file* is used.

See also the previous discussion on the FROM clause for more information.

WHERE Clause

```
                                 ⎧ AND     ⎫
WHERE CURSOR=   cursor-name     ⎨ OR      ⎬  CURSOR= cursor-name2
                                 ⎩ AND NOT ⎭
```

The WHERE clause is used to specify the cursor names defined in the FIND or COMPARE statements that created the ISN lists. Both of thes statements should have the SAVE option.

The COMPARE statement can be used to perform the following logical operations:

AND The resulting ISNs will contain those ISNs that are present in both ISN lists.

OR The resulting ISNs will contain those ISNs that are present in either the first ISN list or the second ISN list or both.

AND NOT The resulting ISN list will contain those ISNs that are present in the first ISN list (identified by *cursor-name1*) but not present in the second ISN list (identified by *cursor-name2*).

OPTIONS Clause

```
OPTIONS [ ⎧ AUTODBID               ⎫ ]
        [ ⎩ DBID= database-name    ⎭ ]
          [ CIPHER= value1 ]
          [ COND-NAME= ⎧ Y ⎫ ]
          [            ⎩ N ⎭ ]
          [ HOLD [ RETURN ] ]
          [ INDEXED= ⎧ Y ⎫ ]
          [          ⎩ N ⎭ ]
          [ ISNSIZE= len ]
          [ PASSWORD= value2 ]
          [ PREFIX= prefix ]
          [ SAVE ]
          [ STATIC= ⎧ Y ⎫ ]
          [         ⎩ N ⎭ ]
          [ SUFFIX= suffix ]
```

AUTODBID Option

The AUTODBID option can be used if the file is linked to a single database. This option indicates to Adabas Native SQL that the database ID is to be taken from the data dictionary. If the file is linked to more than one database, an error message will be issued.

The AUTODBID option and the HOLD option may not be used together. This implies to Adabas Native SQL that you are attempting to update a database other than your default database. Also, AUTODBID and DBID may not be used together.

CIPHER Option

The cipher key must be specified when accessing a ciphered file. See also the previous discussion on the CIPHER option for more information.

COND-NAME Option

This option applies only to COBOL programs.

If the option ’COND-NAME = Y’ is coded, the record buffer generated by Adabas Native SQL includes as level-88 entries the condition names defined in Predict.

If specified here, any value specified with the global parameter OPTIONS will be overridden.

With Cond. names in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See also the previous discussion on the COND-NAME option for more information.

DBID Option

This option should be used if the program accesses more than one database. The *database-name* must be defined in the data dictionary, and the data dictionary description of the database must include the file or files to be accessed.

The DBID option and the HOLD option may not be used together. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

HOLD Option

If the HOLD option is coded, the record retrieved is placed in hold status. As long as a record is in hold status, it cannot be updated or deleted by any other user.

A record that is to be updated or deleted must be in hold status unless the program is running in exclusive-control mode.

See *HOLD Logic* for more information.

The HOLD option may not be used together with the AUTODBID, AUTODBID-ALL or DBID options. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

INDEXED Option

This option applies only to COBOL programs.

If the INDEXED option is specified, all multiple-value fields and periodic groups are generated with the ’INDEXED BY’ keywords. The name of the index is taken from the Predict additional field attribute `3GL specification, Indexed by`. If no index name is specified here, the name of the multiple-value field or periodic group is used, prefixed with ’I-’.

Any specification here will override any setting of the global parameter OPTIONS.

Indexed by in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See also the previous discussion on the INDEXED option and section *Generate COBOL Copy Code* in the *Predict Administration Manual* for more information.

ISNSIZE Option

The ISNSIZE parameter defines the maximum number of ISNs that can be stored in the ISN buffer. If the number of records that satisfy the selection criterion exceeds ISNSIZE, the excess ISNs are stored by Adabas and retrieved automatically when required. This process is transparent to the programmer. See also the previous discussion on the ISNSIZE option for more information.

PASSWORD Option

The password must be specified in each Adabas Native SQL statement that accesses a password-protected file or a file that is protected by security by value, unless it is specified globally in the password clause of the CONNECT statement .

See also the discussion on security options for more information.

PREFIX Option

If the option ’PREFIX = *prefix*’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name prefix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See also the previous discussion on the PREFIX option for more information.

SAVE Option

Use this option if you need to retain the entire ISN list. The saved ISN list can be used later in COMPARE, FIND and SORT statements. The saved ISN list is discarded when:

- A further Adabas Native SQL statement that creates another ISN list with the same name (same command-ID) is executed, or:
- An Adabas Native SQL CLOSE or DBCLOSE statement is executed, or:
- The non-activity time limit or transaction time limit is exceeded.

Under these circumstances, response code 9 is returned when the next Adabas command is attempted.

A CLOSE statement must be executed to release the ISN list after every statement that generates an ISN list (COMPARE, FIND, FIND COUPLED and SORT). If the CLOSE statement is not executed, large amounts of storage will be occupied for the remainder of the Adabas session.

STATIC Option

This option applies to PL/I programs only.

If the option ’STATIC = Y’ is coded here, all buffers generated by Adabas Native SQL will be defined as static. This will override any setting of the global parameter OPTIONS (see page).

Static in the Predict Modify PL/I Defaults screen must be marked with an "X" if you want to specify this option. See also the previous discussion on the STATIC option for more information.

SUFFIX Option

If the option ’SUFFIX = *suffix*’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name suffix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See also the previous discussion on the SUFFIX option for more information.

The CONNECT Statement

```
EXEC ADABAS
    CONNECT
     [ userid ]
     [ WITH password ]
     [ ACC= file ,... ]
     [ UPD= file ,... ]
     [ EXU= file ,... ]

         [ AND USERDATA INTO var ]

        OPTIONS
         [ NORESTRICTED ]
         [ DBID= var ]
         [ MAXISN= value1 ]
         [ MAXHOLD= value2 ]
         [ MAXCID= value3 ]
         [ MAXTIME= value4 ]
         [ TT= value5 ]
         [ TNA= value6 ]
         [ ACODE=value7 ]
         [WCODE=value8 ]

END-EXEC
```

The CONNECT statement is used to indicate the beginning of a user session and to list the files that will be used and the modes in which they are to be opened. The CONNECT statement should not be issued by modules called from the main program. If a CONNECT statement is used, it must be in the main program and it must include not only the files used by the main program but also those used by all modules called from the main program. It must be the first executable Adabas Native SQL statement in the program, with the possible exception of COPY and GENERATE statements (compare with the BEGIN statement).

If the CONNECT statement is omitted, the program will run in ET mode. Any files can be read and updated, with only the customary password and cipher restrictions on access.

If the program is to run in exclusive-control mode or if files are to be accessed in access-only mode (all attempts to modify the database will be rejected), then the CONNECT statement must be included.

If the Adabas user session is still active when the CONNECT statement is issued (from a previous program that was not terminated correctly with the DBCLOSE statement), a ROLLBACK WORK will be performed and Response Code 9 is returned.

This statement generates an Adabas OP (open) command.

USERID Clause

```
userid
```

This clause specifies the user-ID for the user session. A user-ID must be provided if you intend to store and/or read user data and you require this data to be available during a subsequent user session or Adabas session (see also the CHECKPOINT, COMMIT WORK, DBCLOSE, READ USERDATA and WRITE TO LOG statements). The value provided for the user-ID should be unique for this user (that is, it should not be used by any other user). Response Code 48 will be returned if the user-ID is already in use.

The first character must be an upper-case letter or a digit. *userid* may be a constant of up to 8 characters, or the name of a variable containing the user-ID. If *userid* is a variable name, it must be immediately preceded by a colon (’:’), for example ’:NAME’.

Note:
If *userid* is a constant, it must be enclosed in single quotes.

Password Clause

```
WITH password
```

You may, if you wish, specify the password only once in your program, in the PASSWORD clause of the CONNECT statement. Adabas Native SQL will pass this password to all generated Adabas commands.

If you also code the PASSWORD option in an Adabas Native SQL statement, the local specification overrides the global specification in the CONNECT statement for that statement only.

password may be a constant of up to 8 characters or the name of a variable containing the password. If *password* is a variable name, it must be preceded by a colon (’:’) for example ’:SECRET’.

ACC Clause

```
ACC= file ,...
```

This is a list of the names of the Adabas files to be accessed by the program in access-only (read-only) mode. Any attempt to update a file opened in access-only mode or to add or delete records will be rejected (response-code=19).

If this clause is present, all files to be processed by the program must be listed in the CONNECT statement. An attempt to read a file that was not specified will cause an error (response code=17).

Example:

```
EXEC ADABAS
  CONNECT ACC = PERSONNEL, AUTOMOBILES, FINANCE
END-EXEC
```

This program uses the files PERSONNEL, AUTOMOBILES and FINANCE in access-only mode.

Adabas Native SQL automatically adds the ABEND file to the ACC list so that the error texts corresponding to non-zero response codes can be retrieved from it as required by the response code interpretation routine. The default is UPD.

UPD and EXU Clauses

UPD= *file* **,...**
EXU= *file* **,...**

All files updated by the program should be specified in the CONNECT statement. An attempt to update a file that is not specified in the CONNECT statement will cause an error (response code=17).

There are two types of update programs:

ET-mode: These are programs that can update files in parallel with other programs updating the same files. Programs that run in ET mode must put the required record in hold status before updating or deleting it, and must issue the COMMIT WORK statement at the end of each logical transaction. This mode is used for online update programs.

Exclusive mode: These are programs that have exclusive use of the selected files. During the entire execution time, other programs are prevented from updating these files.

Thus, one or more of the following possibilities may be specified:

- ’UPD =’ followed by a list of file names, for programs that run in ET mode. The application program should check the response-code after each Adabas Native SQL statement that generates one or more Adabas commands for the value 9, which would mean that an automatic backout has occurred and the program should restart the transaction from the beginning;

- ’EXU =’ followed by a list of file names, for EXCLUSIVE mode;

Further information about exclusive control updating may be found in the *Adabas Command Reference Manual* and the *Adabas DBA Reference Manual*. Consult your DBA before writing programs that run in exclusive file control mode or file cluster mode.

Examples:

```
EXEC ADABAS
  CONNECT 'MEMUNE'
          ACC = PERSONNEL UPD = AUTOMOBILES
END-EXEC
```

The program uses the PERSONNEL file in access-only mode and updates the AUTOMOBILES file in ET-logic mode.

```
EXEC ADABAS
  CONNECT 'MEMUNE'
          UPD = PERSONNEL EXU=PERSONNEL
END-EXEC
```

The program uses the PERSONNEL file in access-only mode and updates the PERSONNEL file in ET-logic mode.

USERDATA Clause

AND USERDATA INTO *var*

This clause enables the user to retrieve the user data stored in the Adabas system file by a CHECKPOINT, COMMIT WORK or DBCLOSE statement.

The last USERDATA record that was stored with a CHECKPOINT, COMMIT WORK or DBCLOSE statement is read into var. var must be preceded by a colon (':'), for example ':NAME'.

This option may only be used if the user specified the same user-ID for the current user session and also for the session during which the USERDATA were stored.

OPTIONS Clause

```
OPTIONS
  [ NORESTRICTED ]
  [ DBID= var ]
  [ MAXISN= value1 ]
  [ MAXHOLD= value2 ]
  [ MAXCID= value3 ]
  [ MAXTIME= value4 ]
  [ TT= value5 ]
  [ TNA= value6 ]
  [ ACODE=value7 ]
  [ WCODE=value8 ]
```

Note:
Default values for all values except DBID are specified in the Predict Modify Adabas Native SQL Defaults screen.

NORESTRICTED option

If this option is used, the Adabas OPEN command will be generated without the RESTRICTED option, so files which are not specified in CONNECT may be added later to the Adabas user queue element.

DBID Option

This option should be used if the program accesses more than one database. The *database-name* must be defined in the data dictionary, and the data dictionary description of the database must include the file or files to be accessed.

Note:
Unless using the AUTODBID-ATM global option , only read or search access is permitted if the DBID option is used; the INSERT, UPDATE and DELETE statements must not be used. One CONNECT statement must be issued for each database to be accessed.

MAXISN Option

This option specifies the maximum number of ISNs resulting from the execution of Sx commands that Adabas can store internally in its ISN table. Increasing the default setting may reduce access to the Adabas work file.

MAXHOLD Option

This option specifies the maximum number of records that the user may have in hold status at any time.

MAXCID Option

This option specifies the maximum number of Command IDs that may be active for the user at the same time.

MAXTIME Option

This option specifies the time limit for the execution of Sx commands.

The programmer should consult with the DBA about the system default values for these parameters before changing them. For further details, see the *Adabas Command Reference Manual*, section *OP Command*, paragraph *Additions 4*.

TT Option

This option may be used to specify a transaction time limit.

TNA Option

This option may be used to specify a non-activity time limit.

ACODE option

This option allows for providing the encoding key for "A" format fields during the user session.

WCODE option

This option allows for providing the encoding key for "W" format fields during the user session.

The COPY Statement

```
1

EXEC ADABAS
    COPY  file-name  [member-name]
END-EXEC
```

```
2

EXEC ADABAS
    COPY
        FILE=  [file-name]
        MEMBER=  [member-name]
END-EXEC
```

Adabas Native SQL supports the COPY statement as described in the chapter *The Preprocessor* of the *Predict Administration Manual*. A file layout that has been generated as Ada, COBOL, FORTRAN or PL/I code by Predict can be copied into the program by means of this statement.

The *file-name* must always be specified. It is the name of the file as defined in the data dictionary.

The *member-name* must be specified if more than one file layout has been generated for this file.

The *file-name* and *member-name* can be specified as positional parameters (see [1] above) or as keyword parameters (see [2] above).

The DBCLOSE Statement

```
EXEC ADABAS
    DBCLOSE
      [ USERDATA=  var ]
  [ OPTIONS
        DBID=  [database-name] ]
END-EXEC
```

The DBCLOSE statement is used to terminate a user session. All Adabas resources are released.

This statement may be issued at the end of the main program. It should not be issued by modules called by a main program, nor should it be issued at the end of a TP transaction program unless this coincides with the end of the user session.

USERDATA Clause

```
USERDATA= var
```

The user may store user data in the Adabas system file by including the ’USERDATA = *var*’ clause. The user data can be retrieved by a subsequent CONNECT or READ USERDATA statement. *var* is the name of the variable containing the user data. The variable name must be immediately preceded by a colon (’:’), for example ’USERDATA = :NAME’. The length of the user data, that is the number of characters to be written, must not exceed the limit specified in the USERDATA clause of the global OPTIONS parameter.

This statement generates an Adabas CL (close) command.

Example:

```
EXEC ADABAS
  DBCLOSE
     USERDATA = :USERVAR
END-EXEC
```

OPTIONS Clause

```
OPTIONS
  [ DBID=   database-name ]
```

DBID Option

This option may be used if the program has accessed more than one database. The database-name must be defined in the data dictionary, including the file or files that have been accessed. One DBCLOSE statement should be issued per database.

The DELETE Statement

```
EXEC ADABAS
   DELETE
     [ DECLARE   cursor-name   CURSOR ]
       FROM  file1  [alias]

                         ISN= value
       WHERE {                                }
                 CURRENT OF cursor-name1
     [ OPTIONS                              ]
     [   [ PASSWORD= value1 ]              ]
     [   [ CIPHER= value2 ]                ]

END-EXEC
```

The DELETE statement deletes a record from the specified file. The record to be deleted must be retrieved by the FIND statement or one of the READ statements before issuing the DELETE statement. The record must be in hold status unless the program is running in EXU mode (see the CONNECT statement). A record can be ’held’ either by specifying the ’HOLD’ option in the statement that reads it, or by issuing a separate HOLD statement. If the record is not in hold status, it will implicitly be ’held’.

When the logical transaction has been completed, a COMMIT WORK statement should be issued. One of the effects of this statement is to release records that are in hold status.

This statement generates an Adabas E1 command.

DECLARE Clause

```
DECLARE  cursor-name  CURSOR
```

The cursor-name may be up to four characters long. The DECLARE clause will not normally be required, but it may be used if desired to define the Adabas command ID.

Note: This clause should not be used if the WHERE CURRENT OF clause is used.

FROM Clause

```
FROM  file1  [alias]
```

file1 is the Adabas file name or view name, as defined in the data dictionary, of the file from which the record is to be deleted. If the same file appears in another statement, an *alias* should be used.

WHERE Clause

```
WHERE  {        ISN= value         }
       {  CURRENT OF cursor-name1  }
```

The WHERE clause is used to specify the ISN of the record to be deleted.

