VSAM to DB2

Online And Batch Program Conversion Strategy

Version 1.0

Preface

Scope and audience

This document is intended to assist in converting online programs using COBOL CICS and batch programs using COBOL which access VSAM files to access DB2 tables.

.

Objectives

VSAM to DB2 conversion

Structure of this document

Document discusses schema translation strategy initially. Translation of batch COBOL access calls for VSAM to DB2 is discussed. Translation of CICS access calls to DB2 access calls is detailed out next for each of the CICS commands for KSDS files. The document ends with the performance tuning issues to be considered.

Related documents

VSAM to DB2 error handling document

Table of contents

1 Overview

This document discusses translation strategy for VSAM access calls appearing in online COBOL CICS and batch COBOL programs by SQLs to access DB2 table.

Emphasis is on one to one translation of each VSAM access call to DB2 access SQLs.

2 Schema Translation

During this phase of the migration, VSAM file schema needs to be translated to relational schema. For this a complete mapping document needs to be prepared for the VSAM files being converted to DB2 tables. This document has the VSAM file name, the record name, copybook name for this file, the field names of the record, with the corresponding DB2 table names, column names, data description for the columns, indicators to identify whether nulls are allowed, the column is an index and mapping exceptions of any kind. It is understood that in such conversion there are cases of multiple mapping, like one VSAM file mapped to many tables, many VSAM files mapped to one table, different record types of a VSAM file mapped to different tables, one field of a VSAM record mapped to multiple columns of a table, many fields of a VSAM record mapped to one table column.

3 Conversion Strategy

The conversion is based on the following translation rules :

1. COBOL source code will remain same except for the introduction of new code and commenting of existing code as part of translation. New code will be either embedded SQL statements or COBOL constructs or INCLUDE copybooks, to support translation. Comments will be added to indicate the program changes done.
2. DCLGEN variables will be used as host variables. These values will be moved into the VSAM file structure for the program processing.
3. VSAM sequential operations will be translated using cursors on the associated tables.
4. I/O operations on array fields of the record will be translated using cursors for normalised cases. In denormalised cases the array fields will be translated to the corresponding columns on the DB2 table.
5. No cursor will be defined if file is opened in OUTPUT or EXTEND mode.
6. Any supported COBOL construct related to VSAM file, which is translated to relational equivalent or not needed any more will be commented.
7. In case of commenting OPEN statements, if the same statement opens some other non-VSAM files or VSAM files which will not be converted, an OPEN statement to open those files will be added in the program and the original OPEN statement will be commented.
8. In case of commenting CLOSE statements, if the same statement closes some other non-VSAM files or VSAM files which will not be converted, an CLOSE statement to close those files will be added in the program and the original CLOSE statement will be commented.
9. At any point of time, there will be only one cursor opened per table. Cursors will be defined and opened with various conditions based on organization, access mode and file operation.
10. Cursor will be defined

    With ORDER BY condition for indexed files.

    With WHERE condition for START and READ

    With FOR FETCH for read only operation

    With FOR UPDATE for update operations

11. START operations will be translated to cursor declaration with WHERE condition. The condition used in the WHERE condition will be same as the condition stated in the START. If START uses EQUAL TO, then the '=' will be used in the WHERE condition. If START uses GREATER THAN, then '>' will be used in the WHERE condition.
12. START can use a full-key or partial key. The translation will depend on the type of key used as well as the key length specified.If START uses full key then the columns mapped to this will be used in the WHERE clause. If START statement uses partial key there are 2 cases :
    a. if the keylength is a constant known at conversion time, the START will be translated to DECLARE CURSOR using the table columns corresponding to this key length specified.
    b. If the key length is calculated at run time, it needs to be analysed and suitable WHERE clause should be used in the DECLARE CURSOR clause.
13. Any column corresponding to key fields of the VSAM file needs to be used as an index on the table. Alternate record keys can be considered as index on tables. This needs table to be indexed on columns corresponding to alternate record keys.

14. CLOSE will be translated to closing the cursor(s) defined on the corresponding table. If same CLOSE statement closes non-VSAM files or VSAM files not being converted , separate close statement to close the these files will be generated and included in the source code.
15. Standard error Paragraphs will be performed to take care of any SQLCODE checking.
16. All the FILE STATUS logic used by the program will be retained for conditional processing. The FILE STATUS for the converted VSAM file will be set by the standard error processing logic, depending on the value of SQLCODE returned
17. One-to-one mapping will be handled by translating the VSAM access to the corresponding SQL statement.

    Sequential Operation for VSAM file

    a. Sequential READ operations will be translated to cursor FETCH.

    b. Sequential REWRITE will be translated to UPDATE with CURRENT OF condition.

    c. Sequential DELETE will be translated to DELETE with CURRENT OF condition.

    Random Operation for VSAM file

    a. Random READ operations will be translated to SELECT with WHERE condition.

    b. Random REWRITE will be translated to UPDATE with WHERE condition.

    c. Random DELETE will be translated to DELETE with WHERE condition.

    d. WRITE will be translated to INSERT with VALUES clause.

18. In the case of one-to-many mapping, the following scenarios are possible :

    a. One VSAM file having different record types mapped to different tables

    Sequential Operation on the VSAM file

    Example of this is header and detail records in a VSAM file. In DB2 there can be table defined for header records and another table defined for detail records. Any sequential read of this VSAM file will be translated to opening a cursor on the header table, then fetch cursor on the header table. Using the key retrieved from the header table open a cursor on the detail table and continue fetch using this cursor until end-of-cursor, then close the cursor on the detail table. Again fetch the cursor on the header table and continue the process of open cursor and fetch cursor on the detail table.
    For any sequential update or delete operations on this VSAM file, the same translation strategy will be used but with the cursor option FOR UPDATE.

    Random Operation on the VSAM file

    If the VSAM file is being read randomly using a key value moved within the program, then the program flow needs to be analysed to determine which table needs to be accessed.

    b. Fields of one VSAM file mapped to different tables

    In such cases any input-output operation on this VSAM file will be translated to an equivalent SQL statements to access data from all the associated tables.

    Sequential Operation on the VSAM file

    For a sequential read on this file, a cursor will be defined which will join required data from all the associated tables.
    For any sequential update or delete on the file, individual cursors will be defined on the associated tables with the FOR UPDATE clause. In the case of a VSAM file mapped to two

DB2 tables, the cursor on the first table will be fetched, then the cursor on the second table will be fetched. If both the rows fetched have the same key, then the DCLGEN values are formatted as required and moved to the COBOL layout of the file. Values for any updates to be made are moved in, and the tables are updated. In case the keys fetched by the two cursors are not same, then this may be either because of some invalid data in one of the tables, or because some rows are being added into either of the tables from some other VSAM file. Since this may be a valid case as per application design, additional analysis will be required.

Random Operation on the VSAM file

For a random read, translate to a SELECT clause using a join from the associated tables. For write, rewrite or delete, translate to a INSERT, UPDATE or DELETE with conditional execution. If the operation on the first table was successful, then only attempt a similar operation on the other table.

c. Many VSAM files mapped to one table

Sequential Operation on the VSAM file

Translate the sequential operation to cursor operation on the table using only the columns from table which has been mapped from the fields of this VSAM file. While inserting use default values for the columns coming from other files. While updating, update only the table columns mapped from the file being rewritten.

Random Operation on the VSAM file

For read, select only the table columns which have been mapped from the file being read. For delete, verify if the row is present in the table before attempting a delete from the table, If present then do a delete, else do not attempt to delete. This is because the delete operation which might have been performed on the other VSAM file which also maps to this table, might have already deleted the entire row. For update, update only the table columns which have been mapped from this file. For insert, verify if the key being inserted is already present in the table, if present then do an update instead of an insert, else do an insert. This is because this row might already have been inserted when a write was performed on the other file which also maps to this table. While inserting use default values for the columns coming from other files.

d. One field mapped to multiple columns

One field split into different columns

Situations were a field from the VSAM file is split into table columns will be taken care of as per the splitting specified in the mapping input. This will be valid for any fields in the file which is type DISPLAY. But any attempt to split a field defined as COMP or COMP-3 will need additional analysis.

One field mapped to different columns

In situations were a whole field can be mapped to different columns depending on some conditional processing, the exception logic associated with the mapping for that field will have to be used. This exception logic will also be used to assign the proper DCLGEN field to the COBOL field.

19. While populating default values for any NOT NULL fields in the DB2 table, assume 0 for numeric and SPACES for CHAR fields as default.
    Any NULL fetched from the table will be interpreted as a ‘0’ if the receiving field is numeric or as SPACES if the receiving field is alphanumeric.

20. Before updating or inserting into any column which has a numeric data type in DB2, perform a numeric-check. If the value is non-numeric, assign the null indicator for nullable fields and ‘0’ for non-nullable fields.

21. For groups containing numeric items mapped to a single numeric column, the group will be treated as an alphanumeric and moved into the numeric column. Such translation will hold good only if the numeric values in the sub-fields can be treated as an alphanumeric value from the business point of view. For example it is ok to map a group item having sub-items as account and sub-account, both numeric, into a single numeric table column as account. But if the same translation is performed for items like amounts, then the result will be unpredictable.

22. In many situations a field defined in the VSAM layout and the corresponding DB2 table column will have different lengths, this is because the datatypes in VSAM and DB2 are different and have their own field length restrictions.

23. For variable length VSAM files being used in the program, the table will have columns corresponding to the OCCURS fields. The number of columns will be the maximum possible value of the OCCURS clause. After conversion to tables there will be a column for each occurance of the variable occurance field, although all the columns may not be used or not have values populated in them. Within the program the logic will be to use all the occurances of the field, but this may not be in tune with the existing program logic.

24. New fields defined in the tables for any future use will be populated with the default values.

25. In case of CICS programs a record can be read into an area within the program, or into the CICS buffer. In the later situation a pointer is used for accessing the buffer area where the record is read into. After translation the use of this pointer will be avoided for data read from tables. Define a '01' level item with length equal to the record length for the file for which the pointer was used. After accessing data from the tables, format and move this data to the record layout as well as to this newly defined data item. Any use of the data stored in the buffer identified by the pointer will be replaced by using the newly defined data item.

26. Translate a START browse, READNEXT operation into OPEN cursor and FETCH cursor. After each fetch, the key values retrieved will be saved and will be compared with the key value before the next fetch. If the key values match continue with the fetch, else close the cursor and then open a new cursor using the current key values and continue the fetch operation. This feature will allow the skip sequential processing to be converted correctly.

