SINGLE AND MULTIPLE-RECORD PROCESSING

Adabas Native SQL data retrieval statements operate in one of two modes: single-record processing mode and multiple-record processing mode.

The READ ISN statement always operates in single-record mode. The Adabas Native SQL statements in the following list can be used in either single-record processing or multiple-record processing mode:

- COMPARE
- FIND
- FIND COUPLED
- HISTOGRAM
- READ LOGICAL
- READ PHYSICAL SEQUENCE
- SORT.

Adabas Native SQL generates the appropriate data declarations and code for multiple-record processing if the keyword FOR is present in the DECLARE clause of the statement (see list above). If the FOR keyword is not coded or if the DECLARE clause is omitted, Adabas Native SQL generates code for single-record processing.

This chapter covers the following topics:

- Single-Record Processing
- Multiple-Record Processing

Single-Record Processing

If single-record processing is to be used, the OPEN, FETCH and CLOSE statements are not required and only the FIND, READ, etc., statement is required. Adabas Native SQL generates executable code from this statement, which must therefore appear in the procedure division of COBOL programs. In FORTRAN, the statement must be included within the executable statements.

Single-record processing should be used if the user needs to access only one record from the file.

Example (single-record processing):

```
    .
    .
    .
EXEC ADABAS
     SELECT PERSON
     FROM PERSONNEL
     WHERE PERSONNEL-NUMBER = 180001
END-EXEC
DISPLAY NAME FIRST-NAME AGE SEX.
```

In this example, the program uses the single-record processing method to display data from the record located by the WHERE criterion.

Multiple-Record Processing

The OPEN, FETCH and CLOSE statements are used for multiple-record processing. The set of records to be processed is determined using a COMPARE, FIND, HISTOGRAM, READ or SORT statement, followed by an OPEN statement. The records are then processed one by one using the FETCH statement, which will normally be coded in a loop. Finally, the CLOSE statement, which releases the ISN list and other Adabas resources, must be issued if the records were located by a FIND, COMPARE or SORT statement, i.e., if an ISN list was created.

It is the FETCH statement that actually reads each record from the database file and retrieves the values in the fields specified in the SELECT clause of the COMPARE, FIND, HISTOGRAM, READ or SORT statement. The OPEN, FETCH and CLOSE statements generate executable Adabas commands, whereas the COMPARE, FIND, HISTOGRAM, READ or SORT statement merely sets up parameter lists for later use.

The keyword FOR must be specified in the DECLARE clause of COMPARE, FIND, HISTOGRAM, READ or SORT in multiple-record processing mode. Using the DECLARE clause, you define a cursor that associates a ’cursor-name’ with the statement. Once the cursor has been defined, it may be referred to in the OPEN, FETCH and CLOSE statements. These statements have the following syntax:

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

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

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

cursor-name is the name used in the FIND, READ, SORT, COMPARE or HISTOGRAM statement that was previously declared. The cursor-name provides the link between the parameter-defining statement (FIND, READ, SORT, COMPARE or HISTOGRAM) and the corresponding executable statements (OPEN, FETCH and CLOSE).

Example (multiple-record processing):

```
    .
    .
    .
    .
  EXEC ADABAS
       DECLARE PERS CURSOR FOR
       SELECT PERSON
       FROM PERSONNEL
       WHERE NAME > = 'BROWN'
  END-EXEC
    .
    .
    .
    .
  EXEC ADABAS
       OPEN PERS
  END-EXEC
  EXEC ADABAS
       FETCH PERS
  END-EXEC
  PERFORM READ-PERSONNEL UNTIL ADACODE = 3.
  EXEC ADABAS
       CLOSE PERS
  END-EXEC
    .
    .
    .
    .
READ-PERSONNEL.
  DISPLAY NAME FIRST-NAME AGE SEX.
  EXEC ADABAS
       FETCH PERS
  END-EXEC
```