In order to delete a record whose ISN is explicitly known, the ’WHERE ISN = *value*’ option should be used. *value* may be a constant or the name of a variable containing the ISN. If *value* is a variable name, it must be immediately preceded by a colon (’:’), for example ’:NAME’. The colon must not be coded following the ’=’ sign if *value* is a numeric constant, for example ’WHERE ISN = 1234’.

The option ’WHERE CURRENT OF *cursor-name1*’ should be coded in order to delete a record found by a previous Adabas Native SQL statement. *cursor-name1* is the name of the cursor defined in that statement.

OPTIONS Clause

```
OPTIONS
        [ PASSWORD= value1 ]
        [ CIPHER= value2 ]
```

PASSWORD Option

The password must be specified in each Adabas Native SQL statement that accesses a password-protected file or a file that is protected by security by value, unless it is specified globally in the CONNECT statement. See also the previous discussion on this option for more information.

CIPHER Option

The cipher key must be specified when accessing a ciphered file.

See also the previous discussion on this option for more information.

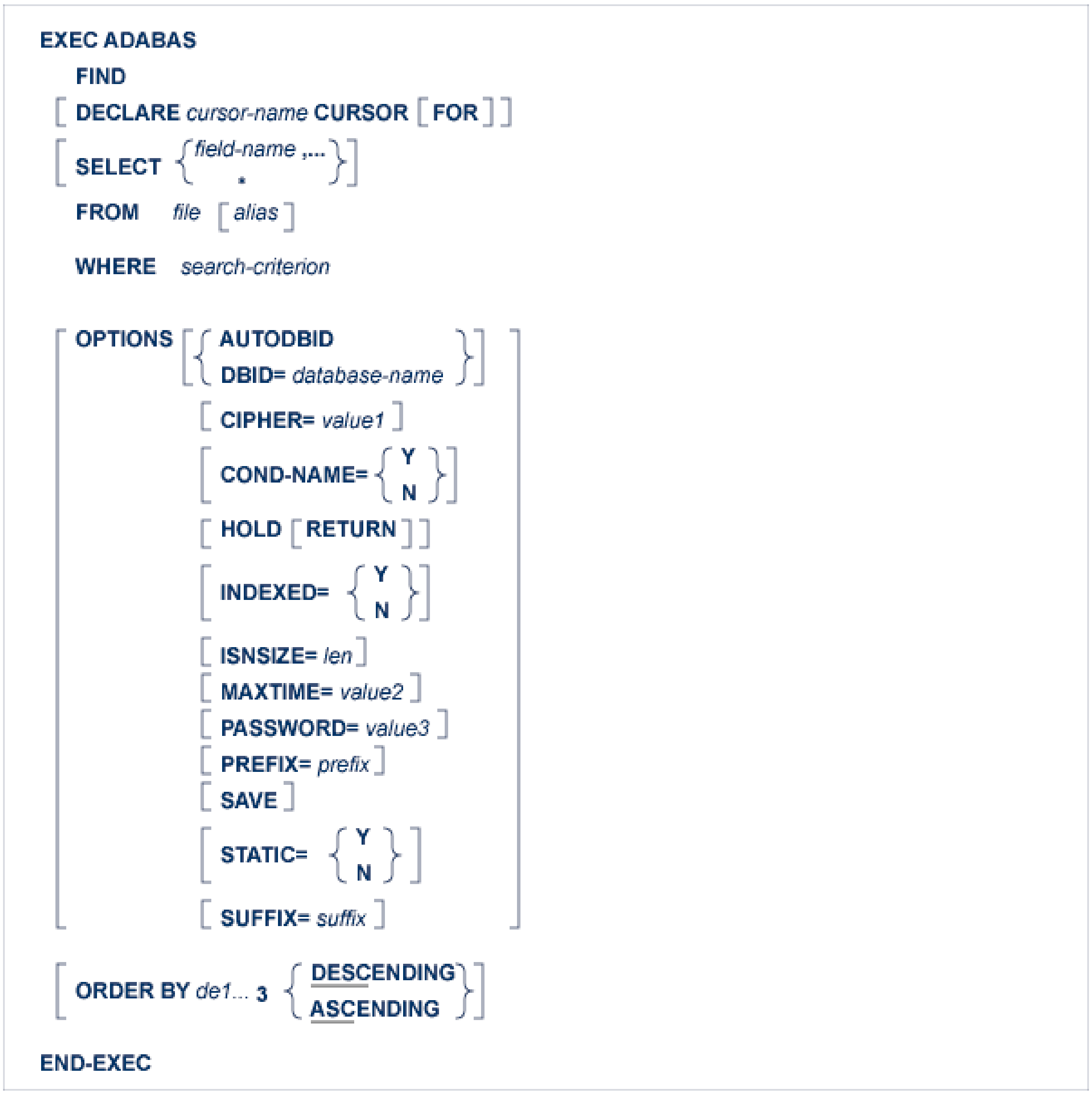
The FETCH Statement

```
EXEC ADABAS
    FETCH cursor-name
END-EXEC
```

This statement is part of the OPEN/FETCH/CLOSE sequence that is used for processing multiple records. See *Multiple-Record Processing* for more information.

The FETCH statement reads the data from the file into the record buffer. An OPEN statement must always be issued before the first FETCH statement can be executed when using multiple-record processing. Each successive FETCH statement automatically reads the next record (or delivers the next descriptor value in the case of the HISTOGRAM statement), until all the records have been passed to the user program. When all records have been read, an end-of-file condition is encountered and the flag ADACODE is set to 3.

The FIND Statement



```
EXEC ADABAS
   FIND
 [ DECLARE cursor-name CURSOR [ FOR ] ]
 [ SELECT { field-name ,...
             *           } ]
   FROM    file  [ alias ]

   WHERE   search-criterion

 [ OPTIONS [ { AUTODBID
               DBID= database-name } ]
             [ CIPHER= value1 ]
             [ COND-NAME= { Y
                            N } ]
             [ HOLD [ RETURN ] ]
             [ INDEXED= { Y
                          N } ]
             [ ISNSIZE= len ]
             [ MAXTIME= value2 ]
             [ PASSWORD= value3 ]
             [ PREFIX= prefix ]
             [ SAVE ]
             [ STATIC= { Y
                         N } ]
             [ SUFFIX= suffix ]                ]

 [ ORDER BY de1... 3 { DESCENDING
                       ASCENDING } ]

END-EXEC
```

The FIND statement performs a retrieval query against a database file, selecting the record or records specified by the search criterion in the WHERE clause. The keyword ’FIND’ may optionally be omitted.

This statement returns either a list of the ISNs of the records that satisfy the search criterion, or an ’end-of-file’ indication in the variable ADACODE (Ada, COBOL or PL/I) or SQLCOD (FORTRAN), indicating that no records satisfied the search criterion.

In general, the FIND statement will return a list containing the ISNs of many records.

If all of the records found by the FIND statement are to be processed, then the FIND statement must include the ’DECLARE *cursor-name* CURSOR FOR’ clause and it must be followed by an OPEN/FETCH/CLOSE sequence as described previously. The fields specified in the SELECT clause are available after execution of the FETCH statement.

If only the first of these records is to be processed, then the DECLARE clause may be omitted and the fields specified in the SELECT clause are available after execution of the FIND statement. In this case, ADABAS Native SQL generates executable code for the FIND statement, which must therefore appear in the procedure division in COBOL programs.

The FIND statement can only retrieve data from the first file (main file) named in the FROM-clause, although the search criterion can include descriptor fields taken from up to five physically-coupled or 16 soft-coupled files (except in the case of VMS which does not support coupled files). The coupling relationships must be defined in PREDICT. If data fields are to be retrieved not from the main file but from a coupled file, the FIND COUPLED statement must be used in conjunction with the FIND statement.

The FIND statement causes an ADABAS S1/S4 command to be generated, or an S2 command if the ’ORDER BY’ clause is coded.

DECLARE Clause

```
DECLARE cursor-name CURSOR [FOR]
```

This clause specifies a cursor-name that identifies, or labels, the current statement. Subsequent statements can refer back to a statement that is labeled with a DECLARE clause by quoting the cursor-name. The cursor-name may be up to four characters long.

If multiple records are to be processed, the ’DECLARE *cursor-name* CURSOR FOR’ construction must be used. The keyword ’FOR’ indicates to ADABAS Native SQL that the statement is used in conjunction with OPEN and FETCH statements that appear later in the program quoting the same cursor-name. If only a single record is to be processed, the DECLARE clause may be omitted.

See also the previous discussion on this clause for more information.

SELECT Clause

```
SELECT { field-name ,... }
       {      *         }
```

The SELECT clause specifies which fields are to be retrieved from the database. All types of fields may be selected, with the exception of phonetic descriptors. The fields must be specified by their full names as defined in the data dictionary.

If the SELECT clause is omitted, then no records are processed, but other functions such as search may be performed.

If an asterisk is specified following the keyword ’SELECT’, all the fields within the userview are read.

See also the previous discussion on this clause for more information.

FROM Clause

```
FROM   file [alias]
```

The FROM clause specifies the file or files that contain the fields used in the search criterion. It is also used, in conjunction with the SELECT clause, to generate the record buffer and to control the retrieval of information from the database.

file is the ADABAS file name or view name as defined in the data dictionary. The *alias*, if present, is used as the name of the record buffer; otherwise, the name *file* is used.

In the FIND statement, up to 5 physically-coupled or 16 soft-coupled files may be specified in the FROM clause. This facility is used if the search criterion includes fields taken from more than one file. Every file containing fields used in the search criterion must be listed in the FROM clause. Data can only be retrieved from the first file (main file) whose name directly follows the keyword 'FROM'.

The second and subsequent files listed in the FIND statement must be physically coupled to the main file. Note that the names of the coupled files are separated by commas, but the alias is not preceded by a comma.

See also the previous discussion on this clause for more information.

Note:
In VMS only one file can be specified in the FROM clause, because coupled files are not supported.

WHERE Clause

```
WHERE   search-criterion
```

The WHERE clause identifies the set of records to be selected. Only database fields that are defined as descriptors, subdescriptors, superdescriptors, hyperdescriptors or phonetic descriptors may be used to form the search-criterion. In ADABAS Version 5, non-descriptor fields may be used, if the NONDE indication in the ADABAS Native SQL DDA allows it.

The search-criterion is made up of descriptors, logical operators and values, according to the type of selection relevant to the application.

search-criterion

```
search-expression  [ { AND }  search-expression ... ]
                     { OR  }
```

search-expression

```
  ┌  { file1  }           { file2  }       ┐
  │  { alias1 } .de1  =   { alias2 } .de2  │
  │  descriptor  comp  value               │
 {   descriptor  BETWEEN  value1  AND  value2  [exception]  }
  │  descriptor  IN (  value,...  )        │
  │  descriptor  IS  [NOT]  NULL           │
  └  SETID= 'cursor-name'                  ┘
```

descriptor

```
  ┌  de1                ┐
 {   { file  }  .de2     }
     { alias }
  └  de3(i)             ┘
```

comp

```
  ┌ NE ┐
  │ EQ │
  │ GT │
  │ GE │
  │ LT │
 {  LE  }
  │ =  │
  │ >  │
  │ >= │
  │ <  │
  └ <= ┘
```

exception

```
AND  { fieldname  NOT=  value3                      }
     { fieldname  NOT BETWEEN  value3  AND  value4  }
```

de1 is the name of the descriptor to be used in the search expression. The name must refer to a descriptor, subdescriptor, superdescriptor, hyperdescriptor or phonetic descriptor. *de1* is a descriptor in the main file, that is, the file whose name appears first in the FROM clause, directly following the keyword.

The second construct, *file.de2*, shows the name of a descriptor (*de2*) qualified by the filename (*file*). The qualifier is required if the descriptor is in a coupled file, i.e., is not in the main file.

de3 (i) is a reference to a specific occurrence of a descriptor which is a field in a periodic group.

file1.de1 = file2.de2

This construction is used to connect two files via the soft coupling option of Adabas 5. The relationship should exist in Predict.

Example:

```
EXEC ADABAS
    FIND
    SELECT *
    FROM PERSONNEL,AUTOMOBILES
    WHERE NAME = 'SMITH' AND AUTOMOBILES.MAKE = 'FORD'
END-EXEC
```

In this example, NAME is a descriptor field in the main file PERSONNEL, whilst MAKE is a descriptor field in the file AUTOMOBILES which is coupled to the main file.

descriptor comp value

value is the descriptor value that is to be sought. value can be either a constant or the name of a variable. In the latter case, the name must be immediately preceded by a colon (':'), for example ':NAME'.

descriptor **BETWEEN** *value1* **AND** *value2* *exception*

The BETWEEN option indicates that any record in which the value of the specified descriptor lies between *value1* and *value2* will satisfy the search expression. *value1* contains the lower limit of the range, and *value2* contains the upper limit of the range.

descriptor **NOT=** *value3*

The NOT = option is used to exclude a specified value of the descriptor from a previous range (given in the BETWEEN option). *value3* must lie between *value1* and *value2* of the preceding BETWEEN option.

descriptor **NOT BETWEEN** *value3* **AND** *value4*

The NOT BETWEEN option is used to exclude a specified range of values from a previous range (given in the BETWEEN option). *value3* and *value4* must lie between *value1* and *value2* of the preceding BETWEEN option.

descriptor **IN (** *value,...* **)**

The IN option is used when the user wishes to select records in which a descriptor has any one of a number of values. The search expression is satisfied if the descriptor has any of the values specified in the list.

SETID= *'cursor-name'*

The search expression may also be a *cursor-name* referring to another FIND statement in which the ’SAVE ISN-list’ option was used. The search expression denotes the ISN list produced by the previous FIND statement. Records can be selected from this ISN list, so the search can be refined, by combining the SETID option with other search expressions.

descriptor **IS NULL**

This search expression will find all records where this descriptor has a relational NULL value (has no value at all).

descriptor **IS NOT NULL**

This search expression will find all records where this descriptor has a value.

Note:
The order of evaluation of the operators within the Adabas Search Algorithm is:

1. Evaluate the range of values and OR between values of the same descriptor.
2. Evaluate the AND operator.
3. Evaluate the new Logical OR operator (the Logical operator between different descriptors and search criteria).

Examples of Search Criteria

```
AGE = 27
AGE = 27 AND CITY = ’NY’
AGE BETWEEN 25 AND 35
CITY IN (’NY’, ’WA’, :CITA)
AGE BETWEEN 18 AND 21 OR AGE BETWEEN 65 AND 120
AGE BETWEEN :XAGE AND :YAGE AND AGE > = 18
AGE > 27 AND SETID = ’PERS’
SETID = ’PER1’ AND SETID = ’PER2’
AGE BETWEEN 18 AND 30 AND AGE NOT BETWEEN 24 AND 26
AUTOMOBILES.MAKE = ’FORD’
AGE = 30 AND AUTOMOBILES.MAKE = ’FORD’
PERSONNEL.PERSONNEL_NUMBER = AUTOMOBILES.OWNER_PERSONNEL_NUMBER AND ...
```

OPTIONS Clause

```
OPTIONS [ { AUTODBID                  } ]
        [ { DBID= database-name       } ]
          [ CIPHER= value1 ]
          [ COND-NAME= { Y } ]
          [            { N } ]
          [ HOLD [ RETURN ] ]
          [ INDEXED= { Y } ]
          [          { N } ]
          [ ISNSIZE= len ]
          [ MAXTIME= value2 ]
          [ PASSWORD= value3 ]
          [ PREFIX= prefix ]
          [ SAVE ]
          [ STATIC= { Y } ]
          [         { N } ]
          [ SUFFIX= suffix ]
```

AUTODBID Option

The AUTODBID option can be used if the file is linked to a single database. This option indicates to Adabas Native SQL that the database ID is to be taken from the data dictionary. If the file is linked to more than one database, an error message will be issued.

The AUTODBID option and the HOLD option may not be used together. This implies to Adabas Native SQL that you are attempting to update a database other than your default database. Also, AUTODBID and DBID may not be used together.

CIPHER Option

The cipher key must be specified when accessing a ciphered file. See also the previous discussion on this option for more information.

COND-NAME Option

This option applies only to COBOL programs.

If the option ’COND-NAME = Y’ is coded, the record buffer generated by Adabas Native SQL includes as level-88 entries the condition names defined in Predict.

If specified here, any value specified with the global parameter OPTIONS will be overridden.

With Cond. names in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See also the previous discussion on this option for more information.

DBID Option

This option should be used if the program accesses more than one database. The *database-name* must be defined in the data dictionary, and the data dictionary description of the database must include the file or files to be accessed.

The DBID option and the HOLD option may not be used together. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

HOLD Option

If the HOLD option is coded, the record retrieved is placed in hold status. As long as a record is in hold status, it cannot be updated or deleted by any other user.