27. Any VSAM files used in internal program sorting will also be translated to SQL statements.

28. After conversion, many file OPEN statements will have been commented because they are not required for an indexed file which is converted to DB2 table. In such cases the file status for these commented files have to be set up properly so that program processing continues normally. This can be done by setting the file status to '00' for the file OPEN which was commented out.

3.1 Batch Program Translation

3.1.1 Specific Translation Rules

Following sections specify the translation strategy for each of the valid file I/O operation for VSAM files. The rules are organized as follows:

- For every organization (sequential/indexed/relative)
- For every access mode (sequential/random/dynamic)
- For every IO mode (input/output/input-output/extend)
- All the IO statements applicable

Following <placeholder> convention is used in defining cursors and queries.

Name	Description
Dependant-cursor-1	Cursor defined on the tables containing columns corresponding to the fields of the first array
Dependant-cursor-n	Cursor defined on the tables containing columns corresponding to the fields of the nth array
Array-1	First array in the record
Db2-array-1	Newly defined structure similar to the first array but scalar, to be used in moving data between tables and the array
Array-1-table-1	First table corresponding to the first array
array-1-table-1-col-1	First column in the first table corresponding to the first array
db2-array-1-table-1-col-1-fld-1	Field in the newly defined record corresponding to the first field of the first array, which has mapping column in the first table
array-1-table-1-col-n	Nth column in the first table corresponding to the first array
db2-array-1-table-1-col-n-fld-n	Field in the newly defined record corresponding to the nth field of the first array, which has mapping column in the first table
array-1-table-n	Nth table corresponding to the first array
array-1-table-n-col-1	First column in the nth table corresponding to the first array
db2-array-1-table-n-col-1-fld-1	Field in the newly defined record corresponding to the first field of the first array, which has mapping column in the nth table
array-1-table-n-col-n	Nth column in the nth table corresponding to the first array
db2-array-1-table-n-col-n-fld-n	Field in the newly defined record corresponding to the nth field of the first array, which has mapping column in the nth table
Db2-array-1-size	Newly generated data item to store the size of the first array. This data item will be used in determining the number of times fetch/select/insert/update to be done on the tables corresponding to the array
Array-n	Nth array in the record
Db2-array-n	Newly defined structure similar to the nth array, to be used in moving data between tables and the array
array-n-table-1	First table corresponding to the nth array

array-n-table-1-col-1	First column in the first table corresponding to the nth array
db2-array-n-table-1-col-1-fld-1	Field in the newly defined record corresponding to the first field of the nth array, which has mapping column in the first table
array-n-table-1-col-n	Nth column in the first table corresponding to the nth array
db2-array-n-table-1-col-n-fld-n	Field in the newly defined record corresponding to the nth field of the nth array, which has mapping column in the first table
array-n-table-n	Nth table corresponding to the nth array
array-n-table-n-col-1	First column in the nth table corresponding to the nth array
db2-array-n-table-n-col-1-fld-1	Field in the newly defined record corresponding to the first field of the nth array, which has mapping column in the nth table
array-n-table-n-col-n	Nth column in the nth table corresponding to the nth array
db2-array-n-table-n-col-n-fld-n	Field in the newly defined record corresponding to the nth field of the nth array, which has mapping column in the nth table
Db2-array-n-size	Newly generated data item to store the size of the nth array. This data item will be used in determining the number of times fetch/select/insert/update to be done on the tables corresponding to the array
table-1-key-col-1	First key column in the first table containing columns corresponding to the scalar fields of the record
table-1-key-col-n	Nth key column in the first table containing columns corresponding to the scalar fields of the record
at-end-statements	Imperative statements to be executed on end of file
not-at-end-statements	Imperative statements to be executed on not end of file
Invalid-key-statements	Imperative statements to be executed on invalid key
Not-invalid-key-statements	Imperative statements to be executed on valid key
Db2-count	Newly generated data item to keep the loop count.

3.1.2 Sequential Organization Sequential Access

All the I/O operations in files with sequential organization and sequential access mode are sequential.

3.1.2.1 INPUT

Cursor Definition :

OPEN-CURSOR

Operation	**Description**	**Translated Code**
OPEN	Open the defined cursor.	EXEC SQL OPEN *<file1>-OPEN-CURSOR* END-EXEC.
READ		Sequential-Read-OPEN-CURSOR
CLOSE	If the CLOSE statement closes any VSAM file, then close the cursor opened for that file.	EXEC SQL CLOSE *<file1>-OPEN-CURSOR* END-EXEC

3.1.2.2 I-O

Cursor Definition :

OPEN-CURSOR

Operation	Description	Translated Code
OPEN	Open the defined cursor.	EXEC SQL OPEN *<file1>-OPEN-CURSOR* END-EXEC.
READ		Sequential-Read-OPEN-CURSOR
REWRITE		Random-Rewrite
CLOSE	Close the opened cursor.	EXEC SQL CLOSE *<file1>-OPEN-CURSOR* END-EXEC

3.1.2.3 OUTPUT/EXTEND

Operation	Description	Translated Code
OPEN	Comment CLOSE statement	None
WRITE	Insert new rows into respective tables.	Insert
CLOSE	Comment the CLOSE statement.	None

3.1.3 Indexed Organization Sequential Access

All the I/O operations in a file with index organization and sequential access are in the ascending order of the value of the record key. File positioning is achieved using either OPEN or START. For files opened in INPUT or I-O mode, query translation varies based on the existence of START on the file. In each of the above two open modes, there are two sections, START EXISTS and START DOES NOT EXIST, specifying the corresponding translation strategies.

3.1.3.1 INPUT

3.1.3.1.1 START EXISTS

This section specifies the translation strategy for VSAM files with indexed organization, sequential access mode opened in INPUT mode where START is used.

Define a cursor for each VSAM file, to select all the rows in the corresponding tables.

If START is used and if it uses full record key in key comparison, define a cursor to select all the rows in the corresponding tables where key columns have values greater than or equal to the record key values.

Define the above cursors to be ORDER BY primary key columns of one of the tables.

Define a condition name in the working storage section, to be set and reset by START and OPEN statements when they open and close the cursor. This condition name will indicate the current opened cursor to be used by other operations.

Definition of the condition name will be

01 *<file1>-CURSOR* PIC 9 VALUE 0.
88 *<file1>-OPEN* value 1.
88 *<file1>-START-RK* value 2.

Where *<file1>-OPEN* will be set when the cursor corresponding to OPEN is opened

<file1>-START-RK will be set when the cursor corresponding to START is opened.

Cursor definition for OPEN :

OPEN-CURSOR

Cursor definition for START :

START-CURSOR

Operation	Description	Translated Code
OPEN	Open the cursor <*file1*>-*OPEN-CURSOR*. This will select all the rows in the table. Set this cursor to be the current valid cursor, which can then be used by successive I/O operations till START repositions the file position indicator or CLOSE closes the file.	EXEC SQL OPEN <*file1*>-*OPEN-CURSOR* END-EXEC. MOVE 1 TO <*file1*>-*CURSOR*.
START	Start repositions the file position indicator by populating record key fields with appropriate values. If the corresponding cursor is already opened, close the cursor and open it again. Reopening the cursor will select the rows based on the current value of the host variables used in the WHERE clause. Set <file1>-START-RK-CURSOR to be the current valid cursor which can be used by successive I/O operations till another START repositions the file position indicator or CLOSE closes the file.	IF <file1>-START-RK THEN EXEC SQL CLOSE <*file1*>-*START-RK-CURSOR* END-EXEC END-IF. EXEC SQL OPEN <*file1*>-*START-RK-CURSOR* END-EXEC. MOVE 2 TO <*file1*>-*CURSOR*.
READ or READ NEXT	In sequential access files, READ NEXT is same as READ. Both of them are sequential read. READ / READ NEXT can follow either any one of OPEN, START or another READ/READ NEXT. If START exists, then cursor to fetch will be decided based on the flag which states the current valid cursor. COBOL EVALUATE statement will be used to make the decision.	EVALUATE <*file1*>-*CURSOR* WHEN <*file1*>-*OPEN* Sequential-Read-OPEN-CURSOR WHEN <*file1*>-*START-RK* Sequential-Read-START-CURSOR END-EVALUATE
CLOSE	If START exists, there will be two opened cursors. Close both the cursors at that point. Set the flag to indicate no cursors are available.	EXEC SQL CLOSE <*file1*>-*OPEN-CURSOR* END-EXEC EXEC SQL CLOSE <*file1*>-*START-RK-CURSOR* END-EXEC MOVE 0 TO <*file1*>-*CURSOR*.

3.1.3.1.2 START DOES NOT EXIST

This section specifies the translation strategy for VSAM files with indexed organization, sequential access mode opened in INPUT mode where START is not used.

Define a cursor for each VSAM file, to select all the rows in the corresponding table, ORDER BY primary key columns.

Cursor definition for OPEN :

OPEN-CURSOR

Operation	Description	Translated Code
OPEN	Open the cursor *<file1>-OPEN-CURSOR*.	EXEC SQL OPEN *<file1>-OPEN-CURSOR* END-EXEC.
READ or READ NEXT	In sequential access files, READ NEXT is same as READ. Both of them are sequential read. FETCH the cursor defined on the corresponding table.	Sequential-Read-OPEN-CURSOR
CLOSE	Close the opened cursor.	EXEC SQL CLOSE *<file1>-OPEN-CURSOR* END-EXEC

3.1.3.2 I-O

3.1.3.2.1 START-EXISTS

This section specifies the translation strategy for VSAM files with indexed organization, sequential access mode opened in I-O mode where START is used.

Define a cursor for each VSAM file, to select all the rows in the corresponding table.

If start is used and if start uses full record key in key comparison, define a cursor to select all the rows in the corresponding tables where key columns have values greater than or equal to the record key values.

Define the above cursors to be ORDER BY primary key columns.

Define a condition name in the working storage section, to be set and reset by START and OPEN statements when they open and close the cursor. This condition name will indicate the current opened cursor to be used by other operations.

Definition of the condition name will be

01 *<file1>-CURSOR* PIC 9 VALUE 0.
88 *<file1>-OPEN* value 1.
88 *<file1>-START-RK* value 2.

Where <file1>-OPEN will be set when the cursor corresponding to OPEN is opened
<file1>-START-RK will be set when the cursor corresponding to START is opened.