A record that is to be updated or deleted must be in hold status unless the program is running in exclusive-control mode.

See *HOLD Logic* for more information.

The HOLD option may not be used together with the AUTODBID or DBID options. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

INDEXED Option

This option applies only to COBOL programs.

If the INDEXED option is specified, all multiple-value fields and periodic groups are generated with the ’INDEXED BY’ keywords. The name of the index is taken from the Predict additional field attribute 3GL specification, Indexed by. If no index name is specified here, the name of the multiple-value field or periodic group is used, prefixed with ’I-’.

Any specification here will override any setting of the global parameter OPTIONS.

Indexed by in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option and section *Generate COBOL Copy Code* in the *Predict Administration Manual* for more information.

ISNSIZE Option

The ISNSIZE parameter defines the maximum number of ISNs that can be stored in the ISN buffer. If the number of records that satisfy the selection criterion exceeds ISNSIZE, the excess ISNs are stored by Adabas and retrieved automatically when required. This process is transparent to the programmer. See the previous discussion on this option for more information.

MAXTIME Option

Limits the time of executing the FIND statement. See the previous discussion on this option for more information.

PASSWORD Option

The password must be specified in each Adabas Native SQL statement that accesses a password-protected file or a file that is protected by security by value, unless it is specified globally in the password clause of the CONNECT statement.

See the previous discussion on this option for more information

PREFIX Option

If the option ’PREFIX = *prefix*’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name prefix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

SAVE Option

Use this option if you need to retain the entire ISN list. The saved ISN list can be used later in COMPARE, FIND and SORT statements. The saved ISN list is discarded when:

- A further Adabas Native SQL statement that creates another ISN list with the same name (same command-ID) is executed, or:
- An Adabas Native SQL CLOSE or DBCLOSE statement is executed, or:
- The non-activity time limit or transaction time limit is exceeded.

Under these circumstances, response code 9 is returned when the next Adabas command is attempted.

A CLOSE statement must be executed to release the ISN list after every statement that generates an ISN list (COMPARE, FIND, FIND COUPLED and SORT). If the CLOSE statement is not executed, large amounts of storage will be occupied for the remainder of the Adabas session.

STATIC Option

This option applies to PL/I programs only.

If the option ’STATIC = Y’ is coded here, all buffers generated by Adabas Native SQL will be defined as static. This will override any setting of the global parameter OPTIONS.

Static in the Predict Modify PL/I Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

SUFFIX Option

If the option ’SUFFIX = *suffix*’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name suffix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

ORDER BY Clause

```
                     { DESCENDING }
ORDER BY de1... 3    {            }
                     { ASCENDING  }
```

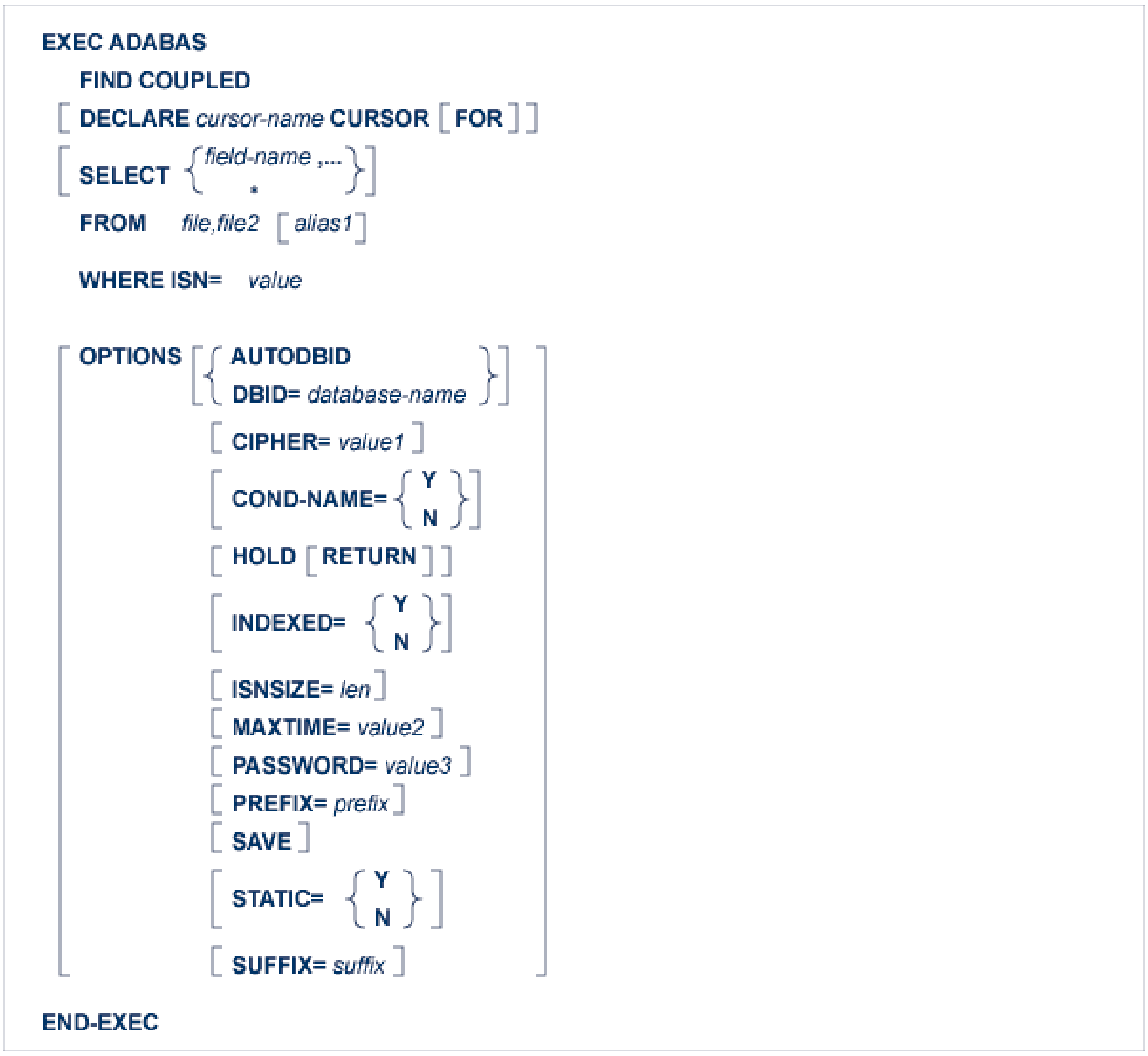
The ORDER BY clause specifies the order in which the records are retrieved.

The ISN list may be sorted on up to three descriptors in ascending or descending sequence.

A descriptor used in an ORDER BY clause may not be a member of a periodic group, nor may it be a phonetic descriptor.

The keyword DESCENDING, which may be abbreviated to DESC, specifies descending sequence, otherwise ascending sequence as default is assumed. If more than one descriptor is specified, the ASCENDING/DESCENDING option applies collectively to all of them. It is not possible to specify ascending sequence for one descriptor and descending sequence for another.

The FIND COUPLED Statement



```
EXEC ADABAS
    FIND COUPLED
  [ DECLARE cursor-name CURSOR [ FOR ] ]
  [ SELECT { field-name ,... } ]
           { *               }
    FROM      file,file2  [ alias1 ]

    WHERE ISN=   value

  [ OPTIONS [ { AUTODBID             } ]
            [ { DBID= database-name  } ]
              [ CIPHER= value1 ]
              [ COND-NAME= { Y } ]
              [            { N } ]
              [ HOLD [ RETURN ] ]
              [ INDEXED= { Y } ]
              [          { N } ]
              [ ISNSIZE= len ]
              [ MAXTIME= value2 ]
              [ PASSWORD= value3 ]
              [ PREFIX= prefix ]
              [ SAVE ]
              [ STATIC= { Y } ]
              [         { N } ]
              [ SUFFIX= suffix ]                ]
END-EXEC
```

The FIND COUPLED statement retrieves fields from a record or records coupled to a given record in another file. Specify the names of both files and the ISN of the record to which the records to be retrieved are coupled.

FIND COUPLED is normally used together with FIND. The FIND statement is used to find a record of interest (the search criterion may include fields from several files); the FIND COUPLED statement is then used to retrieve information from the record or records that are coupled to that record. If more than one record satisfied the search criterion of the original FIND statement, the FIND COUPLED statement must be repeated for each record in the ISN list returned by the FIND statement.

In general, the FIND COUPLED statement will return a list containing the ISNs of several records that are coupled to the specified record in the main file.

If all of the records found by the FIND COUPLED statement are to be processed, then the FIND COUPLED statement must include the ’DECLARE *cursor-name* CURSOR FOR’ clause and it must be followed by an OPEN/FETCH/CLOSE sequence as described from page [2004-08-24 tbd]. The fields specified in the SELECT clause are available after execution of the FETCH statement.

If, however, only the first of these records is to be processed, then the DECLARE clause may be omitted and the fields specified in the SELECT clause are available after execution of the FIND COUPLED statement. In this case, Adabas Native SQL generates executable code for the FIND COUPLED statement, which must therefore appear in the procedure division in COBOL programs.

Examples including the FIND COUPLED statement may be found in the appendices.

Note:
The examples using coupled files cannot be executed under VMS, since coupled files are not supported.

An Adabas S5 command is generated.

DECLARE Clause

```
DECLARE cursor-name CURSOR [ FOR ]
```

This clause specifies a cursor-name that identifies, or labels, the current statement. Subsequent statements can refer back to a statement that is labeled with a DECLARE clause by quoting the cursor-name. The cursor-name may be up to four characters long.

If multiple records are to be processed, the ’DECLARE *cursor-name* CURSOR FOR’ construction must be used. The keyword FOR indicates to Adabas Native SQL that the statement is used in conjunction with OPEN and FETCH statements that appear later in the program quoting the same cursor-name. If only a single record is to be processed, the DECLARE clause may be omitted.

See the previous discussion on this clause for more information.

SELECT Clause

```
       { field-name ,... }
SELECT {                 }
       {     *           }
```

The SELECT clause specifies which fields are to be retrieved from the database. All types of fields may be selected, with the exception of phonetic descriptors. The fields must be specified by their full names as defined in the data dictionary.

If the SELECT clause is omitted, then no records are processed, but other functions such as search may be performed.

If an asterisk is specified following the keyword SELECT, all the fields within the userview are read.

See the previous discussion on this clause for more information.

FROM Clause

```
FROM    file,file2 [ alias1 ]
```

This is the file list. *file1* and *file2* are Adabas file names or view names as defined in the data dictionary. The two files must be coupled. *file1* is the name of the file from which the records are to be read. *file2* is the name of the file containing the record whose ISN is specified in the WHERE clause.

alias1 is the alias associated with *file1*. If present, it is used as the name of the record buffer; otherwise, the name *file1* is used. The alias - which should be unique within the program (including linked modules) - is required if two or more Adabas Native SQL statements within the module refer to the same file. It can then be used as a qualifier in subsequent Ada, COBOL or PL/I statements that wish to refer to the fields in the respective record buffers.

The names of the coupled files are separated by a comma, but the alias - if present - is not preceded by a comma.

Example:

```
EXEC ADABAS
    FIND COUPLED
    SELECT NAME, CITY
    FROM PERSONNEL,AUTOMOBILES
    WHERE ISN = :VAR
END-EXEC
```

WHERE Clause

WHERE ISN= *value*

The WHERE clause specifies the ISN of the record in *file2* to which the records in *file1* are coupled. *value* may be a numeric constant or the name of a variable containing the ISN. If value is a variable name, it must be immediately preceded by a colon (’:’), for example ’:NAME’. The colon must not be coded following the ’=’ sign if *value* is a numeric constant, for example ’WHERE ISN = 1234’.

OPTIONS Clause

```
OPTIONS [ { AUTODBID                } ]
          { DBID= database-name     }
        [ CIPHER= value1 ]
        [ COND-NAME= { Y } ]
                     { N }
        [ HOLD [ RETURN ] ]
        [ INDEXED= { Y } ]
                   { N }
        [ ISNSIZE= len ]
        [ MAXTIME= value2 ]
        [ PASSWORD= value3 ]
        [ PREFIX= prefix ]
        [ SAVE ]
        [ STATIC= { Y } ]
                  { N }
        [ SUFFIX= suffix ]
```

AUTODBID Option

The AUTODBID option can be used if the file is linked to a single database. This option indicates to Adabas Native SQL that the database ID is to be taken from the data dictionary. If the file is linked to more than one database, an error message will be issued.

This option may not be used together with the HOLD option. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

CIPHER Option

The cipher key must be specified when accessing a ciphered file.

See the previous discussion on this option for more information.

COND-NAME Option

This option applies only to COBOL programs.

If the option ’COND-NAME = Y’ is coded, the record buffer generated by Adabas Native SQL includes the condition names defined in Predict as level-88 entries.

If specified here, any value specified with the global parameter OPTIONS will be overridden.

With Cond. names in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

DBID Option

This option should be used if the program accesses more than one database. The database-name must be defined in the data dictionary, and the data dictionary description of the database must include the file or files to be accessed.

This option may not be used together with the HOLD option. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

HOLD Option

If the HOLD option is coded, the record retrieved is placed in hold status. As long as a record is in hold status, it cannot be updated or deleted by any other user.

A record that is to be updated or deleted must be in hold status unless the program is running in exclusive-control mode. See *HOLD Logic* for more information.

The HOLD option may not be used together with the AUTODBID or DBID options. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

INDEXED Option

This option applies only to COBOL programs.

If the INDEXED option is specified, all multiple-value fields and periodic groups are generated with the ’INDEXED BY’ keywords. The name of the index is taken from Predict. If no index name is defined in the data dictionary, the name of the multiple-value field or periodic group is used, prefixed with ’I-’.

Any specification here will override any setting of the global parameter OPTIONS).

Indexed by in the Predict *Modify COBOL Defaults* screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option and section *Generate COBOL Copy Code* in the *Predict Administration Manual* for more information.

ISNSIZE Option

The ISNSIZE parameter defines the maximum number of ISNs that can be stored in the ISN buffer. If the number of records that satisfy the selection criterion exceeds ISNSIZE, the excess ISNs are stored by Adabas and retrieved automatically when required. This process is transparent to the programmer. See the previous discussion on this option for more information.

MAXTIME Option

This option is used to limit the time of executing the FIND statement. The user may specify a number or variable containing a number.

PASSWORD Option

The password must be specified in each Adabas Native SQL statement that accesses a password-protected file or a file that is protected by security by value, unless it is specified globally in the CONNECT statement. See the previous discussion on this option for more information.

PREFIX Option

If the option ’PREFIX = *prefix*’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name prefix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

SAVE Option

Use this option if you need to retain the entire ISN list. The saved ISN list can be used later in COMPARE, FIND and SORT statements. The saved ISN list is discarded when:

- A further Adabas Native SQL statement that creates another ISN list with the same name (same command-ID) is executed, or:
- An Adabas Native SQL CLOSE or DBCLOSE statement is executed, or:
- The non-activity time limit or transaction time limit is exceeded.

Under these circumstances, response code 9 is returned when the next Adabas command is attempted.

A CLOSE statement must be executed to release the ISN list after every statement that generates an ISN list (COMPARE, FIND, FIND COUPLED and SORT). If the CLOSE statement is not executed, large amounts of storage will be occupied for the remainder of the Adabas session.

STATIC Option

This option applies to PL/I programs only.

If the option ’STATIC = Y’ is coded here, all buffers generated by Adabas Native SQL will be defined as static. This will override any setting of the global parameter OPTIONS.

Static in the Predict Modify PL/I Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

SUFFIX Option

If the option ’SUFFIX = *suffix*’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name suffix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

The GENERATE Statement

```
EXEC ADABAS
   GENERATE
   FILE= file-id

 [ OPTIONS
     [ COND-NAME= { Y } ]
                  { N }
     [ INDEXED=   { Y } ]
                  { N }
     [ PREFIX= prefix ]
     [ START-LEVEL= start-level ]
     [ STATIC=    { Y } ]
                  { N }
     [ SUFFIX= suffix ]                ]
END-EXEC
```

This statement is used to generate copy code you wish to include in your program or to regenerate copy code for which the Predict dictionary definitions have been modified after generation.

This Adabas Native SQL statement provides a subset of the facilities provided by the Predict GENERATE statement. If you require any of the extended range of facilities, use the Predict preprocessor.

If more than one preprocessor is used, they must be used in the following order:

- Predict
- Adabas Native SQL
- CICS.

The *start-level* is in the range 1..40.

See the description of the analogous GENERATE command in chapter *The Preprocessor* of the *Predict Administration Manual* for more information.

The HISTOGRAM Statement

```
EXEC ADABAS
    HISTOGRAM
  [ DECLARE cursor-name CURSOR [ FOR ] ]
  [ SELECT [ field-name [(i)] ] [ COUNT(*) ] ]
    FROM   file [ alias ]

  [            { field-name [(i)]  BETWEEN value1 AND value2 } ]
  [            {                   { {GE | >=} }              } ]
  [ WHERE      {                   { {GT | >}  }              } ]
  [            { field-name [(i)]  { {LE | <=} }   value3     } ]
  [            {                   { {LT | <}  }              } ]

  [ OPTIONS                                  ]
  [ [ { AUTODBID               } ]           ]
  [ [ { DBID= database-name    } ]           ]
  [   [ COND-NAME= { Y } ]                   ]
  [   [            { N } ]                   ]
  [   [ PASSWORD= value4 ]                   ]
  [   [ PREFIX= prefix ]                     ]
  [   [ STATIC= { Y } ]                      ]
  [   [         { N } ]                      ]
  [   [ SUFFIX= suffix ]                     ]

    ORDER BY field-name1  { DESCENDING }
                          { ASCENDING  }

    GROUP BY field-name [(i)]

END-EXEC
```

The HISTOGRAM statement is used to determine the values present for a given descriptor in the specified file. The values are returned in ascending or descending sequence. Along with each descriptor value, Adabas Native SQL indicates the number of records that contain that value. This information is read from the Associator inverted lists; no access is made to Data Storage.

The HISTOGRAM statement will normally be used to read many descriptor values in sequence. In this case, the ’DECLARE *cursor-name* CURSOR FOR’ clause must be coded, and the HISTOGRAM statement must be followed by the OPEN and FETCH statements. The descriptor field specified in the SELECT clause and also the QUANTITY, i.e., the number of records with that descriptor value, are available after execution of the FETCH statement.

If only the first (lowest) descriptor value that is greater than or equal to the specified starting value is required, the DECLARE clause may be omitted. The descriptor field specified in the SELECT clause is available directly after execution of the HISTOGRAM statement.

An Adabas L9 command is generated.

DECLARE Clause

```
DECLARE cursor-name CURSOR [FOR]
```

This clause specifies a cursor-name that identifies, or labels, the current statement. Subsequent statements can refer back to a statement that is labeled with a DECLARE clause by quoting the cursor-name. The cursor-name may be up to four characters long.

If multiple descriptor values are to be processed, the ’DECLARE *cursor-name* CURSOR FOR’ construction must be used. The keyword FOR indicates to Adabas Native SQL that the statement is used in conjunction with OPEN and FETCH statements that appear later in the program quoting the same cursor-name. If only a single descriptor value is to be processed, the DECLARE clause may be omitted.

See the previous discussion on this clause for more information.

SELECT Clause

```
SELECT [field-name [(i)]] [COUNT(*)]
```

field-name is the name of the descriptor for which the values are to be returned. *field-name* must be the same descriptor as in the GROUP BY clause. The values are provided in ascending or descending order. Null values are not returned for descriptors that were defined with the null value suppression option.

Use the COUNT(*) option to find out how many records contain each descriptor value. The count will then be returned in the variable QUANTITY attached to the record buffer. Note that the string ’COUNT(*)’ must be written without spaces.

If the descriptor is a field within a periodic group, the field ’ISN’ (Ada, COBOL or PL/I unless the global parameter ’ABORT .’ is specified) or ’SQLISN’ (Ada, COBOL or PL/I if the global parameter ’ABORT .’ is specified; also FORTRAN) will contain the number of the occurrence in which the returned value is located.

FROM Clause

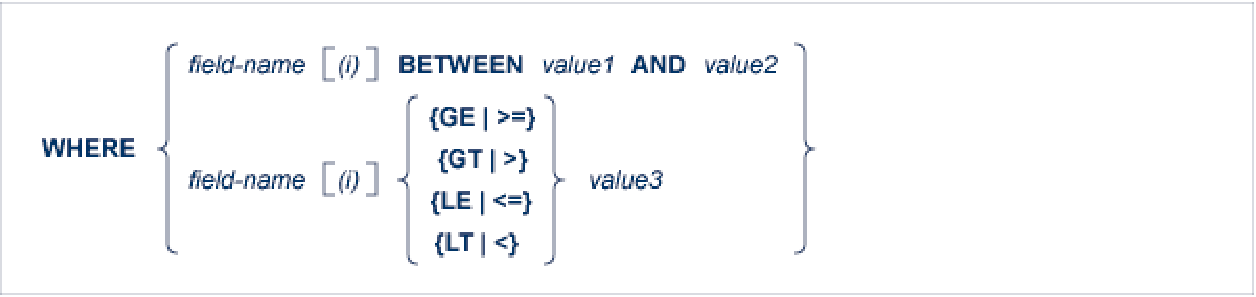
```
FROM file [alias]
```

The FROM clause specifies the file from which the descriptor values are to be retrieved.

file is the Adabas file name or view name as defined in the data dictionary. The *alias*, if present, is used as the name of the record buffer; otherwise, the name *file* is used.

See the previous discussion on this clause for more information.

WHERE Clause



The range of descriptor values to be read may be restricted by coding an appropriate WHERE clause.

Starting and ending values may be specified using the ’WHERE *field-name* BETWEEN *value1* AND *value2*’ option. *value1* represents the value with which reading is to begin and *value2* represents the value with which reading is to end.

The following restriction applies only if ADA-VERSION=62 in the global OPTIONS statement or if the ADA-VERSION= *paramete*r is omitted: to specify only a starting value, use the ’*field-name* >= *value3*’ or ’*field-name* GE *value3*’ option for ascending order or ’*field-name* <= *value3*’ or ’*field-name* LE *value3*’ for descending order (if the Adabas version allows it). *value3* represents the value with which reading is to begin.

In the case of ADA-VERSION=71 in the global OPTIONS statement, all the comparator operators can be used for both ascending and descending order.

The *field-name* must be the descriptor specified in the GROUP BY clause. If the starting value *(value1*, *value3*) is not contained in the file, the next higher value in the list will be used. If no higher value exists, an end-of-file condition will result. *value1*, *value2* and *value3* may be constants or the names of variables containing the values. If they are variable names, they must be immediately preceded by colons (’:’), for example ’:NAME’.

field-name (i) is a reference to an occurrence within a periodic group.

Note:
If a prefix or suffix is used for a field-name specified in the data dictionary, you may not use the BETWEEN option if ADA-VERSION=62 in the global OPTIONS statement or if the ADA-VERSION= parameter is omitted.

OPTIONS Clause

```
OPTIONS
[ { AUTODBID               } ]
[ { DBID= database-name    } ]
  [ COND-NAME= { Y } ]
  [            { N } ]
  [ PASSWORD= value4 ]
  [ PREFIX= prefix ]
  [ STATIC= { Y } ]
  [         { N } ]
  [ SUFFIX= suffix ]
```

AUTODBID Option

The AUTODBID option can be used if the file is linked to a single database. This option indicates to Adabas Native SQL that the database ID is to be taken from the data dictionary. If the file is linked to more than one database, an error message will be issued.

COND-NAME Option

This option applies only to COBOL programs.

If the option ’COND-NAME = Y’ is coded, the record buffer generated by Adabas Native SQL includes the condition names defined in PREDICT as level-88 entries.

If specified here, any value specified with the global parameter OPTIONS) will be overridden.

With Cond. names in the PREDICT Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this optin for more information.

DBID Option

This option should be used if the program accesses more than one database. The *database-name* must be defined in the data dictionary, and the data dictionary description of the database must include the file or files to be accessed.

PASSWORD Option

The password must be specified in each Adabas Native SQL statement that accesses a password-protected file or a file that is protected by security by value, unless it is specified globally in the CONNECT statement.

PREFIX Option

If the option ’PREFIX = prefix’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS) or taken from PREDICT.

Field name prefix in the PREDICT Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

STATIC Option

This option applies to PL/I programs only.

If the option ’STATIC = Y’ is coded here, all buffers generated by Adabas Native SQL will be defined as static. This will override any setting of the global parameter OPTIONS.

Static in the PREDICT Modify PL/I Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

SUFFIX Option

If the option ’SUFFIX = *suffix*’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS or taken from PREDICT.

Field name suffix in the PREDICT Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

ORDER BY Clause

```
                        ⎧ DESCENDING ⎫
ORDER BY  field-name1   ⎨            ⎬
                        ⎩ ASCENDING  ⎭
```

The *field-name1* parameter specifies the descriptor that is to control the reading sequence. Note that the descriptor specified may not be a member of a periodic group, nor may it be a phonetic field.

If the descriptor was defined with the null value suppression option, records having a null value in the descriptor field will not be read.

If the WHERE clause is coded, the descriptor field used in the WHERE clause must be the same as the descriptor field used in the ORDER BY clause.

If DESCENDING is specified, the records are read in descending order.

Note:
The ’GE’ operator cannot be specified together with the DESCENDING keyword if ADA-VERSION=62 in the global OPTIONS statement or if the ADA-VERSION= *parameter* is omitted.

GROUP BY Clause

```
GROUP BY  field-name [(i)]
```

field-name is the descriptor for which the values are to be returned.

The descriptor may not be a phonetic descriptor or a field in a periodic group. The use of descriptors defined as multiple-value fields is not recommended.

If the SELECT, WHERE and/or ORDER BY clauses are coded, the *field-name* used in these clauses must be the same as the *field-name* used in the GROUP BY clause.

The HOLD Statement

```
EXEC ADABAS
    HOLD  cursor-name
  [ OPTIONS RETURN ]

END-EXEC
```

This statement is used to place a record in hold status. This reserves the record for subsequent updating or deleting, preventing other users from updating the record until it is released by a COMMIT WORK, RELEASE or ROLLBACK WORK statement. This statement should be used after reading the record and before updating or deleting it, unless the read statement itself included the HOLD option. The *cursor-name* identifies the statement that retrieved the record.

OPTIONS Clause

```
OPTIONS RETURN
```

If the RETURN option is coded and the record is already being held for another program, the value 145 will be returned in the response-code. This will cause an error printout followed by an ABEND unless a user-written response code interpretation routine is provided.

See description of the ABORT parameter for more information.

If the RETURN option is not coded and the record is being held for another program, the program will be suspended until the record is released.

In many applications, it is preferable to specify the HOLD option in the READ or FIND statement rather than to use a separate HOLD statement.

This statement generates an Adabas HI command.

The INSERT Statement

```
EXEC ADABAS
    INSERT
        INTO  file [alias]
    [ DECLARE  cursor-name   CURSOR ]
    [ WHERE [ ISN=  value ] [CURRENT OF  cursor-name1 ] ]
    [ assignments ]

    [ OPTIONS                              ]
    [   [ PASSWORD=  value1 ]              ]
    [   [ CIPHER=  value2 ]                ]
    [   [ PREFIX=  value3 ]                ]
    [   [ SUFFIX=  value4 ]                ]

END-EXEC
```

The INSERT statement adds a new record to the Adabas file.

When an attempt is made to add a new record with one or more fields that have been defined as unique descriptors, response code 198 will be returned if a record having the same descriptor value as the new record already exists in the file. This will cause an error print-out (response code 98 in VAX, otherwise 198) followed by an ABEND unless the user provides an alternative response code interpretation routine. See description of the ABORT parameter on page .

This statement generates an Adabas N1 command, or an N2 command if the ’WHERE ISN’ clause is coded.

INTO Clause

```
INTO  file [alias]
```

file is the Adabas file name or view name as defined in the data dictionary. The *alias*, if present, is used as the name of the record buffer; otherwise, the name *file* is used. The alias, which should be unique, is required if two or more Adabas Native SQL statements within the module refer to the same file. It can then be used as a qualifier in subsequent Ada, COBOL or PL/I statements that refer to the fields in the record buffer.

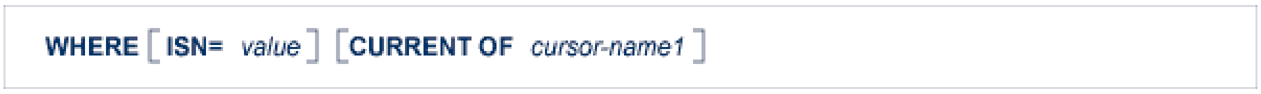
DECLARE Clause

```
DECLARE  cursor-name   CURSOR
```

This clause specifies a cursor-name that identifies, or labels, the current statement. Subsequent statements can refer back to a statement that is labeled with a DECLARE clause by quoting the cursor-name. The cursor-name may be up to four characters long.

The *cursor-name* specified in the DECLARE clause is used by Adabas as the command-ID. Adabas avoids re-translating the format buffer when it recognizes a command-ID that has been used before, so using this clause can improve performance, particularly if the ’WHERE ISN’ option is used.

WHERE Clause



Use one or both options. If both options are used, they can be specified in any order.

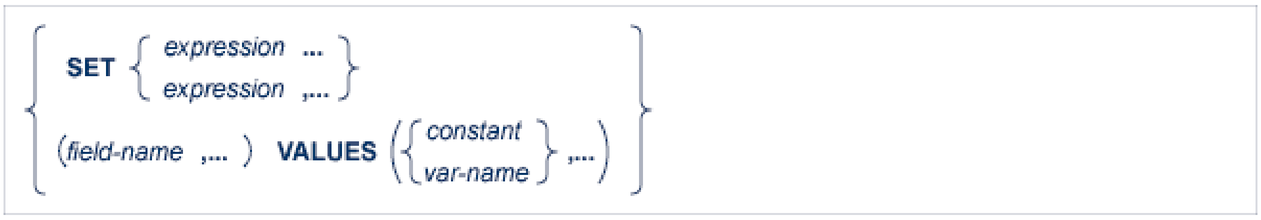
The ’WHERE ISN=*value*’ option is used if you wish to specify the ISN of the record to be added (user-ISN option). *value* may be either a constant or the name of a variable containing the ISN. The ISN must lie between 1 and the maximum ISN value that was defined for the file. If *value* is a variable name, it must be immediately preceded by a colon (’:’), for example ’:NAME’. The colon must not be coded following the ’=’ sign if *value* is a numeric constant, for example ’WHERE ISN = 1234’. If a record with the specified ISN already exists, the value 113 will be returned in the response-code. The ’DECLARE *cursor-name* CURSOR’ clause should be used if ’WHERE ISN=value’ is coded, in order to avoid unnecessary format buffer translations.

If the option ’WHERE CURRENT OF *cursor-name1*’ is used and no assignments are used, it is not necessary to build a new record buffer; the existing record buffer is written to the database. This can improve performance.

If the WHERE clause is omitted, the ISN of the new record will be allocated automatically by Adabas.

Assignments

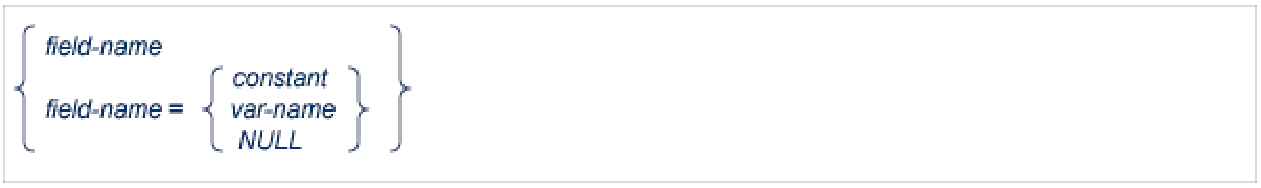
Note:
If the option ’WHERE CURRENT OF *cursor-name1*’ is used, multiple-value fields and periodic groups are not supported in the assignments. If multiple-value fields or periodic groups are present, all assignments must be made before the statement. No assignments are permitted within the statement.



This clause specifies the fields to be written to the record and, optionally, the values to be assigned to them. The expressions may be separated by blanks (spaces) or commas.

A new record buffer is built if this clause is coded. Avoiding this clause may improve performance, because the record buffer of the statement specified in the CURRENT OF clause is used.

expression



field-name denotes the name of the elementary field. This is the full field name as defined in the data dictionary. If necessary, the *field-name* can be subscripted to select the required field from a multiple-value field, from a periodic group, or from a multiple-value field within a periodic group.

Note:
field-name can be a multiple-value or a part of a periodic group, but in this case an index must be specified within parentheses. For a multiple-value within a periodic group the user should move the value by himself before the INSERT/UPDATE statement.

The option ’SET *field-name*’ is used when the required value has already been assigned to the field in the record buffer by means of normal Ada, COBOL, FORTRAN or PL/I statements.