Cursor definition for OPEN :

OPEN-CURSOR

Cursor definition for START :

START-CURSOR

Operation	Description	Translated Code
OPEN	Open the cursor *<file1>-OPEN-CURSOR*. This will select all the rows in the table. Set *<file1>-OPEN-CURSOR* to be the current valid cursor which can then be used by successive I/O operations till START repositions the file position indicator or CLOSE closes the file.	EXEC SQL OPEN *<file1>-OPEN-CURSOR* END-EXEC. MOVE 1 TO *<file1>-CURSOR*.
START	Start repositions the file position indicator by populating record key fields with appropriate values. If the corresponding cursor is already opened, close the cursor and open it again. Reopening the cursor will	IF <file1>-START-RK THEN EXEC SQL CLOSE *<file1>-START-RK-CURSOR*

	select the rows based on the current value of the host variables used in the WHERE clause. Set <file1>-START-RK-CURSOR to be the current valid cursor which can be used by successive I/O operations till another START repositions the file position indicator or CLOSE closes the file.	END-EXEC END-IF. EXEC SQL OPEN *<file1>-START-RK-CURSOR* END-EXEC. MOVE 2 TO *<file1>-CURSOR*.
READ	In sequential access files, READ NEXT is same as READ. Both of them are sequential read. READ / READ NEXT can follow either any one of OPEN, START or another READ/READ NEXT. If START exists, then cursor to fetch will be decided based on the flag which states the current valid cursor. COBOL EVALUATE statement will be used to make the decision.	EVALUATE *<file1>-CURSOR* WHEN *<file1>-OPEN* Sequential-Read-OPEN-CURSOR WHEN *<file1>-START-RK* Sequential-Read-START-CURSOR END-EVALUATE
REWRITE		Random-Rewrite
WRITE	Insert the new rows into respective tables.	Insert
DELETE		Delete
CLOSE	As there will be two opened cursors, one by the OPEN statement and another by START statement, both the cursors will be closed at this point.	EXEC SQL CLOSE *<file1>-OPEN-CURSOR* END-EXEC EXEC SQL CLOSE *<file1>-START-RK-CURSOR* END-EXEC MOVE 0 TO *<file1>-CURSOR*

3.1.3.2.2 START–DOES NOT EXIST

This section specifies the translation strategy for VSAM files with indexed organization, sequential access mode opened in I-O mode where START is not used.

Define a cursor for each VSAM file, to select all the rows in the corresponding table, ORDER BY primary key columns.

Define the above cursors to be ORDER BY primary key columns.

Cursor definition for OPEN :

OPEN-CURSOR

Operation	**Description**	**Translated Code**
OPEN	Open the cursor defined for the corresponding file.	EXEC SQL OPEN *<file1>-OPEN-CURSOR* END-EXEC.
READ	In sequential access files, READ NEXT is same as READ. Both of them are sequential read. READ / READ NEXT can follow either any one of OPEN, START or another READ/READ NEXT. Fetch the *<file1>-OPEN-CURSOR*.	Sequential-Read-OPEN-CURSOR
REWRITE		Random-Rewrite
WRITE	Insert new rows into respective tables.	Insert

DELETE		Delete
CLOSE	Close the cursor opened corresponding to the OPEN statement.	EXEC SQL CLOSE *<file1>-OPEN-CURSOR* END-EXEC

3.1.3.3 OUTPUT / EXTEND

In this mode, there will be no READ / READ NEXT / START statements. Only allowed operations are OPEN, WRITE and CLOSE.

Operation	Description	Translated Code
OPEN	Comment OPEN statement.	None
WRITE	Insert the new row into the corresponding table. Multiple INSERT	Insert
CLOSE	Comment CLOSE statement	None

3.1.4 Indexed Organization Random Access

All the I/O operations in a file with index organization and random access mode, are random i.e., will use key values to insert or retrieve records. All the queries will be translated using WHERE condition.

3.1.4.1 INPUT

Operation	Description	Translated Code
OPEN	Comment the OPEN statement.	None
READ	Select the record with primary key columns having values equal to record key values.	Random-Read
CLOSE	Comment the CLOSE statement	None

3.1.4.2 I-O

Operation	Description	Translated Code
OPEN	Comment the OPEN statement	None
READ	Select the row with primary key columns having values equal to record key values.	Random-Read
REWRITE	Update the row with primary key columns having values equal to record key values. Multiple UPDATE	Random-Rewrite
INSERT	Insert the new row into the table corresponding to the file.	Insert
DELETE	Delete the row with primary key columns having values equal to record key values.	Delete
CLOSE	Comment CLOSE statement.	None

3.1.4.3 OUTPUT

Operation	Description	Translated Code
OPEN	Comment the OPEN statement	None
WRITE	Insert the new row into the table corresponding to the file.	Insert
CLOSE	Comment the CLOSE statement	None

3.1.5 Indexed Organization Dynamic Access

I/O operations in files with indexed organization and dynamic access, can be either sequential or random or both. OPEN opens the file with file position indicator pointing to first record. File position indicator can be

repositioned using START or random READ. Records can be read sequentially using READ NEXT and randomly using RANDOM READ.

Translation strategy varies based on the existence of START, READ and READ NEXT. File position indicator can be repositioned using either random READ or START.

For files opened in INPUT or I-O mode, file I/O operations can be sequential or random or both based on the existence of READ NEXT, RANDOM READ and START.

Following are six possible cases classifying the file operation to be sequential or random or both, based on the existence of READ NEXT, RANDOM READ and START.

Sl.No.	READ NEXT	RANDOM READ	START	Operation
1	Yes	No	No	Sequential
2	Yes	Yes	No	Sequential and Random
3	Yes	No	Yes	Sequential and Random
4	Yes	Yes	Yes	Sequential and Random
5	No	Yes	Yes	Random
6	No	Yes	No	Random

Query translation for

- Case 1 is same as for files with indexed organization and sequential access without start
- Case 3 is same as for files with indexed organization and sequential access with start
- Case 5 and 6 is same as for files with indexed organization and random access. Refer to the corresponding sections for their translation strategy. Following sections specify the translation strategies for cases 2 and 4.

3.1.5.1 INPUT

3.1.5.1.1 Case 2

This section specifies the translation strategy for VSAM files with indexed organization, dynamic access mode opened in INPUT mode where READ exists, RANDOM READ exists and START does not exist.

Define a cursor for each VSAM file, to select all the rows in the corresponding table.

If RANDOM READ is used and if it uses record key in key comparison, define a cursor to select all the rows in the corresponding table where key columns have values greater than record key values.

Define the above cursors to be ORDER BY primary key columns.

Define a condition name in the working storage section, to be set and reset by RANDOM READ and OPEN statements when they open and close the cursor. This condition name will indicate the current opened cursor to be used by other operations.

Definition of the condition name will be

01 <*file1*>-*CURSOR* PIC 9 VALUE 0.
88 <*file1*>-*OPEN* value 1.
88 <*file1*>-*READ-RK* value 2.

Where <*file1*>-*OPEN* will be set when the cursor corresponding to OPEN is opened
<file2>-READ-RK will be set when the cursor corresponding to READ is opened.

Cursor definition for OPEN :

OPEN-CURSOR

Cursor definition for READ :

READ-CURSOR

Operation	Description	Translated Code

OPEN	Open the cursor *<file1>-OPEN-CURSOR*.	EXEC SQL OPEN *<file1>-OPEN-CURSOR* END-EXEC. MOVE 1 TO *<file1>-CURSOR*.
RANDOM READ	RANDOM READ translates to SELECT with WHERE clause. As RANDOM READ repositions the file position indicator, any previously opened cursor should be closed to prevent operations following RANDOM READ, from using the cursor.	Random-Read EVALUATE *<file1>-CURSOR* WHEN *<file1>-OPEN* EXEC SQL CLOSE *<file1>-OPEN-CURSOR* END-EXEC WHEN *<file1>-READ-RK* EXEC SQL CLOSE *<file1>-READ-RK-CURSOR* END-EXEC END-EVALUATE MOVE 0 TO *<file1>-CURSOR*
READ NEXT	READ NEXT is a sequential operation and should follow successful OPEN or RANDOM READ or READ NEXT. If the previous operation is OPEN, then that would have opened the appropriate cursor and hence FETCH on the cursor corresponding to OPEN will get the correct row. If the previous operation is RANDOM READ, then that would have selected the correct record. Record key fields(host variables) would have got primary key column values of the selected record. Opening the cursor, *<file1>-READ-RK-CURSOR* , will select all the rows in the corresponding table with primary key columns have values greater than record key values and FETCH on that cursor will get the correct record. If the previous operation is READ NEXT, it would have opened the appropriate cursor and hence fetch on that cursor will get the correct record.	EVALUATE *<file1>-CURSOR* WHEN *<file1>-OPEN* Sequential-Read-OPEN-CURSOR WHEN *<file1>-READ-RK* Sequential-Read-READ-CURSOR WHEN 0 EXEC SQL OPEN *<file1>-READ-RK-CURSOR*. END-EXEC Sequential-Read-READ-CURSOR MOVE 2 TO *<file1>-CURSOR* END-EVALUATE
CLOSE	Close the opened cursors.	IF *<file1>-OPEN-CURSOR* THEN EXEC SQL CLOSE *<file1>-OPEN-* CURSOR END-EXEC END-IF. IF *<file1>-READ-RK-CURSOR* THEN EXEC SQL CLOSE *<file1>-READ-RK-CURSOR* END-EXEC END-IF.

3.1.5.1.2 Case 4

This section specifies the translation strategy for VSAM files with indexed organization, dynamic access mode opened in INPUT mode where READ exists, RANDOM READ exists and START exists.

Define a cursor for each VSAM file, to select all the rows in the corresponding table.

If RANDOM READ is used and if it uses record key in key comparison, define a cursor to select all the rows in the corresponding table where key columns have values greater than the record key values.

If START is used and if it uses record key in key comparison, define a cursor to select all the rows in the corresponding table where key columns have values greater than or equal to the record key values.

Define the above cursors to be ORDER BY primary key columns.

Define a condition name in the working storage section, to be set and reset by RANDOM READ, START and OPEN statements when they open and close the cursor. This condition name will indicate the current opened cursor to be used by other operations.

Definition of the condition name will be

01 *<file1>-CURSOR* PIC 9 VALUE 0.
88 *<file1>-OPEN* value 1.
88 *<file1>-READ-RK* value 2.
88 *<file1>-START-RK* value 3.

Where *<file1>-OPEN* will be set when the cursor corresponding to OPEN is opened,
<file1>-READ-RK will be set when the cursor corresponding to READ is opened,
<file1>-START-RK will be set when the cursor corresponding to START is opened.