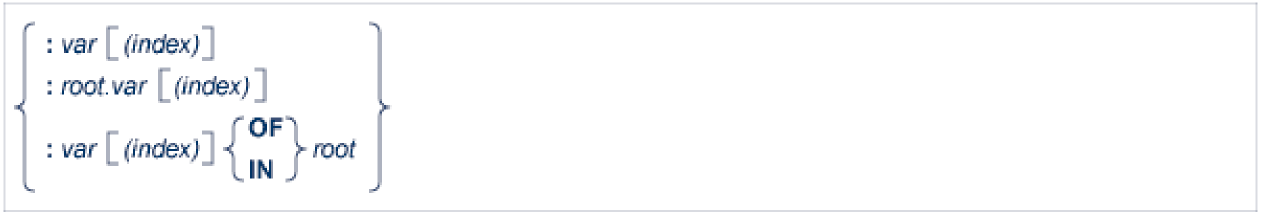
The option ’SET *field-name* = *constant*’ or ’SET field-name = *var-name*’ is used to specify the value to be assigned to the field.

constant denotes a constant (literal) value and *var-name* denotes the name of a variable defined in the Ada, COBOL, FORTRAN or PL/I program, which must be preceded by a colon.

If NULL is specified, Adabas Native SQL will move -1 (x’FFFF’) to the Null field indicator of the specified field in the Record buffer used for updating the file.

If the user uses the SET clause and specifies a real value or a variable for a field which has a Null value indicator, Adabas Native SQL will automatically reset the Null field indicator of that field. If the user does not specify the SET clause, but initiates the fields in the Record buffer by himself, he should also reset or turn on the Null field indicator.

var-name



If the variable name is unique within the program, it can be specified as :*var*. Otherwise, it should be made unique by preceding it by root, a higher-level data name (qualifier) in the data structure hierarchy. Both the COBOL-type construction (:*var* OF *root* or :*var* IN *root*) and the PL/I-type construction (*:root.var*) are valid in Ada, COBOL and PL/I programs.

Both the ’SET *field-name*’ option and the ’SET *field-name* = *data*’ option can be used in the same SET clause.

The optional *index* may be an integer constant or the name of a variable preceded by a colon. Note that blanks (spaces) are not allowed between the :*var* and the parenthesis preceding the *index*.

OPTIONS Clause

```
OPTIONS
  [ PASSWORD= value1 ]
  [ CIPHER= value2 ]
  [ PREFIX= value3 ]
  [ SUFFIX= value4 ]
```

PASSWORD Option

The password must be specified in each Adabas Native SQL statement that accesses a password-protected file or a file that is protected by security by value, unless it is specified globally in the CONNECT statement.

CIPHER Option

The cipher key must be specified when accessing a ciphered file. See the previous discussion on this option for more information.

PREFIX Option

If the option ’PREFIX = *prefix*’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name prefix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

SUFFIX Option

If the option ’SUFFIX = *suffix*’ is coded, the field names generated for the record buffer will include the specified suffix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name suffix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

Example 1: Ada

```
type RECORD_BUFPERS is
     record
     SALARY: STRING (...
      .
      .
      .
     end record;
FINANCE: RECORD_BUFPERS;

FINANCE.OIL_CREDIT_2:= "0001000";
EXEC ADABAS
     INSERT
       INTO FINANCE
         SET PERSONNEL-NUMBER = "005333756"
         OIL_CREDIT(3) = "0002000"
         OIL_CREDIT(1) = "0001000"
         INSURANCE_COMPANY(1(1)) = "AAA "
         OIL_CREDIT(2)
END-EXEC
```

Example 2: COBOL

```
01 REC
   02 SALARY ......
   02 AGE ......
   02 PERSON-NAME ......hg
    .
    .
    .
MOVE 1000 TO OIL-CREDIT-2
EXEC ADABAS
    INSERT
    INTO FINANCE
    SET PERSONNEL-NUMBER =  5333756
         OIL-CREDIT(3) = 2000
         OIL-CREDIT(1) = 1000
         INSURANCE-COMPANY(1(1)) = ’AAA’
         OIL-CREDIT(2)
END-EXEC
```

Example 3: FORTRAN

```
CHARACTER* 8 PERBER
CHARACTER* 7 OCRE1
CHARACTER* 7 OCRE3
CHARACTER*25 INCOM (00001 , 00001)
CHARACTER* 7 OCRE2
CHARACTER* 14002 FINNCE
.....
OCRE2 = ’0001000’

EXEC ADABAS
    INSERT
    INTO FINANCE
    SET PERSONNEL-NUMBER = ’005333756’
        OIL-CREDIT(1) = ’0002000’
        OIL-CREDIT(3) = ’0001000’
        INSURANCE-COMPANY(1(1)) = ’AAA’
        OIL-CREDIT (2)
END-EXEC
```

Note:
Synonyms are assumed to be defined in the data dictionary as shown in Appendix B, and truncation is assumed to occur in the middle of the word. (The maximum length of names is operating-system dependent.)

The field FINNCE encompasses all other fields and is the equivalent of the record buffer in Ada, COBOL and PL/I.

Example 4: PL/I

```
DCL 01 REC,
      02 SALARY ......,
      02 AGE ......,
      02 PERSON_NAME ......;
    .
    .
    .
OIL_CREDIT_2 = 1000;
EXEC ADABAS
     INSERT
     INTO FINANCE
     SET PERSONNEL-NUMBER = 5333756
     OIL-CREDIT(3) = 2000
     OIL-CREDIT(1) = 1000
     INSURANCE-COMPANY(1(1)) = ’AAA’
     OIL-CREDIT(2)
END-EXEC
```

The OPEN Statement

```
EXEC ADABAS
   OPEN cursor-name
END-EXEC
```

This statement is part of the OPEN/FETCH/CLOSE sequence that is used for processing multiple records.

The OPEN statement processes the parameters defined in the WHERE clause of the statement referenced by *cursor-name* and then issues the actual Adabas command if necessary. Once the OPEN statement has been executed, the contents of the WHERE clause are not re-examined. Therefore, changes to the variables in a WHERE clause will not have any effect until the OPEN statement is re-executed.

In the case of the HISTOGRAM, READ LOGICAL and READ PHYSICAL SEQUENCE statements, the OPEN statement does nothing more than to initialize the variables for the executable Adabas commands. For the COMPARE, FIND, FIND COUPLED and SORT statements, the OPEN statement initializes the variables and also executes the command (FIND, SORT,...) that produces the ISN list. No records are actually fetched from the file until the FETCH statement is executed.

When used in conjunction with a COMPARE, FIND, FIND COUPLED or SORT statement, the OPEN statement puts the ISN quantity in the record buffer. Thus the number of records can be found before executing the first FETCH statement.

The READ ISN Statement

```
EXEC ADABAS
    READ ISN
  [ DECLARE cursor-name CURSOR ]
  [ SELECT { field-name ,... } ]
           {      *          }
    FROM   file [ alias ]
    WHERE  { ISN= value               }
           { CURRENT OF cursor-name1  }
  [ OPTIONS [ { AUTODBID              } ]  ]
  [         [ { DBID= database-name   } ]  ]
  [           [ CIPHER= value1 ]           ]
  [           [ COND-NAME= { Y } ]         ]
  [           [            { N } ]         ]
  [           [ HOLD [ RETURN ] ]          ]
  [           [ INDEXED= { Y } ]           ]
  [           [          { N } ]           ]
  [           [ PASSWORD= value2 ]         ]
  [           [ PREFIX= prefix ]           ]
  [           [ SAVE ]                     ]
  [           [ STATIC= { Y } ]            ]
  [           [         { N } ]            ]
  [           [ SUFFIX= suffix ]           ]
END-EXEC
```

The READ ISN statement is used to read from a file a single record whose ISN is known, or the first record whose ISN is greater than a specified value.

This statement causes an Adabas L1 command to be generated, or an L4 command if the HOLD option is coded.

DECLARE Clause

```
DECLARE cursor-name CURSOR
```

This clause specifies a cursor-name that identifies, or labels, the current statement. Subsequent statements can refer back to a statement that is labeled with a DECLARE clause by quoting the cursor-name. The cursor-name may be up to four characters long. See the previous discussion on this clause for more information.

SELECT Clause

```
SELECT { field-name ,... }
       {      *          }
```

The SELECT clause specifies which fields are to be retrieved from the database. All types of fields may be selected, with the exception of subdescriptors, superdescriptors and phonetic descriptors. The fields must be specified by their full names as defined in the data dictionary.

If the SELECT clause is omitted, then no records are processed, but other functions such as search may be performed.

If an asterisk ('*') is specified following the keyword 'SELECT', all the fields within the userview are read.

See the previous discussion on this clause for more information.

FROM Clause

```
FROM   file [alias]
```

The FROM clause specifies the file from which data are to be retrieved. It is used together with the SELECT clause to generate the record buffer and to control the retrieval of information from the database.

file is the Adabas file name or view name as defined in the data dictionary. The *alias*, if present, is used as the name of the record buffer; otherwise, the name *file* is used.

See the previous discussion on this clause for more information.

WHERE Clause

```
WHERE { ISN=  value               }
      { CURRENT OF  cursor-name1  }
```

The WHERE clause is used to specify the ISN of the record to be read. If the SEQUENCE option is not specified, an error (response-code = 113) will result if the file does not contain a record with this ISN. If the SEQUENCE option is specified and the file does not contain a record with the given ISN, then the record with the next higher ISN will be read, or end-of-file will be signaled if there is no such record.

value can be a constant or the name of a variable. If *value* is a variable name, it must be immediately preceded by a colon (':'), for example ':NAME'. Note that the colon is not part of the '=' sign that follows the 'ISN' keyword.

If the programmer wishes Adabas Native SQL to use the ISN of a record found by a previous statement, he should use the 'CURRENT OF' option, specifying the *cursor-name* of that statement.

OPTIONS Clause

```
OPTIONS [ { AUTODBID               } ]
        [ { DBID= database-name    } ]
          [ CIPHER= value1 ]
          [ COND-NAME= { Y } ]
          [            { N } ]
          [ HOLD [ RETURN ] ]
          [ INDEXED= { Y } ]
          [          { N } ]
          [ PASSWORD= value2 ]
          [ PREFIX= prefix ]
          [ SAVE ]
          [ STATIC= { Y } ]
          [         { N } ]
          [ SUFFIX= suffix ]
```

AUTODBID Option

The AUTODBID option can be used if the file is linked to a single database. This option indicates to Adabas Native SQL that the database ID is to be taken from the data dictionary. If the file is linked to more than one database, an error message will be issued.

This option may not be used together with the HOLD option. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

CIPHER Option

The cipher key must be specified when accessing a ciphered file.

See the previous discussion on this option for more information.

COND-NAME Option

This option applies only to COBOL programs.

If the option 'COND-NAME = Y' is coded, the record buffer generated by Adabas Native SQL includes the condition names defined in Predict as level-88 entries.

If specified here, any value specified with the global parameter OPTIONS will be overridden.

With Cond. names in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

DBID Option

This option should be used if the program accesses more than one database. The *database-name* must be defined in the data dictionary, and the data dictionary description of the database must include the file or files to be accessed.

This option may not be used together with the HOLD option. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

HOLD Option

If the HOLD option is coded, the record retrieved is placed in hold status. As long as a record is in hold status, it cannot be updated or deleted by any other user.

A record that is to be updated or deleted must be in hold status unless the program is running in exclusive-control mode.

The HOLD option may not be used together with the AUTODBID or DBID options. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

See *HOLD Logic* for more information.

INDEXED Option

This option applies only to COBOL programs.

If the INDEXED option is specified, all multiple-value fields and periodic groups are generated with the ’INDEXED BY’ keywords. The name of the index is taken from Predict. If no index name is defined in the data dictionary, the name of the multiple-value field or periodic group is used, prefixed with ’I-’.

Any specification here will override any setting of the global parameter OPTIONS.

Indexed by in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option and section *Generate COBOL Copy Code* in the *Predict Administration Manual* for more information.

PASSWORD Option

The password must be specified in each Adabas Native SQL statement that accesses a password-protected file or a file that is protected by security by value, unless it is specified globally in the CONNECT statement.

See the previous discussion on this option for more information.

PREFIX Option

If the option ’PREFIX = *prefix*’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name prefix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

SEQUENCE Option

If this option is coded, the record with the specified ISN or the next higher ISN is read. The ISN of the record that was read is returned in the field ’ISN’, which is appended to every record buffer. If the file does not contain a record having an ISN higher than the specified ISN, end-of-file is signaled. Therefore, when using this option, the flag ADACODE (Ada, COBOL, PL/I) or SQLCOD (FORTRAN) should be checked for end-of-file status.

If this option is not specified, the record with the specified ISN is read. If the file does not contain a record having the specified ISN, an error is reported (response-code = 113). This causes the program to terminate unless a user-written response code interpretation routine is provided.

See description of the ABORT parameter on page .

STATIC Option

This option applies to PL/I programs only.

If the option ’STATIC = Y’ is coded here, all buffers generated by Adabas Native SQL will be defined as static. This will override any setting of the global parameter OPTIONS.

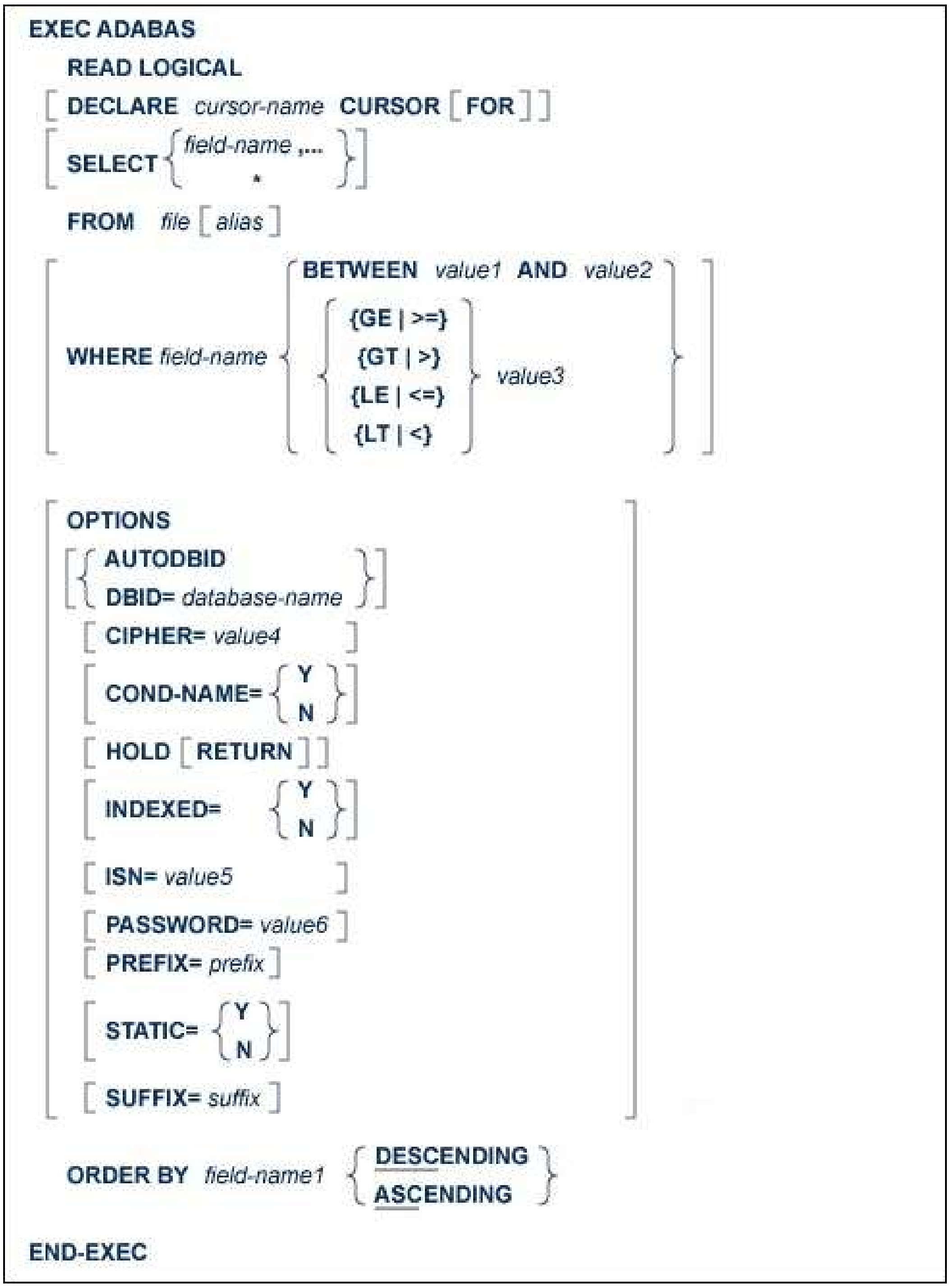
Static in the Predict Modify PL/I Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

SUFFIX Option

If the option ’SUFFIX = *suffix*’ is coded, the field names generated for the record buffer will include the specified suffix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name suffix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

The READ LOGICAL Statement



```
EXEC ADABAS
   READ LOGICAL
  [ DECLARE cursor-name CURSOR [ FOR ] ]
  [ SELECT { field-name ,...  } ]
           {      *           }
   FROM  file [ alias ]
  [                  { BETWEEN value1 AND value2   } ]
  [                  {   { {GE | >=} }             } ]
  [ WHERE field-name {   { {GT | >}  }  value3     } ]
  [                  {   { {LE | <=} }             } ]
  [                  {   { {LT | <}  }             } ]

  [ OPTIONS
   [{ AUTODBID               }]
   [{ DBID= database-name    }]
    [ CIPHER= value4          ]
    [ COND-NAME= { Y } ]
    [            { N } ]
    [ HOLD [ RETURN ] ]
    [ INDEXED=  { Y } ]
    [           { N } ]
    [ ISN= value5             ]
    [ PASSWORD= value6 ]
    [ PREFIX= prefix ]
    [ STATIC= { Y } ]
    [         { N } ]
    [ SUFFIX= suffix ]                                  ]
   ORDER BY  field-name1  { DESCENDING }
                          { ASCENDING  }
END-EXEC
```

Note: The BETWEEN clause only applies if ADA-VERSION=71 in the global OPTIONS statement.

The READ LOGICAL statement is used to read a file, or portion thereof, in logical sequential order based on the ascending or descending sequence of the values for a given descriptor.

This statement will normally be used to read many records in logical sequence. In this case, the ’DECLARE *cursor-name* CURSOR FOR’ clause must be coded, and the READ LOGICAL statement must be followed by the OPEN and FETCH statements. The fields specified in the SELECT clause are available after execution of the FETCH statement.

If only the first record in the file is required, the DECLARE clause may be omitted and the fields specified in the SELECT clause are available directly after execution of the READ LOGICAL statement.

This statement causes an Adabas L3 command to be generated, or an L6 command if the HOLD option is coded.

DECLARE Clause

```
DECLARE cursor-name CURSOR [FOR]
```

This clause specifies a cursor-name that identifies, or labels, the current statement. Subsequent statements can refer back to a statement that is labeled with a DECLARE clause by quoting the cursor-name. The cursor-name may be up to four characters long.

If multiple records are to be processed, the ’DECLARE *cursor-name* CURSOR FOR’ construction must be used. The keyword ’FOR’ indicates to Adabas Native SQL that the statement is used in conjunction with OPEN and FETCH statements that appear later in the program quoting the same cursor-name. If only a single record is to be processed, the DECLARE clause may be omitted.

SELECT Clause

```
SELECT { field-name ,... }
       {      *         }
```

The SELECT clause specifies which fields are to be retrieved from the database. All types of fields may be selected, with the exception of subdescriptors, superdescriptors and phonetic descriptors. The fields must be specified by their full names as defined in the data dictionary.

If the SELECT clause is omitted, then no records are processed, but other functions such as search may be performed.

If an asterisk is specified following the keyword ’SELECT’, all the fields in the userview are read.

See the previous discussion on this clause for more information.

FROM Clause

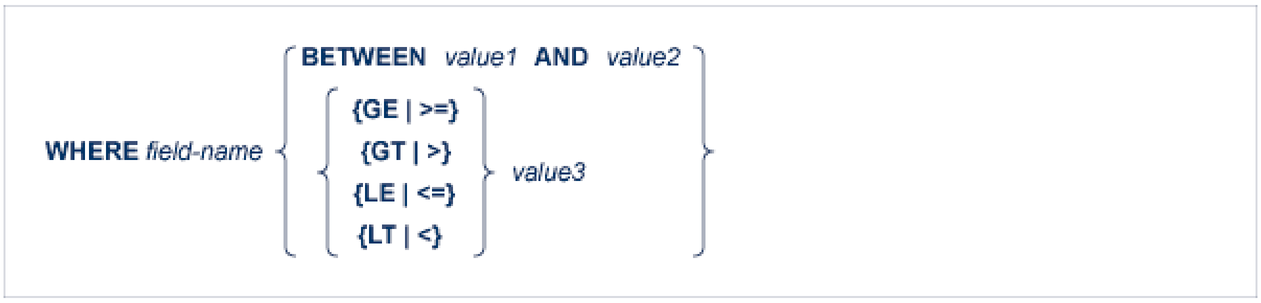
```
FROM  file [alias]
```

The FROM clause specifies the file from which data are to be retrieved. It is used together with the SELECT clause to generate the record buffer and to control the retrieval of information from the database.

file is the Adabas file name or view name as defined in the data dictionary. The *alias*, if present, is used as the name of the record buffer; otherwise, the name *file* is used.

See the previous discussion on this clause for more information.

WHERE Clause



The records may be read starting from a particular descriptor value by using the WHERE clause, where *value* represents the value from which reading is to begin. *field-name1* must be the name of the descriptor specified in the ORDER BY clause (see below).

If the starting value is not found in the file, the next higher value in the file will be used for ascending sequence. If no higher value exists, an end-of-file condition (in ADA, COBOL and PL/I programs: ADACODE = 3; in FORTRAN programs: SQLCOD = 3) will result. *value* may be a constant or the name of a variable.

If *value1*, *value2* or *value3* is a variable name, it must be immediately preceded by a colon (’:’), for example ’:NAME’.

The BETWEEN clause only applies when Adabas Version 7.1 or higher is used and the ADA-VERSION parameter in the global OPTIONS statement is set to 71 or when Adabas Version 3.1 or higher in OpenVMS is used.

OPTIONS Clause

```
OPTIONS
    [ { AUTODBID                }  ]
    [ { DBID= database-name     }  ]
      [ CIPHER= value4 ]
      [ COND-NAME= { Y } ]
      [            { N } ]
      [ HOLD [ RETURN ] ]
      [ INDEXED= { Y } ]
      [          { N } ]
      [ ISN= value5 ]
      [ PASSWORD= value6 ]
      [ PREFIX= prefix ]
      [ STATIC= { Y } ]
      [         { N } ]
      [ SUFFIX= suffix ]
```

AUTODBID Option

This option can be used if the file is linked to a single database. This option indicates to Adabas Native SQL that the database ID is to be taken from the data dictionary. If the file is linked to more than one database, an error message will be issued. (If the file is linked to more than one database, the DBID option should be used.)

This option may not be used together with the HOLD option. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

CIPHER Option

The cipher key must be specified when accessing a ciphered file. See the previous discussion on this option for more information.

COND-NAME Option

This option applies only to COBOL programs.

If the option ’COND-NAME = Y’ is coded, the record buffer generated by Adabas Native SQL includes the condition names defined in Predict as Level-88 entries.

If specified here, any value specified with the global parameter OPTIONS will be overridden.

With Cond. names in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

DBID Option

This option should be used if the program accesses more than one database. The *database-name* must be defined in the data dictionary, and the data dictionary description of the database must include the file or files to be accessed.

This option may not be used together with the HOLD option. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

HOLD Option

If the HOLD option is coded, the record retrieved is placed in hold status. As long as a record is in hold status, it cannot be updated or deleted by any other user.

A record that is to be updated or deleted must be in hold status unless the program is running in exclusive-control mode.

See *HOLD Logic* for more information.

The HOLD option may not be used together with the AUTODBID or DBID options. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

INDEXED Option

This option applies only to COBOL programs.

If the INDEXED option is specified, all multiple-value fields and periodic groups are generated with the ’INDEXED BY’ keywords. The name of the index is taken from Predict. If no index name is defined in the data dictionary, the name of the multiple-value field or periodic group is used, prefixed with ’I-’.

Any specification here will override any setting of the global parameter OPTIONS.

Indexed by in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option and section *Generate COBOL Copy Code* in the *Predict Administration Manual* for more information.

ISN Option

The ISN parameter indicates the place within a group of records with the same descriptor value where reading is to begin. Of the records which satisfy the selection criterion *field-name1 = value* (see the WHERE clause), reading begins at the record whose ISN is greater than *value3*. If there is no record with *field-name1 = value* whose ISN is greater than *value3*, the first record with the next descriptor value *field-name1 > value* is read. If there is none, the end-of-file condition (in Ada, COBOL and PL/I programs: ADACODE=3; in FORTRAN programs: SQLCOD=3) will be set.

The ISN value is specified in the *value3* field. *value3* may be a constant or the name of a variable containing the ISN. If *value3* is a variable name, it must be immediately preceded by a colon (’:’), for example ’:NAME’.

PASSWORD Option

The password must be specified in each Adabas Native SQL statement that accesses a password-protected file or a file that is protected by security by value, unless it is specified globally in the CONNECT statement. See the previous discussion on this option for more information.

PREFIX Option

If the option ’PREFIX = *prefix*’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name prefix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

STATIC Option

This option applies to PL/I programs only.

If the option ’STATIC = Y’ is coded here, all buffers generated by Adabas Native SQL will be defined as static. This will override any setting of the global parameter OPTIONS.

Static in the Predict Modify PL/I Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

SUFFIX Option

If the option ’SUFFIX = *suffix*’ is coded, the field names generated for the record buffer will include the specified suffix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name suffix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

ORDER BY Clause

```
                         { DESCENDING }
ORDER BY  field-name1    {            }
                         { ASCENDING  }
```

The *field-name1* parameter specifies the descriptor that is to control the reading sequence. Note that the descriptor specified may not be a member of a periodic group, nor may it be a phonetic field.

If the descriptor was defined with the null value suppression option, records having a null value in the descriptor field will not be read.

If the WHERE clause is coded, the descriptor field used in the WHERE clause must be the same as the descriptor field used in the ORDER BY clause.

If DESCENDING is specified, the records are read in descending order.

Note:
If the ADA-VERSION parameter in the global OPTIONS statement is set to 62 or if the ADA-VERSION= parameter is omitted, the ’GE’ operator cannot be specified together with the DESCENDING keyword on mainframe platforms, and if the ’LE’ operator is specified, the DESCENDING keyword may be omitted on mainframe platforms.

The READ PHYSICAL SEQUENCE Statement

```
EXEC ADABAS
    READ [PHYSICAL [SEQUENCE ]]
  [ DECLARE  cursor-name  CURSOR [FOR]]
  [        { field-name ,... }]
  [ SELECT {       *         }]

    FROM   file [alias]

  [ OPTIONS                                   ]
  [ [{ AUTODBID               }]              ]
  [ [{ DBID= database-name    }]              ]
  [   [ CIPHER= value1        ]               ]
  [   [             { Y }]                    ]
  [   [ COND-NAME=  { N }]                    ]
  [   [ HOLD [RETURN]]                        ]
  [   [            { Y }]                     ]
  [   [ INDEXED=   { N }]                     ]
  [   [ ISN= value2 ]                         ]
  [   [ PASSWORD= value3 ]                    ]
  [   [ PREFIX= prefix]                       ]
  [   [          { Y }]                       ]
  [   [ STATIC=  { N }]                       ]
  [   [ SUFFIX= suffix ]                      ]

END-EXEC
```

This statement is used to read records in the sequence in which they are physically located in the data files. It does not read records in any particular logical order.

This statement may be used to read an entire file at maximum speed, since no access is required to the Associator.

This statement is normally used to read many records (possibly the entire file). In this case, the ’DECLARE *cursor-name* CURSOR FOR’ clause must be coded, and the READ PHYSICAL SEQUENCE statement must be followed by the OPEN and FETCH statements. The fields specified in the SELECT clause are available after execution of the FETCH statement.

If only the first record in the file is required, the DECLARE clause may be omitted and the fields specified in the SELECT clause are available directly after execution of the READ PHYSICAL SEQUENCE statement.

This statement causes an Adabas L2 command to be generated, or an L5 command if the HOLD option is coded.

DECLARE Clause

```
DECLARE cursor-name CURSOR [FOR]
```

This clause specifies a cursor-name that identifies, or labels, the current statement. Subsequent statements can refer back to a statement that is labeled with a DECLARE clause by quoting the cursor-name. The cursor-name may be up to four characters long.

If multiple records are to be processed, the ’DECLARE *cursor-name* CURSOR FOR’ construction must be used. The keyword ’FOR’ indicates to Adabas Native SQL that the statement is used in conjunction with OPEN and FETCH statements that appear later in the program quoting the same cursor-name. If only a single record is to be processed, the DECLARE clause may be omitted.

See the previous discussion on this clause for more information.

SELECT Clause

```
        { field-name ,... }
SELECT  {                 }
        {        *        }
```

The SELECT clause specifies which fields are to be retrieved from the database. All types of fields may be selected, with the exception of subdescriptors, superdescriptors and phonetic descriptors. The fields must be specified by their full names as defined in the data dictionary.

If this clause is omitted, no records are processed, but other functions such as search may be performed.

If an asterisk is specified following the keyword ’SELECT’, all the fields within the userview are read.

See the previous discussion on this clause for more information.

FROM Clause

```
FROM  file [alias]
```

This clause specifies the file from which data are to be retrieved. It is used together with the SELECT clause to generate the record buffer and to control the retrieval of information from the database.

file is the Adabas file name or view name as defined in the data dictionary. The *alias*, if present, is used as the name of the record buffer; otherwise, the name *file* is used. See the previous discussion on this clause for more information.

OPTIONS Clause

```
OPTIONS
   [ { AUTODBID               } ]
   [ { DBID= database-name    } ]
    [ CIPHER= value1 ]
    [ COND-NAME= { Y } ]
    [            { N } ]
    [ HOLD [ RETURN ] ]
    [ INDEXED= { Y } ]
    [          { N } ]
    [ ISN= value2 ]
    [ PASSWORD= value3 ]
    [ PREFIX= prefix ]
    [ STATIC= { Y } ]
    [         { N } ]
    [ SUFFIX= suffix ]
```

AUTODBID Option

The AUTODBID option can be used if the file is linked to a single database. This option indicates to Adabas Native SQL that the database ID is to be taken from the data dictionary. If the file is linked to more than one database, an error message will be issued.

This option may not be used together with the HOLD option. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

CIPHER Option

The cipher key must be specified when accessing a ciphered file. See the previous discussion on this option for more information.

COND-NAME Option

This option applies only to COBOL programs.

If the option ’COND-NAME = Y’ is coded, the record buffer generated by Adabas Native SQL includes the condition names defined in Predict as level-88 entries.

If specified here, any value specified with the global parameter OPTIONS will be overridden.

With Cond. names in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

DBID Option

This option should be used if the program accesses more than one database. The *database-name* must be defined in the data dictionary, and the data dictionary description of the database must include the file or files to be accessed.

This option may not be used together with the HOLD option. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

HOLD Option

If the HOLD option is coded, the record retrieved is placed in hold status. As long as a record is in hold status, it cannot be updated or deleted by any other user. A record that is to be updated or deleted must be in hold status unless the program is running in exclusive-control mode.

See *HOLD Logic* for more information.

The HOLD option may not be used together with the AUTODBID or DBID options. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

INDEXED Option

This option applies only to COBOL programs.

If the INDEXED option is specified, all multiple-value fields and periodic groups are generated with the ’INDEXED BY’ keywords. The name of the index is taken from Predict. If no index name is defined in the data dictionary, the name of the multiple-value field or periodic group is used, prefixed with ’I-’.

Any specification here will override any setting of the global parameter OPTIONS.

Indexed by in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option and section *Generate COBOL Copy Code* in the *Predict Administration Manual* for more information.

ISN Option

This option is used if the file is to be read in physical sequence starting at some position other than the beginning of the file.

The ISN parameter specifies the ISN of the record preceding the record where reading is to begin. The ISN is specified in the *value2* field. *value2* may be a constant or the name of a variable containing the ISN. If *value2* is a variable name, it must be immediately preceded by a colon (’:’), for example ’:NAME’. This field is updated automatically by Adabas and need not be modified by the user every time the next record is to be read.

PASSWORD Option

The password must be specified in each Adabas Native SQL statement that accesses a password-protected file or a file that is protected by security by value, unless it is specified globally in the CONNECT statement. See the previous discussion on this option for more information.

PREFIX Option

If the option ’PREFIX = *prefix*’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name prefix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

STATIC Option

This option applies to PL/I programs only.

If the option ’STATIC = Y’ is coded here, all buffers generated by Adabas Native SQL will be defined as static. This will override any setting of the global parameter OPTIONS.