Cursor definition for OPEN :

OPEN-CURSOR

Cursor definition for START :

START-CURSOR

Cursor definition for READ :

READ-CURSOR

Operation	**Description**	**Translated Code**
OPEN	Open the cursor defined corresponding to OPEN statement.	EXEC SQL OPEN *<file1>-OPEN-CURSOR* END-EXEC. MOVE 1 TO *<file1>-CURSOR*.
START	Start repositions the file position indicator by populating record key fields with appropriate values. If the corresponding cursor is already opened, close the cursor and open it again. Reopening the cursor will select the rows based on the current value of the host variables used in the WHERE clause. Set <file1>-START-RK-CURSOR to be the current valid cursor which can be used by successive I/O operations till another START/RANDOM READ repositions the file position indicator or CLOSE closes the file.	IF <file1>-START-RK THEN EXEC SQL CLOSE *<file1>-START-RK-CURSOR* END-EXEC END-IF. EXEC SQL OPEN *<file1>-START-RK-CURSOR* END-EXEC. MOVE 3 TO *<file1>-CURSOR*.
RANDOM READ	RANDOM READ translates to SELECT with WHERE clause. As RANDOM READ	Random-Read EVALUATE *<file1>-CURSOR*

	repositions the file position indicator, any previously opened cursor should be closed to prevent operations following RANDOM READ, from using the cursor.	WHEN *<file1>-OPEN* EXEC SQL CLOSE *<file1>-OPEN-CURSOR* END-EXEC WHEN *<file1>-READ-RK* EXEC SQL CLOSE *<file1>-READ-RK-CURSOR* END-EXEC WHEN *<file1>-START-RK* EXEC SQL CLOSE *<file1>-START-RK-CURSOR* END-EXEC END-EVALUATE MOVE 0 TO *<file1>-CURSOR*
READ NEXT	READ NEXT is a sequential operation and should follow either successful OPEN or RANDOM READ or READ NEXT or START. If the previous operation is OPEN or START, then that would have opened the appropriate cursor and hence FETCH on the cursor corresponding to OPEN or START will get the correct row. If the previous operation is RANDOM READ, then that would have selected the correct record. Record key fields(host variables) would have got primary key column values of the selected record. Opening the cursor, *<file1>-READ-RK-CURSOR* , will select all the rows in the corresponding table with primary key columns have values greater than or equal to record key values and FETCH on that cursor will get the correct record. If the previous operation is READ NEXT, it would have opened the appropriate cursor and hence on that cursor will get the correct record.	EVALUATE *<file1>-CURSOR* WHEN *<file1>-OPEN* Sequential-Read-OPEN-CURSOR WHEN *<file1>-READ-RK* Sequential-Read-READ-CURSOR WHEN *<file1>-START-RK* Sequential-Read-START-CURSOR WHEN 0 EXEC SQL OPEN *<file1>-READ-RK-CURSOR.* END-EXEC Sequential-Read-READ-CURSOR MOVE 2 TO *<file1>-CURSOR* END-EVALUATE
CLOSE	Close the opened cursors. Set the flag to indicate that there are no cursors available.	IF *<file1>-OPEN-CURSOR* THEN EXEC SQL CLOSE *<file1>-OPEN-CURSOR* END-EXEC END-IF. IF *<file1>-READ-RK-CURSOR* THEN EXEC SQL CLOSE *<file1>-READ-RK-CURSOR* END-EXEC END-IF. IF *<file1>-START-RK-CURSOR* THEN EXEC SQL CLOSE *<file1>-START-RK-*

		CURSOR END-EXEC END-IF. MOVE 0 TO <*file1*>-*CURSOR*

3.1.5.2 I-O

3.1.5.2.1 Case 2

This section specifies the translation strategy for VSAM files with indexed organization, dynamic access mode opened in I-O mode where READ exists, RANDOM READ exists and START does not exist.

Define a cursor for each VSAM file, to select all the rows in the corresponding table.

If RANDOM READ is used and if it uses record key in key comparison, define a cursor to select all the rows in the corresponding table where key columns have values greater than record key values.

Define the above cursors to be ORDER BY primary key columns.

Define a condition name in the working storage section, to be set and reset by RANDOM READ and OPEN statements when they open and close the cursor. This condition name will indicate the current opened cursor to be used by other operations.

Definition of the condition name will be

01 <*file1*>-*CURSOR* PIC 9 VALUE 0.
88 <*file1*>-*OPEN* value 1.
88 <*file1*>-*READ-RK* value 2.

Where <*file1*>-*OPEN* will be set when the cursor corresponding to OPEN is opened
<file2>-READ-RK will be set when the cursor corresponding to READ is opened.

Cursor definition for OPEN :

OPEN-CURSOR

Cursor definition for READ :

READ-CURSOR

Operation	**Description**	**Translated Code**
OPEN	Open the cursor <*file1*>-*OPEN-CURSOR* defined corresponding to OPEN statement.	EXEC SQL OPEN <*file1*>-*OPEN-CURSOR* END-EXEC. MOVE 1 TO <*file1*>-*CURSOR*.
RANDOM READ	RANDOM READ translates to SELECT with WHERE clause. As RANDOM READ repositions the file position indicator, any previously opened cursor should be closed to prevent operations following RANDOM READ, from using the cursor.	Random-Read EVALUATE <*file1*>-*CURSOR* WHEN <*file1*>-*OPEN* EXEC SQL CLOSE <*file1*>-*OPEN-CURSOR* END-EXEC WHEN <*file1*>-*READ-RK* EXEC SQL CLOSE <*file1*>-*READ-RK-CURSOR* END-EXEC END-EVALUATE

		MOVE 0 TO *<file1>-CURSOR*
READ NEXT	READ NEXT is a sequential operation and should follow successful OPEN or RANDOM READ or READ NEXT. If the previous operation is OPEN, then that would have opened the appropriate cursor and hence FETCH on the cursor corresponding to OPEN will get the correct row. If the previous operation is RANDOM READ, then that would have selected the correct record. Record key fields(host variables) would have got primary key column values of the selected record. Opening the cursor, *<file1>-READ-RK-CURSOR* , will select all the rows in the corresponding table with primary key columns have values greater than record key values and FETCH on that cursor will get the correct record. If the previous operation is READ NEXT, it would have opened the appropriate cursor and hence fetch on that cursor will get the correct record.	EVALUATE *<file1>-CURSOR* WHEN *<file1>-OPEN* Sequential-Read-OPEN-CURSOR WHEN *<file1>-READ-RK* Sequential-Read-READ-CURSOR WHEN 0 EXEC SQL OPEN *<file1>-READ-RK-CURSOR*. END-EXEC Sequential-Read-READ-CURSOR MOVE 2 TO *<file1>-CURSOR* END-EVALUATE
REWRITE	Record to be rewritten, is specified by the value contained in the prime RECORD KEY. REWRITE will translate to UPDATE with WHERE clause.	Random-Rewrite
DELETE	Record to be deleted is specified by the value contained in the prime RECORD KEY. DELETE will translate to DELETE with WHERE clause.	Delete
WRITE	Record to be written is specified by the value contained in the prime RECORD KEY. WRITE will translate to INSERT with WHERE clause.	Insert
CLOSE	Close the opened cursors. Set the flag to indicate that there are no cursors available.	IF *<file1>-OPEN-CURSOR* THEN EXEC SQL CLOSE *<file1>-OPEN-CURSOR* END-EXEC END-IF. IF *<file1>-READ-RK-CURSOR* THEN EXEC SQL CLOSE *<file1>-READ-RK-CURSOR* END-EXEC END-IF. MOVE 0 TO *<file1>-CURSOR*

3.1.5.2.1 Case 4

Define a cursor for each VSAM file, to select all the rows in the corresponding table.

If RANDOM READ is used and if it uses record key in key comparison, define a cursor to select all the rows in the corresponding table where key columns have values greater than record key values.

If START is used and if it uses record key in key comparison, define a cursor to select all the rows in the corresponding table where key columns have values greater than or equal to the record key values.

Define the above cursors to be ORDER BY primary key columns.

Define a condition name in the working storage section, to be set and reset by RANDOM READ, START and OPEN statements when they open and close the cursor. This condition name will indicate the current opened cursor to be used by other operations.

Definition of the condition name will be

01 *<file1>-CURSOR* PIC 9 VALUE 0.
88 *<file1>-OPEN* value 1.
88 *<file1>-READ-RK* value 2.
88 *<file1>-START-RK* value 3.

Where *<file1>-OPEN* will be set when the cursor corresponding to OPEN is opened,
<file1>-READ-RK will be set when the cursor corresponding to READ is opened,
<file1>-START-RK will be set when the cursor corresponding to START is opened.

Cursor definition for OPEN :

OPEN-CURSOR

Cursor definition for READ :

READ-CURSOR

Cursor definition for START :

START-CURSOR

Operation	Description	Translated Code
OPEN	Open the cursor *<file1>-OPEN-CURSOR*, defined corresponding to OPEN statement.	EXEC SQL OPEN *<file1>-OPEN-CURSOR* END-EXEC. MOVE 1 TO *<file1>-CURSOR*.
START	Start repositions the file position indicator by populating relative key fields with appropriate values. If the corresponding cursor is already opened, close the cursor and open it again. Reopening the cursor will select the rows based on the current value of the host variables used in the WHERE clause. Set <file1>-START-RK-CURSOR to be the current valid cursor which can be used by successive I/O operations till another START/RANDOM READ repositions the file position indicator or CLOSE closes the file.	IF <file1>-START-RK THEN EXEC SQL CLOSE *<file1>-START-RK-CURSOR* END-EXEC END-IF. EXEC SQL OPEN *<file1>-START-RK-CURSOR* END-EXEC. MOVE 3 TO *<file1>-CURSOR*.
RANDOM READ	RANDOM READ translates to SELECT with WHERE clause. As RANDOM READ repositions the file position indicator, any previously opened cursor should be closed to prevent operations following RANDOM READ, from using the cursor.	Random-Read EVALUATE *<file1>-CURSOR* WHEN *<file1>-OPEN* EXEC SQL CLOSE *<file1>-OPEN-CURSOR* END-EXEC WHEN *<file1>-READ-RK* EXEC SQL CLOSE *<file1>-READ-RK-CURSOR*