Static in the Predict Modify PL/I Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

SUFFIX Option

If the option ’SUFFIX = *suffix*’ is coded, the field names generated for the record buffer will include the specified suffix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name suffix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

The READ USERDATA Statement

```
EXEC ADABAS
    READ USERDATA
        INTO   var1
      [ USERID=   value ]

END-EXEC
```

This statement reads user data previously stored in the Adabas system file by a CHECKPOINT, COMMIT WORK or DBCLOSE statement.

The user data will be read into the variable whose name is *var1*. The variable name must be immediately preceded by a colon (’:’), for example ’READ USERDATA INTO :NAME’.

This statement generates an Adabas RE command.

USERID Clause

```
USERID=   value
```

If you wish to read data stored by another user, or stored by you during a different Adabas session, the USERID parameter must be used, specifying the user-ID that was used when the data was written. *value* can be an alphanumeric constant or the name of a variable containing the user-ID. If *value* is a variable name, it must be immediately preceded by a colon (’:’). The colon must not be coded if *value* is a constant.

Examples:

```
EXEC ADABAS
  READ USERDATA INTO :USER-VAR
        USERID = ’US01’
END-EXEC

EXEC ADABAS
  READ USERDATA INTO :TEMP1
        USERID = :HISNAME
END-EXEC
```

The RELEASE Statement

```
EXEC ADABAS
   RELEASE [cursor-name]
     [ FOR [FORMAT] [GLOBAL] [SEQ] [LIST ] ]

END-EXEC
```

You will not normally need this statement. It is used to release the Adabas global format-ID and/or an Adabas command-ID.

The command-ID has three functions:

- to identify a format buffer so that further use of the same format buffer with the same command-ID is more efficient,
- to identify the next READ statement in a sequential read process,
- to identify a list of ISNs found in a FIND statement.

You can release the command-ID from one, two or all three of the above functions. If the FOR clause is not specified, then the command-ID will be released from all the functions and in addition the global format-ID will be released.

Function	Meaning
FORMAT	Releases the command-ID from the internal format buffer pool.
GLOBAL	Releases the ADABAS global format-ID.
SEQ	Releases the command-ID from the table of sequential commands.
LIST	Releases the command-ID from the table of ISN lists.

The command-ID that will be released is the command-ID generated by Adabas Native SQL for the set of buffers identified by *cursor-name*.

If *cursor-name* is not specified, all command-IDs will be released.

See the description of the RC command in the *Adabas Command Reference Manual* for more information.

This statement generates an Adabas RC command.

The RELEASE ISN Statement

```
EXEC ADABAS
    RELEASE ISN   cursor-name
END-EXEC
```

This statement releases from hold status a record that has been held by a previous READ or HOLD statement with the same *cursor-name* identification.

If you are using ET logic, do not use this statement to release a record that has been updated during your current session.

The COMMIT WORK statement, which is used in ET-mode programs to mark the end of a logical transaction, automatically releases records that were put into hold status during the current transaction.

This statement generates an Adabas RI command.

The RESTORE Statement

```
EXEC ADABAS
    RESTORE
        cursor   var1
END-EXEC
```

This statement is used in programs that run under the control of a TP monitor, for example CICS in pseudo-conversational mode or UTM with multiple-step transactions.

The data to be restored must be passed to the RESTORE statement in *var1*, which must have a length of 80 bytes. The name of the variable *var1* must be preceded by a colon (’:’). The data is passed to the Adabas Native SQL statement identified by *cursor*.

The data must be the same data that was returned from a preceding SAVE statement. The user is responsible for preserving the data between the SAVE statement and the RESTORE statement.

See also the complementary SAVE statement.

The ROLLBACK WORK Statement

```
EXEC ADABAS
    ROLLBACK WORK
        [ WITHOUT   filename ]
END-EXEC
```

This statement is used to remove all the database modifications (insertions, deletions and updates) that have been performed since the beginning of the Adabas user session or the last COMMIT WORK or ROLLBACK statement. Note that the ROLLBACK WORK statement can modify the state of files other than the files used in the program that issued the statement. After the ROLLBACK WORK has been completed, the database has the status that it had when the last COMMIT WORK was issued.

This statement generates an Adabas BT (backout transaction) command.

WITHOUT Clause

```
WITHOUT   filename
```

The user may backout all files except one by specifying the appropriate file name in the WITHOUT parameter.

Example:

```
EXEC ADABAS
  ROLLBACK WORK
      WITHOUT PERSONNEL
END-EXEC
```

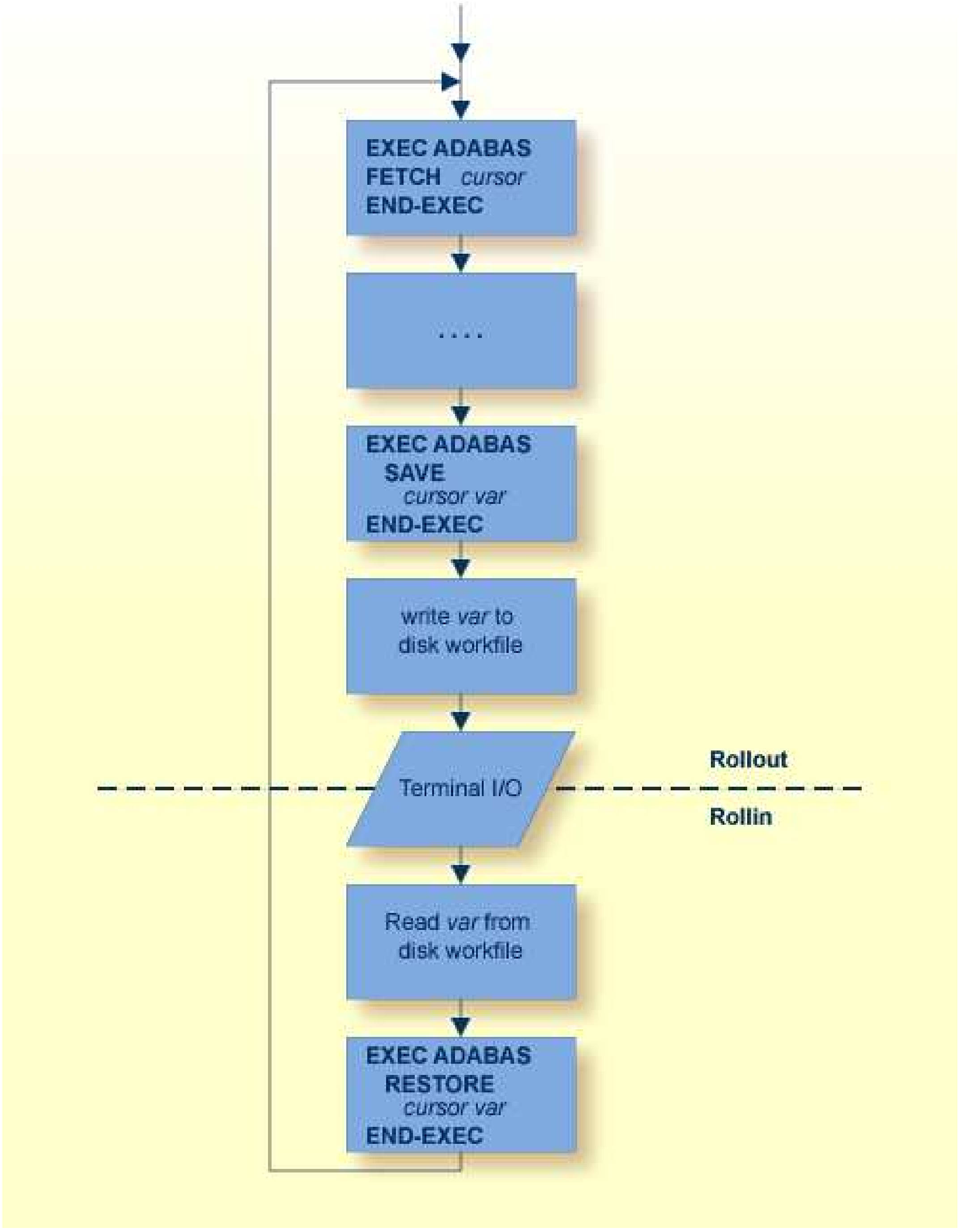
In this example, all files in the database should be backed out with the exception of file PERSONNEL.

The SAVE Statement

```
EXEC ADABAS
    SAVE
        cursor   var1
END-EXEC
```

This statement is used in programs that run under the control of a TP monitor, for example CICS in pseudo-conversational mode or UTM with multiple-step transactions. Several SAVE statements may be used, one for each Adabas Native SQL statement whose context must be preserved over an I/O transaction. However, unnecessary SAVE statements should be avoided.

A typical sequence of operations is shown in the following diagram:



The data to be saved from the Adabas Native SQL statement identified by cursor is returned in *var1*, which must have a length of 80 bytes. The name of the variable *var1* must be preceded by a colon (':'). The data will normally be used in a subsequent RESTORE statement. The user is responsible for preserving the data between the SAVE statement and the RESTORE statement.

See also the complementary RESTORE statement.

Programs that use the SAVE statement must not use the ISNSIZE option in any Adabas SQL statements.

The SORT Statement

```
EXEC ADABAS
    SORT [ISN [LISTS]]
  [DECLARE cursor-name CURSOR [FOR]]
  [SELECT { field-name ,... }]
          { *              }
    FROM  file [alias]
    WHERE CURSOR= cursor-name

  [OPTIONS
    [{ AUTODBID           }]
    [{ DBID= database-name }]
     [CIPHER= value1        ]
     [COND-NAME= { Y }]
     [           { N }]
     [HOLD [RETURN]]
     [INDEXED= { Y }]
     [         { N }]
     [ISNSIZE= len  ]
     [PASSWORD= value2 ]
     [PREFIX= prefix]
     [SAVE          ]
     [STATIC= { Y }]
     [        { N }]
     [SUFFIX= suffix ]
  ]
  [ORDER BY  de 1..3 { DESCENDING }]
  [                  { ASCENDING  }]
END-EXEC
```

The SORT statement may be used to sort an ISN list that was created by a COMPARE or FIND statement. The SAVE option must be used in the COMPARE or FIND statement in order to save the ISN list.

The ISN list is sorted according to the values of one, two or three descriptors in the records indicated by the entries in the given ISN list. The keyword DESCENDING, which may be abbreviated to DESC, specifies descending sequence, otherwise ascending sequence will be assumed. If more than one descriptor is specified, the ASCENDING/DESCENDING option applies collectively to all of them. It is not possible to specify ascending sequence for one descriptor and descending sequence for another.

The ISN list to be sorted must be in ascending ISN sequence. An ISN list that was produced by a FIND statement with the ORDER BY clause or a SORT command cannot be sorted.

In general, the SORT statement will return a list containing the ISNs of many records.

If more than one of the records listed in the ISN list returned by the SORT statement are to be processed, then the SORT statement must include the 'DECLARE *cursor-name* CURSOR FOR' clause and it must be followed by an OPEN/FETCH/CLOSE sequence. The fields specified in the SELECT clause are available after execution of the FETCH statement.

If, however, only the first of these records is to be processed, then the DECLARE clause may be omitted and the fields specified in the SELECT clause are available after execution of the SORT statement. In this case, Adabas Native SQL generates executable code for the SORT statement, which must therefore appear in the procedure division in COBOL programs.

An Adabas S9 command is generated.

DECLARE Clause

```
DECLARE cursor-name CURSOR [FOR]
```

This clause specifies a cursor-name that identifies, or labels, the current statement. Subsequent statements can refer back to a statement that is labeled with a DECLARE clause by quoting the cursor-name. The cursor-name may be up to four characters long.

If multiple records are to be processed, the 'DECLARE *cursor-name* CURSOR FOR' construction must be used. The keyword 'FOR' indicates to Adabas Native SQL that the statement is used in conjunction with OPEN and FETCH statements that appear later in the program quoting the same cursor-name. If only a single record is to be processed, the DECLARE clause may be omitted.

SELECT Clause

```
SELECT { field-name ,... }
       {      *          }
```

The SELECT clause specifies which fields are to be retrieved from the database. All types of fields may be selected, with the exception of subdescriptors, superdescriptors and phonetic descriptors. The fields must be specified by their full names as defined in the data dictionary.

If an asterisk is specified following the keyword 'SELECT', all the fields within the userview are read.

See page for more information.

FROM Clause

```
FROM    file  [alias]
```

The FROM clause specifies the file from which data are to be retrieved. It is used together with the SELECT clause to generate the record buffer and to control the retrieval of information from the database.

file is the Adabas file name or view name as defined in the data dictionary. The *alias*, if present, is used as the name of the record buffer; otherwise, the name *file* is used.

See the previous discussion on this clause for more information

WHERE Clause

```
WHERE CURSOR=   cursor-name
```

The *cursor-name* is the name coded in the DECLARE clause of the statement that created the ISN list to be sorted. This statement must include the SAVE option. It must not be a FIND statement with the ORDER BY clause or a SORT statement.

OPTIONS Clause

```
OPTIONS
    [ { AUTODBID                  } ]
    [ { DBID= database-name       } ]
     [ CIPHER= value1 ]
     [                { Y } ]
     [ COND-NAME=     { N } ]
     [ HOLD [ RETURN ] ]
     [              { Y } ]
     [ INDEXED=     { N } ]
     [ ISNSIZE= len ]
     [ PASSWORD= value3 ]
     [ PREFIX= prefix ]
     [ SAVE ]
     [           { Y } ]
     [ STATIC=   { N } ]
     [ SUFFIX= suffix ]
```

AUTODBID Option

The AUTODBID option can be used if the file is linked to a single database. This option indicates to Adabas Native SQL that the database ID is to be taken from the data dictionary. If the file is linked to more than one database, an error message will be issued.

If the file is linked to more than one database, the DBID option should be used.

This option may not be used together with the HOLD option. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

CIPHER Option

The cipher key must be specified when accessing a ciphered file. See the previous discussion on this option for more information.

COND-NAME Option

This option applies only to COBOL programs.

If the option 'COND-NAME = Y' is coded, the record buffer generated by Adabas Native SQL includes the condition names defined in Predict as level-88 entries.

If specified here, any value specified with the global parameter OPTIONS will be overridden.

With Cond. names in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

DBID Option

This option should be used if the program accesses more than one database. The *database-name* must be defined in the data dictionary, and the data dictionary description of the database must include the file or files to be accessed.

This option may not be used together with the HOLD option. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

HOLD Option

If the HOLD option is coded, the record retrieved is placed in hold status. As long as a record is in hold status, it cannot be updated or deleted by any other user.

A record that is to be updated or deleted must be in hold status unless the program is running in exclusive-control mode. See *HOLD Logic* for more information.

The HOLD option may not be used together with the AUTODBID or DBID options. This implies to Adabas Native SQL that you are attempting to update a database other than your default database.

INDEXED Option

This option applies only to COBOL programs.

If the INDEXED option is specified, all multiple-value fields and periodic groups are generated with the 'INDEXED BY' keywords. The name of the index is taken from Predict. If no index name is defined in the data dictionary, the name of the multiple-value field or periodic group is used, prefixed with 'I-'.

Any specification here will override any setting of the global parameter OPTIONS.

Indexed by in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option and section *Generate COBOL Copy Code* in the *Predict Administration Manual* for more information.

ISNSIZE Option

The ISNSIZE parameter defines the maximum number of ISNs that can be stored in the ISN buffer. For the SORT statement, the ISN buffer must either be defined with size 0, or else it must be large enough to contain the entire ISN list that is to be sorted. If an ISN buffer is defined that is too small to contain the entire ISN list, the value 1 will be returned in the response-code.

The value of len must be either 0 or at least four times the number of records to be sorted.

PASSWORD Option

The password must be specified in each Adabas Native SQL statement that accesses a password-protected file or a file that is protected by security by value, unless it is specified globally in the CONNECT statement (see the previous discussion on this option for more information).

PREFIX Option

If the option ’PREFIX = *prefix*’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name prefix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

SAVE Option

The SAVE option is used if the programmer needs to retain the entire ISN list. The saved ISN list is discarded when:

- a further Adabas Native SQL statement that creates another ISN list with the same name (same command ID) is executed, or
- an Adabas Native SQL CLOSE or DBCLOSE statement is executed, or
- the non-activity time limit or transaction time limit is exceeded. Under these circumstances, response code 9 is returned when the next Adabas command is attempted.

A CLOSE statement must be executed to release the ISN list after every statement that generates an ISN list (COMPARE, FIND, FIND COUPLED and SORT). If the CLOSE statement is not executed, large amounts of storage will be occupied for the remainder of the Adabas session.

STATIC Option

This option applies to PL/I programs only.

If the option ’STATIC = Y’ is coded here, all buffers generated by Adabas Native SQL will be defined as static. This will override any setting of the global parameter OPTIONS.

Static in the Predict Modify PL/I Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

SUFFIX Option

If the option 'SUFFIX = *suffix*' is coded, the field names generated for the record buffer will include the specified suffix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name suffix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

ORDER BY Clause

```
                    { DESCENDING }
ORDER BY  de 1..3   {            }
                    { ASCENDING  }
```

The ORDER BY clause specifies the order in which the records are retrieved.

The ISN list may be sorted on up to three descriptors in ascending or descending sequence.

A descriptor used in an ORDER BY clause may not be a member of a periodic group, nor may it be a phonetic descriptor.

The keyword DESCENDING, which may be abbreviated to DESC, specifies descending sequence, otherwise ascending sequence is assumed. If more than one descriptor is specified, the ASCENDING/DESCENDING option applies collectively to all of them. It is not possible to specify ascending sequence for one descriptor and descending sequence for another.

If the ISN list is too big to be sorted, an error is reported with response-code=1. See also the LS (sort work space) parameter in the *Adabas Operations Manual*.

Note: Sorting large ISN lists may take a long time.

The TRACE Statement

```
EXEC ADABAS
       { ON  }
 TRACE {     }
       { OFF }
END-EXEC
```

This statement is used in conjunction with the global option MODE TRACE.

Provided 'MODE TRACE.' has been specified, the TRACE ON and TRACE OFF statements can be used within the application program to control trace output statically. Trace output will only be produced in those program sections where TRACE ON is in effect.

Tracing is also controlled dynamically by a variable with the name TRCE (Ada, COBOL, PL/I) or SQDE00 (FORTRAN) in sections where TRACE ON is in effect. The application program can disable tracing dynamically by setting the content of this variable to the value 'OFF', and can re-enable tracing by setting its content to any other value.

Three conditions must be satisfied for tracing to be active:

- the global option ’MODE TRACE.’ must be set,
- the ’TRACE ON’ statement must be issued, and
- the variable ’TRCE’ or ’SQDE00’ must not contain the value ’OFF’.

Note:
tracing is switched off when the program is started. No trace output will be produced until a TRACE ON statement is executed.

The UPDATE Statement

```
EXEC ADABAS
    UPDATE file [alias]
  [ DECLARE cursor-name CURSOR ]
    WHERE { ISN= value
          { CURRENT OF cursor-name1 }
  [ SET expression ,... ]

  [ OPTIONS
      [ CIPHER= value1 ]
      [ COND-NAME= { Y
                   { N } ]
      [ INDEXED= { Y
                 { N } ]
      [ PASSWORD= value2 ]
      [ PREFIX= prefix ]
      [ STATIC= { Y
                { N } ]
      [ STATUS ]
      [ SUFFIX= suffix ]      ]

END-EXEC
```

The UPDATE statement is used to update one or more fields of a record in the specified file. The record to be updated must be retrieved by the FIND statement or one of the READ statements before issuing the UPDATE statement. The record must be in hold status unless the program is running in EXU mode (see the CONNECT statement). A record can be ’held’ either by specifying the HOLD option in the statement that reads it, or by issuing a separate HOLD statement.

When the logical transaction has been completed, a COMMIT WORK statement should be issued. One of the effects of this statement is to release records that are in hold status.

UPDATE *file* [*alias*]

file is the Adabas file name or view name as defined in the data dictionary. The *alias*, if present, is used as the name of the record buffer; otherwise, the name *file* is used. The alias, which should be unique, is required if two or more Adabas Native SQL statements within the module refer to the same file. It can be used as a qualifier in subsequent Ada, COBOL, FORTRAN or PL/I statements that refer to the fields in the record buffer.

This statement generates an Adabas A1 command.

DECLARE Clause

DECLARE *cursor-name* **CURSOR**

The cursor-name may be up to four characters long. A cursor-name should be specified if this Adabas Native SQL statement is executed repeatedly; Adabas can recognize the cursor-name, which is also used as the Adabas command-ID, and avoid re-translating the format buffer when the statement is executed subsequently.

WHERE Clause

$$\textbf{WHERE} \left\{ \begin{array}{l} \textbf{ISN=}\ \ \textit{value} \\ \textbf{CURRENT OF}\ \ \textit{cursor-name1} \end{array} \right\}$$

The WHERE clause is used to specify the ISN of the record to be updated.

To update a record having a specific ISN, the programmer should use the ’ISN = value’ option. value may be a constant or the name of a variable containing the ISN. If value is a variable name, it must be immediately preceded by a colon (’:’), for example ’:NAME’. The colon must not be coded following the ’=’ sign if value is a numeric constant, for example ’WHERE ISN = 1234’. If the ’WHERE ISN = value’ option is used, the SET clause must be coded.

To update a record using the ISN returned by a previous Adabas Native SQL statement, the programmer should use the ’CURRENT OF’ option. *cursor-name1* is the cursor-name defined in that statement.

If the user uses the ’CURRENT OF *cursor-name1*’ option in the WHERE clause and the DECLARE and SET clauses are omitted, Adabas Native SQL will use the Adabas variables generated for the statement identified by cursor-name1 and will not generate variables for this statement. In this case, modify the desired fields before issuing the UPDATE statement.

Example:

```
EXEC ADABAS
     FIND
     DECLARE PERS CURSOR
     SELECT SALARY
     FROM PERSONNEL
     WHERE PERSONNEL-NUMBER = 180001
     OPTIONS HOLD
```

```
END-EXEC
   .
   .
   .
SALARY = SALARY * 1.2
EXEC ADABAS
     UPDATE PERSONNEL
     WHERE CURRENT OF PERS
END-EXEC
```

SET Clause

SET *expression* ,...

The SET clause specifies the fields to be updated and, optionally, the values to be given to these fields. The expressions may be separated by blanks (spaces) or commas.

The SET clause must always be coded if the option ’WHERE ISN = value’ is used.

If the SET clause is coded, it is recommended that the ’DECLARE *cursor-name* CURSOR’ clause be used as well to enhance performance.

Coding the SET clause causes Adabas Native SQL to generate a record buffer for this statement. If the SET clause is not coded, the record buffer of the statement referenced by cursor-name1 will be used to update the database.

expression

$$\left\{ \begin{array}{l} \textit{field-name} \\ \textit{field-name} = \left\{ \begin{array}{l} \textit{constant} \\ \textit{var-name} \\ \textit{NULL} \end{array} \right\} \end{array} \right\}$$

field-name denotes the name of the field to be updated. This is the full field name as defined in the data dictionary. If necessary, the *field-name* can be subscripted to select the required field from a multiple-value field, from a periodic group, or from a multiple-value field within a periodic group.

The option ’SET *field-name*’ is used when the required value has already been assigned to the field by means of normal Ada, COBOL, FORTRAN or PL/I statements.

Note:
field-name can be a multiple-value or a part of a periodic group, but in this case an index must be specified within parentheses. For a multiple-value within a periodic group the user should move the value by himself before the INSERT/UPDATE statement.

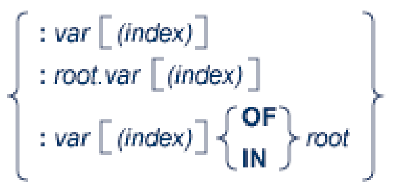
The option ’SET *field-name* = *constant*’ or ’SET *field-name* = *var-name*’ is used to specify the new value to be assigned to the field.

constant denotes a constant (literal) value and *var-name* denotes the name of a variable defined in the Ada, COBOL, FORTRAN or PL/I program, which must be preceded by a colon.

If NULL is specified, Adabas Native SQL will move -1 (x’FFFF’) to the Null field indicator of the specified field in the Record buffer used for updating the file.

If the user uses the SET clause and specifies a real value or a variable for a field which has a Null value indicator, Adabas Native SQL will automatically reset the Null field indicator of that field. If the user does not specify the SET clause, but initiates the fields in the Record buffer by himself, he should also reset or turn on the Null field indicator.

var-name



If the variable name is unique within the program, it can be specified as :*var*. Otherwise, it should be made unique by preceding it by *roo*t, a higher-level data name (qualifier) in the data structure hierarchy. Both the COBOL-type construction (:*var* OF *root* or :*var* IN *root*) and the PL/I-type construction (:*root.var*) are valid in Ada, COBOL, FORTRAN and PL/I programs.

Both the ’SET *field-name*’ option and the ’SET *field-name* = *data*’ option can be used in the same SET clause.

The optional *index* may be an integer constant or the name of a variable preceded by a colon. Note that blanks (spaces) are not allowed between the :*var* and the parenthesis preceding the *index*.

Example 1: Ada

```
type REC_1 is
  record
     SALARY     : STRING (1..6);
     AGE        : STRING (1..2);
     PERSON_NAME: STRING (1..20);
  end record;
REC: REC_1;
    .
    .
    .
EXEC ADABAS
     FIND
     DECLARE PERS CURSOR
     FROM PERSONNEL PRSNNL
     WHERE PERSONNEL_NUMBER = "00180001"
     OPTIONS HOLD
END-EXEC
     .
     .
PERSONNEL.PHONE_NR = "00746127";
EXEC ADABAS
     UPDATE PERSONNEL
     WHERE CURRENT OF PERS
     SET NAME             = :REC.PERSON-NAME
```

```
         AGE              = :REC.AGE
         SALARY           = :REC.SALARY
         PHONE_NR
         ZIP              = "06100"
         STATE            = "BS"
END-EXEC
```

Example 2: COBOL

```
01 REC
   02 SALARY ......
   02 AGE ......
   02 PERSON-NAME ......
      .
      .
EXEC ADABAS
     FIND
     DECLARE PERS CURSOR
     FROM PERSONNEL PRSNNL
     WHERE PERSONNEL-NUMBER = 180001
     OPTIONS HOLD
END-EXEC
      .
      .
MOVE 746127 to PHONE-NR OF PERSONNEL
EXEC ADABAS
     UPDATE PERSONNEL
     WHERE CURRENT OF PERS
     SET NAME   = :PERSON-NAME
         AGE    = :AGE OF REC
         SALARY = :REC.SALARY
         PHONE-NR
         ZIP    = 35
         STATE  = 'BS'
END-EXEC
```

Example 3: FORTRAN

```
CHARACTER*    20 VARNAM
CHARACTER*     2 VARAGE
CHARACTER*     6 VARSAL
.......
CHARACTER*    20 NAME
CHARACTER*     2 AGE
CHARACTER*     6 SALARY
CHARACTER*     8 PHONE
CHARACTER*     5 ZIP
CHARACTER*     2 STATE
CHARACTER*    43 PERNEL

EXEC ADABAS
    DECLARE PERS CURSOR
    FROM PERSONNEL
    WHERE PERSONNEL-NUMBER = '00180001'
    OPTIONS HOLD PREFIX=A
END-EXEC

PNONE = '00746127'

EXEC ADABAS
    UPDATE PERSONNEL
    WHERE  CURRENT OF PERS
    SET NAME     =   :VARNAM
        AGE      =   :VARAGE
        SALARY   =   :VARSAL
        PHONE
        ZIP      = ’35’
        STATE    = ’BS’
END-EXEC
```

Note:
Synonyms are assumed to be defined in the data dictionary as shown in Appendix B, and truncation is assumed to occur in the middle of the word. (The maximum length of names is operating-system dependent.)

Note:
The field PERNEL encompasses all other fields and is the equivalent of the record buffer in Ada, COBOL and PL/I.

Example 4: PL/I

```
DCL 01 REC,
      02 SALARY ......,
      02 AGE ......,
      02 PERSON_NAME ......;
     .
     .
EXEC ADABAS
     FIND
     DECLARE PERS CURSOR
     FROM PERSONNEL PRSNNL
     WHERE PERSONNEL-NUMBER = 180001
     OPTIONS HOLD
END-EXEC
    .
    .
PERSONNEL.PHONE_NR = 746127;
EXEC ADABAS
     UPDATE PERSONNEL
     WHERE CURRENT OF PERS
     SET NAME             = :PERSON-NAME
         AGE              = :AGE OF REC
         SALARY           = :REC.SALARY
         PHONE-NR
         ZIP              = 6100
         STATE            = ’BS’
END-EXEC
```

OPTIONS Clause

```
OPTIONS
 [ CIPHER= value1 ]
 [ COND-NAME= { Y } ]
 [            { N } ]
 [ INDEXED= { Y } ]
 [          { N } ]
 [ PASSWORD= value2 ]
 [ PREFIX= prefix ]
 [ STATIC= { Y } ]
 [         { N } ]
 [ STATUS ]
 [ SUFFIX= suffix ]
```

CIPHER Option

The cipher key must be specified when accessing a ciphered file. See the previous discussion on this option for more information.

COND-NAME Option

This option applies only to COBOL programs.

If the option ’COND-NAME = Y’ is coded, the record buffer generated by Adabas Native SQL includes the condition names defined in Predict as level-88 entries.

If specified here, any value specified with the global parameter OPTIONS will be overridden.

With Cond. names in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

INDEXED Option

This option applies only to COBOL programs.

If the INDEXED option is specified, all multiple-value fields and periodic groups are generated with the ’INDEXED BY’ keywords. The name of the index is taken from Predict. If no index name is defined in the data dictionary, the name of the multiple-value field or periodic group is used, prefixed with ’I-’.

Any specification here will override any setting of the global parameter OPTIONS.

Indexed by in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option and section *Generate COBOL Copy Code* in the *Predict Administration Manual* for more information.

PASSWORD Option

The password must be specified in each Adabas Native SQL statement that accesses a password-protected file or a file that is protected by security by value, unless it is specified globally in the CONNECT statement. See the previous discussion on this option for more information.

PREFIX Option

If the option ’PREFIX = *prefix*’ is coded, the field names generated for the record buffer will include the specified prefix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name prefix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

STATIC Option

This option applies to PL/I programs only.

If the option ’STATIC = Y’ is coded here, all buffers generated by Adabas Native SQL will be defined as static. This will override any setting of the global parameter OPTIONS.

Static in the Predict Modify PL/I Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

STATUS Option (available with Adabas Version 4 only)

The STATUS option invokes the Status Protection option of Adabas. This causes the data protection information for the statement to be physically written to the Data Protection Log at the time the statement is processed.

Note:
Use of the STATUS option is not recommended. See section *Status Protection Option* in chapter *Concepts and Facilities* of the *Adabas Command Reference Manual* for more information.

SUFFIX Option

If the option ’SUFFIX = *suffix*’ is coded, the field names generated for the record buffer will include the specified suffix. Any value here will override values specified with the global parameter OPTIONS or taken from Predict.

Field name suffix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this option for more information.

The WHENEVER Statement

```
EXEC ADABAS
                            ┌ CONTINUE     ┐
     WHENEVER ┌ NOT FOUND ┐ │              │
              └ SQLERROR  ┘ ┤ GO TO  label ├
                            │              │
                            └ GOTO   label ┘
END-EXEC
```

The WHENEVER statement is used to control the error handling of the program. It affects the code generated by the Adabas Native SQL preprocessor for handling exception conditions.

The 'WHENEVER NOT FOUND GOTO *label*' statement specifies a label to which the program should jump if the 'no records found' condition occurs as a result of the execution of a COMPARE, FIND, FIND COUPLED or SORT statement.

The 'WHENEVER SQLERROR GOTO *label*' statement specifies a label to which the program should jump if an error response code (response code neither = 0 nor = 3) occurs as a result of the execution of an Adabas Native SQL statement.

The 'WHENEVER ... CONTINUE' statement causes the Adabas Native SQL preprocessor to stop generating test-&-branch code after each ADABAS Native SQL statement.

If a 'WHENEVER SQLERROR ...' statement is coded, it deactivates the error handling routine of the standard abort module. You should normally use the SQLERROR together with ABORT.

The variables ISN, QUANTITY and RESPONSE_CODE (Ada, COBOL and PL/I unless the global parameter 'ABORT .' is coded) or SQLISN, SQLQTY and SQLRSP (Ada, COBOL and PL/I if the global parameter 'ABORT .' is coded; also FORTRAN) contain the values from the most recent ADABAS Native SQL statement. These can be used for error analysis.

See sections *Additional Fields in the Record Buffers* and *Response Code Interpretation* for more information.

See also description of the ABORT parameter for more information on error processing.

The WRITE TO LOG Statement

```
EXEC ADABAS
     WRITE TO LOG
          USERDATA= var
END-EXEC
```

This statement is used to write user data to the Adabas data protection log. This data may be read and displayed with the ADASEL utility program. See the *Adabas Utilities Manual* for more information.

USERDATA Clause

```
USERDATA= var
```

The data to be written must be stored in the variable denoted by var. The variable name must be immediately preceded by a colon (’:’), for example ’USERDATA = :NAME’. The length of the user data, that is, the number of characters to be written, must not exceed the limit specified in the USERDATA clause of the global parameter. OPTIONS.