		END-EXEC WHEN *<file1>-START-RK* EXEC SQL CLOSE *<file1>-START-RK-CURSOR* END-EXEC END-EVALUATE MOVE 0 TO *<file1>-CURSOR*
READ NEXT	READ NEXT is a sequential operation and should follow either successful OPEN or RANDOM READ or READ NEXT or START. If the previous operation is OPEN or START, then that would have opened the appropriate cursor and hence FETCH on the cursor corresponding to OPEN or START will get the correct row. If the previous operation is RANDOM READ, then that would have selected the correct record. Record key fields(host variables) would have got primary key column values of the selected record. Opening the cursor, *<file1>-READ-RK-CURSOR* , will select all the rows in the corresponding table with primary key columns have values greater than record key values and FETCH on that cursor will get the correct record. If the previous operation is READ NEXT, it would have opened the appropriate cursor and hence fetch on that cursor will get the correct record.	EVALUATE *<file1>-CURSOR* WHEN *<file1>-OPEN* Sequential-Read-OPEN-CURSOR WHEN *<file1>-READ-RK* Sequential-Read-READ-CURSOR WHEN *<file1>-START-RK* Sequential-Read-START-CURSOR WHEN 0 EXEC SQL OPEN *<file1>-READ-RK-CURSOR.* END-EXEC Sequential-Read-READ-CURSOR MOVE 2 TO *<file1>-CURSOR* END-EVALUATE
REWRITE	Record to be rewritten is specified by the value contained in the prime RECORD KEY. REWRITE will translate to UPDATE with WHERE clause.	Random-Rewrite
DELETE	Record to be deleted is specified by the value contained in the prime RECORD KEY. DELETE will translate to DELETE with WHERE clause.	Delete
WRITE	Record to be written is specified by the value contained in the prime RECORD KEY. WRITE will translate to INSERT with WHERE clause.	Insert
CLOSE	Close the opened cursors.	IF *<file1>-OPEN-CURSOR* THEN EXEC SQL CLOSE *<file1>-OPEN-CURSOR* END-EXEC END-IF. IF *<file1>-READ-RK-CURSOR* THEN EXEC SQL CLOSE *<file1>-READ-RK-CURSOR* END-EXEC END-IF. IF *<file1>-START-RK-CURSOR* THEN

		EXEC SQL CLOSE <*file1*>-*START-RK-CURSOR* END-EXEC END-IF. MOVE 0 TO <*file1*>-*CURSOR*

3.1.6.3 OUTPUT

In this mode, there will be no READ / READ NEXT / START statements. Only allowed operations are OPEN, WRITE and CLOSE.

Operation	**Description**	**Translated Code**
OPEN	Comment OPEN statement.	None
WRITE	Insert the record into the corresponding table.	Insert
CLOSE	Comment CLOSE statement	None

3.2 Online Program Translation

Each original program with embedded VSAM access call is assumed to have following template structure. Template also identifies some record structures that will be added to the working storage section of the program.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. Cicsprgm.

ENVIRONMENT DIVISION.

DATA DIVISION.

WORKING-STORAGE SECTION.

01      DATA-RECORD.
        05 DATA-FIELD-1  ….
        05 DATA-FIELD-2  ….
                .
                .

01      KEY-RECORD.
        05 KEY-FIELD-1  ….
        05 KEY-FIELD-2 ….

01      DATA-REC-LENGTH….
1       KEY-REC-LENGTH….
1       TOKEN-FIELD….
1       REQID-FILED…
1       COUNT-FIELD…
01      NUMERIC-COUNT…

**** record structures below needs to be added in the new code ****

 1      DCLGEN-RECORD.
        5       DCLGEN-KEY.
                10      DCLGEN-KEY-FIELD1
                10      DCLGEN-KEY-FIELD2
                .
                .
        5       DCLGEN-DATA.
                10      DCLGEN-DATA-FIELD1
                10      DCLGEN-DATA-FIELD2
                .
                .

/* this definition needs to be repeated appropriately */
/* for each TOKEN used in program                     */
01      TOKEN-DCLGEN-KEY.
        5       TOKEN-DCLGEN-KEY-FIELD1
        5       TOKEN-DCLGEN-KEY-FIELD2
        .
        .
```

```
/*Indicator to identify if the cursor is defined for a READNEXT or READPREV operation */
01      WS-FUNCTION-CURSOR          PIC X(01)       VALUE ' '.
        88      WS-NEXT-CURSOR                      VALUE 'N'.
        88      WS-PREV-CURSOR                      VALUE 'P'.
        88      WS-NO-CURSOR                        VALUE ' '.

/* this definition needs to be repeated appropriately for each RESETBR on a VSAM file using different
browse criteria */

01      WS-FUNCTION-CURSOR-1        PIC X(01)       VALUE ' '.
        88      WS-NEXT-CURSOR-1                    VALUE 'N'.
        88      WS-PREV-CURSOR-1                    VALUE 'P'.
        88      WS-NO-CURSOR-1                      VALUE ' '.




****                    End of the new addition                 ****

PROCEDURE DIVISION.
        .
        .
        EXEC CICS
                Vsam access call
        END-EXEC.
        .
        .
```

Discussion below will assume this program template and will identify necessary changes to the code corresponding to translation of each command.

3.2.1 READ

3.2.1.1 Function

Read a record from a file.

3.2.1.2 CICS Syntax

EXEC CICS

READ

FILE (vsam-filename)	M	file to be read
INTO (DATA-RECORD)	M	data area where to read the data
LENGTH (DATA-REC-LENGTH)	O	CICS will populate with actual record length
RIDFLD (KEY-RECORD)	M	key record, This will identify record to be read
KEYLENGTH (KEY-REC-LENGTH)	O	no of position of key to be matched for partial/generic search (mandatory when generic option is specified)
GENERIC	O	partial key match
GTEQ / EQUAL	O	match key for Equal or Greater than or Equal condition
RBA	O	when this option is specified RIDFLD contains actual relative byte address
RRN	O	when this option is specified RIDFLD contains actual Relative Record Number
UPDATE	O	read record will be used for UPDATE/REWRITE
TOKEN (TOKEN-FIELD)	O	used to identify multiple records read for UPDATE/REWRITE

END-EXEC

3.2.1.1 Translation of Syntax

If Relational operator in the READ is '='

.

MOVE KEY-RECORD TO DCLGEN-KEY. {need to move appropriate fields depending on sizes }

```
EXEC    SQL
        SELECT          *
        INTO            { :DCLGEN-RECORD fields…. }
        FROM            {  db2-tablename corresponding to vsam-filename }
        WHERE           {
                                If  ( KEYLENGTH  is specified )
                                {
                                        identify exact fields from KEY-RECORD;
                                        corresponding to these fields identify DCLGEN-KEY fields
                                }
                                else
                                {
                                        all fields of DCLGEN-KEY will participate in WHERE clause
                                }

                                using  DB2 column names and corresponding DCLGEN-KEY fields as
                                identified before and by using relational operator form the WHERE
                                clause
                        }
```

If Relational operator in the READ is '>='

Declare a cursor to select from the corresponding DB2 table using the key fields and key values in the WHERE clause. Since this cursor will be used only to read the first record matching the WHERE clause, The OPTIMIZE FOR 1 ROW option will be used to improve performance.

Cursor declaration will be as follows :

```
EXEC SQL DECLARE cursor-name CURSOR FOR
        SELECT          *
        FROM            db2-table-name
        WHERE           db2-column-name >= : dclgen-record fields
        OPTIMIZE FOR 1 ROW
        FOR FETCH ONLY
        END-EXEC.
```

Translation for the READ will be as follows :

```
MOVE  KEY-RECORD   TO      DCLGEN-KEY. {need to move appropriate fields depending on sizes }

EXEC    SQL
        OPEN cursor-name
        END-EXEC.

EXEC SQL
        FETCH cursor-name
        INTO    : dclgen-record fields
        END-EXEC.

EXEC SQL
        OPEN cursor-name
        END-EXEC.
```

For all the above cases:

```
IF ( READ is for UPDATE )
{
        /* if token is specified explicitly in program find appropriate TOKEN-DCLGEN-KEY */
        /* correspondingly.                                                              */
        MOVE  DCLGEN-KEY  to        TOKEN-DCLGEN-KEY
}
```

3.2.2 STARTBR

3.2.2.1 Function

Start browse of a file. This is used to position the file position indicator at the desired record so that VSAM read can be done from this record onwards.

3.2.2.2 CICS Syntax

EXEC CICS

STARTBR		
FILE (VSAM-FILENAME)	M	file to be read
RIDFLD (KEY-RECORD)	M	key record, need to be populated before call
KEYLENGTH (KEY-REC-LENGTH)	O	no of position of key to be matched for partial/generic search (mandatory when generic option is specified)
GENERIC	O	partial key match
GTEQ / EQUAL	O	match key for Equal or Greater than or Equal condition
RBA	O	when this option is specified RIDFLD contains actual relative byte address
RRN	O	when this option is specified RIDFLD contains actual Relative Record Number
REQID (REQID-FIELD)	O	used to identify is there are more than one browse on same file

END-EXEC

3.2.2.3 Translation of Syntax

```
.
.
IF ( REQID-FIELD is specified )
{
```

```
        use it in determining  NEXT-CURSORNAME and PREV-CURSORNAME
}

If  ( KEYLENGTH  is specified )
{
        identify exact fields from KEY-RECORD;
        corresponding to these fields identify DCLGEN-KEY fields
}
else
{
        all fields of DCLGEN-KEY will participate in WHERE clause
}
```

CURSOR DECLARATION

If only READNEXT operation is present in the program for a VSAM file, then declare only the cursor for readnext (NEXT-CURSORNAME). If only READPREV operation is present in the program for a VSAM file then declare only the cursor for readprev (PREV-CURSORNAME). Both the cursors will be declared if READPREV and READNEXT is being done on a file in the program.

```
EXEC SQL
        DECLARE CURSOR NEXT-CURSORNAME FOR
                SELECT          *
                FROM            { db2-tablename corresponding to vsam-filename }
                WHERE           { appropriate where clause with tablefields and >= operator }
END-EXEC

EXEC SQL
        DECLARE CURSOR PREV-CURSORNAME FOR
                SELECT          *
                FROM            { db2-tablename corresponding to vsam-filename }
                WHERE           { appropriate where clause with tablefields and <= operator }
END-EXEC
```

REPLACING THE STARTBR COMMAND

The WHERE EXISTS clause is used first to do a SELECT on the table. If any rows are found with matching condition then only the cursor is opened. For this purpose the Standard Dummy table SYSIBM.SYSDUMMY1 is used.
The translated code will look as given below :

```
        EXEC SQL SELECT IBMREQD FROM SYSIBM.SYSDUMMY1
                                INTO :WS-IBMREQD
                                WHERE EXISTS
                                (SELECT * FROM db2-table-name
                                        WHERE db2-column-name = (or >=  ) : dclgen-record fields)
                                END-EXEC.
```

The STARTBR is translated to a OPEN CURSOR assuming that the next function will be READNEXT. This is due to the fact that the same STARTBR command can be used either for a READNEXT or READPREV function, and this will not be known at the time of translating the STARTBR to the OPEN CURSOR.

```
IF SQLCODE = +0
{
        Move the VSAM key values to the dclgen-record fields to use in the WHERE clause
        SET  WS-NEXT-CURSOR TO TRUE
        EXEC    SQL
                OPEN NEXT-CURSORNAME
        END-EXEC
}
```

3.2.3 READNEXT / READPREV

3.2.3.1 Function

Read next record or previous record during a browse of a file.

3.2.3.2 CICS Syntax

EXEC CICS

READNEXT / READPREV

FILE (vsam-filename)	M	file to be read
INTO (DATA-RECORD)	M	data area where to read the data
LENGTH (DATA-REC-LENGTH)	O	CICS will populate with actual record length
RIDFLD (KEY-RECORD)	M	key record size should be sufficient to hold complete key
KEYLENGTH (KEY-REC-LENGTH)	O	no of position of key to be matched for partial/generic search (mandatory when generic option is specified)
RBA	O	when this option is specified RIDFLD contains actual relative byte address
RRN	O	when this option is specified RIDFLD contains actual Relative record Number
REQID (REQID-FIELD)	O	used to identify is there are more than one browse on same file

END-EXEC

3.2.3.3 Translation of Syntax

FOR READNEXT COMMAND

```
IF WS-NEXT-CURSOR
(       EXEC   SQL
               FETCH  NEXT-CURSORNAME
               INTO   :DCLGEN-RECORD
        END-EXEC
)
ELSE
        IF WS-PREV-CURSOR
        (
               EXEC   SQL
                      CLOSE PREV-CURSORNAME
               END-EXEC

               Move the VSAM key values to the dclgen-record fields to use in the WHERE clause
               SET WS-NEXT-CURSOR TO TRUE
               EXEC SQL
                      OPEN NEXT-CURSORNAME
               END-EXEC

               EXEC   SQL
                      FETCH  NEXT-CURSORNAME
                      INTO   :DCLGEN-RECORD
               END-EXEC
        )
END-IF

* this logic is to take care of the RESETBR done on the file, with different criteria than the STARTBR

IF WS-NEXT-CURSOR-1
(       EXEC   SQL
               FETCH  NEXT-CURSORNAME -1
               INTO   :DCLGEN-RECORD
        END-EXEC
)
ELSE
        IF WS-PREV-CURSOR-1
        (
               EXEC   SQL
                      CLOSE PREV-CURSORNAME-1
               END-EXEC

               Move the VSAM key values to the dclgen-record fields to use in the WHERE clause
               SET WS-NEXT-CURSOR TO TRUE
               EXEC SQL
                      OPEN NEXT-CURSORNAME-1
               END-EXEC

               EXEC   SQL
                      FETCH  NEXT-CURSORNAME-1
```

```
                    INTO    :DCLGEN-RECORD
                END-EXEC
        )
END-IF


FOR READPREV COMMAND

IF WS-PREV-CURSOR
(       EXEC    SQL
                FETCH  PREV-CURSORNAME
                INTO    :DCLGEN-RECORD
        END-EXEC
)
ELSE
        IF WS-NEXT-CURSOR
        (
                EXEC    SQL
                        CLOSE NEXT-CURSORNAME
                END-EXEC

                Move the VSAM key values to the dclgen-record fields to use in the WHERE clause
                SET WS-PREV-CURSOR-1 TO TRUE
                EXEC SQL
                        OPEN PREV-CURSORNAME
                END-EXEC

                EXEC    SQL
                        FETCH  PREV-CURSORNAME
                        INTO    :DCLGEN-RECORD
                END-EXEC
        )
END-IF

* this logic is to take care of the RESETBR done on the file, with different criteria than the STARTBR

IF WS-PREV-CURSOR-1
(       EXEC    SQL
                FETCH  PREV-CURSORNAME-1
                INTO    :DCLGEN-RECORD
        END-EXEC
)
ELSE
        IF WS-NEXT-CURSOR-1
        (
                EXEC    SQL
                        CLOSE NEXT-CURSORNAME-1
                END-EXEC

                Move the VSAM key values to the dclgen-record fields to use in the WHERE clause
                SET WS-PREV-CURSOR-1 TO TRUE
                EXEC SQL
                        OPEN PREV-CURSORNAME-1
                END-EXEC

                EXEC    SQL
```

```
                    FETCH  PREV-CURSORNAME-1
                    INTO    :DCLGEN-RECORD
                END-EXEC
        )
END-IF



MOVE  DCLGEN-RECORD       TO       DATA-RECORD.
.
.
```

3.2.4 WRITE

3.2.4.1 Function

Write a record using the key values specified.

3.2.4.2 CICS Syntax

EXEC CICS

WRITE

FILE (vsam-filename)	M	file to be read
FROM (DATA-RECORD)	M	data area from where to record needs to be written
LENGTH (DATA-REC-LENGTH)	O	length of data record to be written
RIDFLD (KEY-RECORD)	M	key record, need to be populated before call
KEYLENGTH (KEY-REC-LENGTH)	O	generally this is specified as length of complete key field
RBA	O	when this option is specified RIDFLD contains actual relative byte address
RRN	O	when this option is specified RIDFLD contains actual Relative Record Number

END-EXEC

3.2.4.3 Translation of Syntax

```
.
.
MOVE  DATA-RECORD          TO      DCLGEN-RECORD.

EXEC  SQL
      INSERT        INTO    { db2-tablename corresponding to vsam-filename }
      VALUES        ARE     {       :DCLGEN-RECORD fields }
END-EXEC
```

3.2.5 REWRITE

3.2.5.1 Function

Update a record in a file for the key values specified. REWRITE can be done after executing a READ FOR UPDATE or it can be a random REWRITE also.

3.2.5.2 CICS Syntax

EXEC CICS

REWRITE		
FILE (vsam-filename)	M	file to be read
FROM (DATA-RECORD)	M	data area from where to record needs to be written
LENGTH (DATA-REC-LENGTH)	O	length of data record to be rewritten
TOKEN (TOKEN-FIELD)	O	used to identify multiple records read for UPDATE/REWRITE

END-EXEC

3.2.5.3 Translation of Syntax

```
.
.
MOVE  DATA-RECORD          TO      DCLGEN-RECORD

If  ( token is explicitly specified in command}
{
        use it to identify corresponding TOKEN-DCLGEN-KEY
        which can be used in where clause
}

EXEC    SQL
        UPDATE      { db2-tablename corresponding to vsam-filename }
        SET         db2-column-names = :DCLGEN-RECORD fields
        WHERE       db2-column-names =  :TOKEN-DCLGEN-KEY or DCLGEN-RECORD-KEY
END-EXEC
.
```

3.2.6 DELETE

3.2.6.1 Function

Delete a record from a file

3.2.6.2 CICS Syntax

EXEC CICS

DELETE

FILE (vsam-filename)	M	file for delete
RIDFLD (KEY-RECORD)	O	key record, need to be populated before call
KEYLENGTH (KEY-REC-LENGTH)	O	no of position of key to be matched for partial/generic match (mandatory when generic option is specified)
GENERIC	O	partial key match
RBA	O	when this option is specified RIDFLD contains actual relative byte address
RRN	O	when this option is specified RIDFLD contains actual Relative Record Number
TOKEN (TOKEN-FIELD)	O	used to identify record to be deleted which was identified by read for UPDATE/REWRITE
NUMERIC (NUMERIC-COUNT)	O	This variable is populated with number of records deleted

END-EXEC

3.2.6.3 Translation Of Syntax

```
IF ( Token is Specified  )
{
        MOVE   TOKEN-DCLGEN-KEY   TO        DCLGEN-KEY.
}

IF ( RIDFLD is specified )
{

        MOVE   KEY-RECORD    to DCLGEN-KEY ;
```

```
        If  ( KEYLENGTH  is specified )
        {
                identify exact fields from KEY-RECORD;
                corresponding to these fields identify DCLGEN-KEY fields
        }
        else
        {
                all fields of DCLGEN-KEY will participate in WHERE clause
        }
}
```

Relational Operator is =

using DB2 columnnames and corresponding DCLGEN-KEY fields as identified before and by using relational operator , form the WHERE clause ;

```
EXEC    SQL
        DELETE (db2-tablename corresponding to vsam-filename )
        WHERE  (  where clause as formated before )
END-EXEC

MOVE  SQLERRD(3)     TO        NUMERIC-COUNT
```

3.2.7 UNLOCK

3.2.7.1 Function

Release exclusive control of the file. UNLOCK file is usually done after a READ FOR UPDATE is executed on the file and it is decided that no update needs to be done to the record. UNLOCK also needs to be performed after VSAM MASSINSERT.

3.2.7.2 CICS Syntax

EXEC CICS

UNLOCK

FILE (vsam-filename)	M	file to be unlocked
TOKEN (TOKEN-FIELD)	O	Explicitly specified READ for update to be unlocked

END-EXEC

3.2.7.3 Translation of Syntax

No SQL statement will be added for this because DB2 takes care of data locking depending on the bind options used. Any resources locked will be released after a COMMIT.
Since the READ with UPDATE option is being translated to a SQL SELECT, and the corresponding REWRITE is being translated to the an UPDATE on the table, the scenario of the FILE UNLOCK does not appear in the table operation.

Replace this command with initialization of Temporary Storage Records for READ with UPDATE.
If specific TOKEN is specified then initialize the specific record corresponding to that TOKEN.

3.2.8 ENDBR

3.2.8.1 Function

End browse of a file.

3.2.8.2 CICS Syntax

```
EXEC CICS
        ENDBR
        FILE ( vsam-filename )                  M     file to be read
        REQID ( REQID-FIELD )                   O     used to identify if there are more
                                                      than one browse on same file
END-EXEC
```

3.2.8.3 Translation Of Syntax

```
IF ( REQID-FIELD is specified )
{
        use it in determining  NEXT-CURSORNAME and PREV-CURSORNAME
}

If  ( READNEXT operation exists)
{
EXEC    SQL
        CLOSE  NEXT-CURSORNAME
END-EXEC
}
else
        If  ( READPREV operations exists )
        {
                EXEC    SQL
                        CLOSE  PREV-CURSORNAME
                        END-EXEC
        }
```

3.2.9 RESETBR

3.2.9.1 Function

Reset start of browse. The file position indicator is set to the position at the record with the new key values specified.

3.2.9.2 CICS Syntax

EXEC CICS

RESETBR

FILE (VSAM-FILENAME)	M	file to be read
RIDFLD (KEY-RECORD)	M	key record, need to be populated before call
KEYLENGTH (KEY-REC-LENGTH)	O	no of position of key to be matched for partial/generic search (mandatory when generic option is specified)
GENERIC	O	partial key match
GTEQ / EQUAL	O	match key for Equal or Greater than or Equal condition
RBA	O	when this option is specified RIDFLD contains actual relative byte address
RRN	O	when this option is specified RIDFLD contains actual Relative Record Number
REQID (REQID-FIELD)	O	used to identify is there are more than one browse on same file

END-EXEC

3.2.9.3 Translation Of Syntax

```
IF ( REQID-FIELD is specified )
{
        use it in determining  NEXT-CURSORNAME and PREV-CURSORNAME
}
```

Move the key values specified in the RESETBR to the DCLGEN fields.

CURSOR DECLARATION

The RESETBR can use either the same criteria that the STARTBR used or it can reset the file position indicator using some other criteria, example, the STARTBR may be having a EQUAL condition, but the

RESETBR can use a GTEQ condition. Verify if the conditions specified in the RESETBR is different than in the STARTBR. If the conditions are different then new cursors will have to be defined for RESETBR using the new criteria.

In case new cursors have to be defined, if only READNEXT operation is present in the program for the VSAM file, then declare only the cursor for readnext (NEXT-CURSORNAME). If only READPREV operation is present in the program for a VSAM file then declare only the cursor for readprev (PREV-CURSORNAME). Both the cursors will be declared if READPREV and READNEXT is being done on a file in the program.

```
EXEC SQL
        DECLARE CURSOR NEXT-CURSORNAME-1 FOR
                SELECT          *
                FROM            { db2-tablename corresponding to vsam-filename }
                WHERE           { appropriate where clause with tablefields and >= operator }
END-EXEC

EXEC SQL
        DECLARE CURSOR PREV-CURSORNAME-1 FOR
                SELECT          *
                FROM            { db2-tablename corresponding to vsam-filename }
                WHERE           { appropriate where clause with tablefields and <= operator }
END-EXEC
```

REPLACING THE RESETBR COMMAND

```
If WS-NEXT-CURSOR
{
SET WS-NO-CURSOR TO TRUE
EXEC   SQL
        CLOSE  NEXT-CURSORNAME
END-EXEC
}
else
        If  WS-PREV-CURSOR
        {
                SET WS-NO-CURSOR TO TRUE
                EXEC   SQL
                        CLOSE  PREV-CURSORNAME
                END-EXEC
        }.

If  WS-NEXT-CURSOR-1
{
SET WS-NO-CURSOR-1 TO TRUE
EXEC   SQL
        CLOSE  NEXT-CURSORNAME-1
END-EXEC
}
else
        If  WS-PREV-CURSOR-1
        {
                SET WS-NO-CURSOR-1 TO TRUE
                EXEC   SQL
                        CLOSE  PREV-CURSORNAME-1
```

```
            END-EXEC
}.
```

The WHERE EXISTS clause is used first to do a SELECT on the table. If any rows are found with matching condition then only the cursor is opened. For this purpose the Standard Dummy table SYSIBM.SYSDUMMY1 is used.
The translated code will look as given below :

```
EXEC SQL SELECT IBMREQD FROM SYSIBM.SYSDUMMY1
                        INTO :WS-IBMREQD
                        WHERE EXISTS
                        (SELECT * FROM db2-table-name
                                WHERE db2-column-name = (or >= ) : dclgen-record fields)
                        END-EXEC.
```

The RESETBR is translated to a OPEN CURSOR assuming that the next function will be READNEXT. This is due to the fact that the same RESETBR command can be used either for a READNEXT or READPREV function, and this will not be known at the time of translating the RESETBR to the OPEN CURSOR.

```
IF SQLCODE = +0
{
        Move the VSAM key values to the dclgen-record fields to use in the WHERE clause
        SET  WS-NEXT-CURSOR-1 TO TRUE
        EXEC    SQL
                OPEN NEXT-CURSORNAME
        END-EXEC
}
```

4 Optimizing For Performance

4.1 Selecting only the required fields

It is preferred that the only the required columns are selected in the query.

4.2 Adding Commit logic

Batch programs which handle large volume of data and do updates, inserts or deletes on the DB2 tables will have better performance if commit logic is added to commit the changes after the number of rows processed reaches a predetermined value. However for any cursor operations for which the commit logic is used should have the cursor defined with option WITH HOLD, otherwise the cursor position is lost after the commit.

4.3 Update only the required columns

Only columns required to be updated should be retained in the update statements. This is especially true if the column is indexed and is being updated unnecessarily.

5 Appendix

5.1.1 Sequential Read OPEN-CURSOR

```
EXEC SQL
        FETCH <file1>-OPEN-CURSOR
        INTO <respective-fields>
END-EXEC
IF SQLCODE = 100
THEN
        <at-end-imperative-stmts> | NEXT SENTENCE
ELSE
        IF SQLCODE = 0
        THEN
                EXEC SQL
                        OPEN CURSOR dependant-cursor-1
                END-EXEC
                MOVE 1 TO <db2-count>
                PERFORM FETCH-PARA-1 UNTIL <db2-count> > <db2-array-1-size> AND
        SQLCODE = 100
                EXEC SQL
                        CLOSE CURSOR dependant-cursor-1
                END-EXEC
                ...
                EXEC SQL
                        OPEN CURSOR dependant-cursor-n
                END-EXEC
                MOVE 1 TO <db2-count>
                PERFORM FETCH-PARA-n UNTIL <db2-count> > <db2-array-n-size> AND
        SQLCODE =100
                EXEC SQL
                        CLOSE CURSOR dependant-cursor-n
                END-EXEC
                <not-at-end-imperative-stmts>
        ELSE
                DISPLAY " ERROR IN FETCH "
        END-IF
END-IF.
```

5.1.2 Sequential Read START-CURSOR

```
EXEC SQL
        FETCH <file1>-START-CURSOR
        INTO <respective-fields>
END-EXEC
IF SQLCODE = 100
THEN
        <at-end-imperative-stmts> | NEXT SENTENCE
ELSE
        IF SQLCODE = 0
        THEN
                EXEC SQL
                        OPEN CURSOR dependant-cursor-1
                END-EXEC
                MOVE 1 TO <db2-count>
                PERFORM FETCH-PARA-1 UNTIL <db2-count> > <db2-array-1-size> AND
        SQLCODE = 100
                EXEC SQL
                        CLOSE CURSOR dependant-cursor-1
                END-EXEC
                ...
                EXEC SQL
                        OPEN CURSOR dependant-cursor-n
                END-EXEC
                MOVE 1 TO <db2-count>
                MOVE <array-n-size> TO <DB2-ARRAY-n-SIZE>
                PERFORM FETCH-PARA-n UNTIL <db2-count> <= <db2-array-n-size> AND
        SQLCODE = 100
                EXEC SQL
                        CLOSE CURSOR dependant-cursor-n
                END-EXEC
                <not-at-end-imperative-stmts>
        ELSE
                DISPLAY " ERROR IN FETCH "
        END-IF
END-IF.
```

5.1.3 Sequential Read READ-CURSOR

```
EXEC SQL
        FETCH <file1>-READ-CURSOR
        INTO <respective-fields>
END-EXEC
IF SQLCODE = 100
THEN
        <at-end-imperative-stmts> | NEXT SENTENCE
ELSE
        IF SQLCODE = 0
        THEN
                EXEC SQL
                        OPEN CURSOR dependant-cursor-1
                END-EXEC
                MOVE 1 TO <db2-count>
                PERFORM FETCH-PARA-1 UNTIL <db2-count> > <db2-array-1-size> AND
        SQLCODE = 100
                EXEC SQL
                        CLOSE CURSOR dependant-cursor-1
                END-EXEC
                ...
                EXEC SQL
                        OPEN CURSOR dependant-cursor-n
                END-EXEC
                MOVE 1 TO <db2-count>
                PERFORM FETCH-PARA-n UNTIL <db2-count> > <db2-array-n-size> AND
        SQLCODE = 100
                EXEC SQL
                        CLOSE CURSOR dependant-cursor-n
                END-EXEC
                <not-at-end-imperative-stmts>
        ELSE
                DISPLAY " ERROR IN FETCH "
        END-IF
END-IF.
```

5.1.4 Random Read

```
EXEC SQL
        SELECT <table-1-col-1> ... <table-1-col-n>,
                ...,
                <table-n-col-1>...<table-n-col-n>
        FROM <table-1> ...<table-n>
        INTO <respective-fields>
END-EXEC
IF SQLCODE = 0
THEN
        EXEC SQL
                OPEN CURSOR dependant-cursor-1
        END-EXEC
        MOVE 1 TO <db2-count>
        PERFORM FETCH-PARA-1 UNTIL <db2-count> > <db2-array-1-size> AND SQLCODE =
        100
        EXEC SQL
                CLOSE CURSOR dependant-cursor-1
        END-EXEC
        ...
        EXEC SQL
                OPEN CURSOR dependant-cursor-n
        END-EXEC
        MOVE 1 TO <db2-count>
        PERFORM FETCH-PARA-n UNTIL <db2-count> > <db2-array-n-size> AND SQLCODE = 100
        EXEC SQL
                CLOSE CURSOR dependant-cursor-n
        END-EXEC
        <not-invalid-key-imperative-stmts>
ELSE
        <invalid-key-imperative-statements> | NEXT SENTENCE
END-IF.
```

5.1.5 Insert

```
EXEC SQL
        INSERT INTO <table-1>(<table-1-col-1> ... <table-1-col-n>)
        VALUES (<respective-fields>)
END-EXEC.
IF SQLCODE = 0
THEN
        EXEC SQL
                INSERT INTO <table-2>(<table-2-col-1> ... <table-2-col-n>)
                VALUES (<respective-fields>)
        END-EXEC
END-IF
...
IF SQLCODE = 0
THEN
        EXEC SQL
                INSERT INTO <table-n>(<table-n-col-1> ... <table-n-col-n>)
                VALUES (<respective-fields>)
        END-EXEC
END-IF
IF SQLCODE = 0
THEN
        MOVE 1 TO <db2-count>
        PERFORM INSERT-PARA-1 UNTIL <db2-count> > <db2-array-1-size> AND SQLCODE = 0
        ...
        MOVE 1 TO <db2-count>
        PERFORM INSERT-PARA-n UNTIL <db2-count> > <db2-array-n-size> AND SQLCODE = 0
END-IF
IF SQLCODE = 0
THEN
        <not-invalid-key-statements> | NEXT SENTENCE
ELSE
        <invalid-key-statements> | DISPLAY “Error In Insert “
END-IF
```

5.1.6 Random Rewrite

```
EXEC SQL
        UPDATE <table-1>
        SET     <table-1-col-1> = <table-1-col-1-fld-1>,
                ...
                <table-1-col-n> = <table-1-col-n-fld-n>
        WHERE <table-1-key-col-1> = <table-1-key-col-1-key-fld-1>,
                ...
                <table-1-key-col-n> = <table-1-key-col-n-key-fld-n>
END-EXEC
IF SQLCODE = 0
THEN
        EXEC SQL
                UPDATE <table-2>
                SET     <table-2-col-1> = <table-2-col-1-fld-1>,
                        ...
                        <table-2-col-n> = <table-2-col-n-fld-n>
                WHERE<table-2-key-col-1> = <table-2-key-col-1-key-fld-1>,
                        ...
                        <table-2-key-col-n> = <table-2-key-col-n-key-fld-n>
        END-EXEC
END-IF
...
IF SQLCODE = 0
THEN
        EXEC SQL
                UPDATE <table-n>
                SET     <table-n-col-1> = <table-n-col-1-fld-1>,
                        ...
                        <table-n-col-n> = <table-n-col-n-fld-n>
                WHERE<table-n-key-col-1> = <table-n-key-col-1-key-fld-1>,
                        ...
                        <table-n-key-col-n> = <table-n-key-col-n-key-fld-n>
        END-EXEC
END-IF
IF SQLCODE = 0
THEN
        EXEC SQL
                OPEN CURSOR dependant-updatable-cursor-1
        END-EXEC
        MOVE 1 TO <db2-count>
        PERFORM UPDATE-PARA-1 UNTIL <db2-count> > <db2-array-1-size> AND SQLCODE = 0
        EXEC SQL
                CLOSE CURSOR dependant-updatable-cursor-1
        END-EXEC
        ...
        EXEC SQL
                OPEN CURSOR dependant-updatable-cursor-n
        END-EXEC
        MOVE 1 TO <db2-count>.
        PERFORM UPDATE-PARA-n UNTIL <db2-count> > <db2-array-n-size> AND SQLCODE = 0
        EXEC SQL
                CLOSE CURSOR dependant-updatable-cursor-n
        END-EXEC
```

```
END-IF
IF SQLCODE = 0
THEN
        <not-invalid-key-statements> | NEXT SENTENCE
ELSE
        <invalid-key-statements> | DISPLAY “Error In Update “
END-IF
```

5.1.7 Delete

```
EXEC SQL
        DELETE <table-1>
        WHERE<table-1-key-col-1> = <table-1-key-col-1-key-fld-1>,
                ...
                <table-1-key-col-n> = <table-1-key-col-n-key-fld-n>
END-EXEC.
IF SQLCODE = 0
THEN
        EXEC SQL
                DELETE <table-2>
                WHERE<table-2-key-col-1> = <table-2-key-col-1-key-fld-1>,
                        ...
                        <table-2-key-col-n> = <table-2-key-col-n-key-fld-n>
        END-EXEC.
END-IF
...
IF SQLCODE = 0
THEN
        EXEC SQL
                DELETE <table-n>
                WHERE<table-n-key-col-1> = <table-n-key-col-1-key-fld-1>,
                        ...
                        <table-n-key-col-n> = <table-n-key-col-n-key-fld-n>
        END-EXEC.
END-IF
IF SQLCODE = 0
THEN
        <not-invalid-key-statements> | NEXT SENTENCE
ELSE
        <invalid-key-statements> | DISPLAY “Error In Delete“
END-IF
```

5.1.8 OPEN-CURSOR

If file organization is sequential then

```
EXEC SQL
        DECLARE <file1>-OPEN-CURSOR CURSOR FOR
        SELECT <table-1-col-1> ... <table-1-col-n>,
                ...,
                <table-n-col-1>...<table-n-col-n>
        FROM <table-1> ...<table-n>
END-EXEC
```

If file organization is indexed or relative then

```
EXEC SQL
        DECLARE <file1>-OPEN-CURSOR CURSOR FOR
        SELECT <table-1-col-1> ... <table-1-col-n>,
                ...,
                <table-n-col-1>...<table-n-col-n>
        FROM <table-1> ...<table-n>
        ORDER BY <table-1-key-col-1> ... <table-1-key-col-n>
END-EXEC
```

5.1.9 START-CURSOR

```
EXEC SQL
        DECLARE <file1>-START-RK-CURSOR CURSOR FOR
        SELECT <table-1-col-1> ... <table-1-col-n>,
                ...,
                <table-n-col-1>...<table-n-col-n>
        FROM <table-1> ...<table-n>
        WHERE <table-1-key-col-1>  >= <rec-key-fld-1>,
                <table-1-key-col-n>  >=  <rec-key-fld-n>,
                <table-2-key-col-n>  =  <table-1-key-col-1>
                <table-2-key-col-n>  =  <table-1-key-col-n>
                ... ,
                <table-n-key-col-n>  =  <table-1-key-col-1>
                <table-n-key-col-n>  =  <table-1-key-col-n>
        ORDER BY <table-1-key-col-1> ... <table-1-key-col-n>
END-EXEC
```

5.1.10 READ-CURSOR

```
EXEC SQL
        DECLARE <file1>-READ-RK-CURSOR CURSOR FOR
        SELECT <table-1-col-1> ... <table-n-col-n>,
                ...,
                <table-n-col-1>...<table-n-col-n>
        FROM <table-1> ...<table-n>
        WHERE <table-1-key-col-1>  >= <rec-key-fld-1>,
                <table-1-key-col-n>  >=  <rec-key-fld-n>,
                <table-2-key-col-n>  =  <table-1-key-col-1>
                <table-2-key-col-n>  =  <table-1-key-col-n>
                ... ,
                <table-n-key-col-n>  =  <table-1-key-col-1>
                <table-n-key-col-n>  =  <table-1-key-col-n>
        ORDER BY <table-1-key-col-1> ... <table-1-key-col-n>
END-EXEC
```

5.1.11 Dependant-Cursor-1

```
EXEC SQL
        DECLARE <dependant-cursor-1> CURSOR FOR
        SELECT <array-1-table-1-col-1> ... <array-1-table-n-col-n>,
                ...,
                <array-1-table-n-col-1>...<array-1-table-n-col-n>
        FROM <array-1-table-1> ...<array-1-table-n>
END-EXEC
```

5.1.12 Dependant-Cursor-n

```
EXEC SQL
        DECLARE <dependant-cursor-n> CURSOR FOR
        SELECT <array-n-table-1-col-1> ... <array-n-table-n-col-n>,
                ...,
                <array-n-table-n-col-1>...<array-n-table-n-col-n>
        FROM <array-n-table-1> ...<array-n-table-n>
END-EXEC
```

5.1.13 Dependant-Updatable-Cursor-1

```
EXEC SQL
        DECLARE <dependant-updatable-cursor-1> CURSOR FOR
        SELECT <array-1-table-1-col-1> ... <array-1-table-1-col-n>
        FROM <array-1-table-1>  FOR UPDATE OF <columns>
END-EXEC
```

5.1.14 Dependant-Updatable-Cursor-n

```
EXEC SQL
        DECLARE <dependant-updatable-cursor-n> CURSOR FOR
        SELECT <array-n-table-1-col-1> ... <array-n-table-1-col-n>
        FROM <array-n-table-1> FOR UPDATE OF <columns>
END-EXEC
```

5.1.15 FETCH-PARA-1

EXEC-SQL
FETCH dependant-cursor-1
INTO <*db2-array-1-table-1-col-1-fld-1*>, .. <*db2-array-1-table-1-col-n-fld-n*>
...
<*db2-array-1-table-n-col-1-fld-1*>, .. <*db2-array-1-table-n-col-n-fld-n*>
END-EXEC
MOVE <*db2-array-1*> TO <*array-1*>(<*db2-count*>)
COMPUTE <*db2-count*> = <*db2-count*> + 1

5.1.16 FETCH-PARA-n

EXEC-SQL
FETCH dependant-cursor-n
INTO <*db2-array-1-table-1-col-1-fld-1*>, .. <*db2-array-1-table-1-col-n-fld-n*>
...
<*db2-array-1-table-n-col-1-fld-1*>, .. <*db2-array-1-table-n-col-n-fld-n*>
END-EXEC.
MOVE <*db2-array-1*> TO <*array-n*>(<*db2-count*>)
COMPUTE <*db2-count*> = <*db2-count*> + 1

5.1.17 INSERT-PARA-1

```
MOVE <ARRAY-1>(<db2-count>) TO <DB2-ARRAY-1>.
EXEC SQL
        INSERT INTO <array-1-table-1>(<array-1-table-1-col-1>... <array-1-table-1-col-n>)
        VALUES (<db2-array-1-table-1-col-1-fld-1>, .. <db2-array-1-table-1-col-n-fld-n>)
END-EXEC
IF SQLCODE = 0
THEN
        EXEC SQL
                INSERT INTO <array-1-table-2>(<array-1-table-2-col-1> ... <array-1-table-
2-col-n>)
                VALUES (<db2-array-1-table-2-col-1-fld-1>, .. <db2-array-1-table-2-col-n-
fld-n>)
        END-EXEC
END-IF
...
IF SQLCODE = 0
THEN
        EXEC SQL
                INSERT INTO <array-1-table-n>(<array-1-table-n-col-1> ... <array-1-table-
n-col-n>)
                VALUES (<db2-array-1-table-n-col-1-fld-1>, .. <db2-array-1-table-n-col-n-
fld-n>)
        END-EXEC
END-IF
COMPUTE <db2-count> = <db2-count> + 1
```

5.1.18 INSERT-PARA-n.

```
MOVE <array-n>(<db2-count>) TO <db2-array-n>.
EXEC SQL
        INSERT INTO <array-n-table-1>(<array-n-table-1-col-1>... <array-n-table-1-col-n>)
        VALUES (<db2-array-1-table-1-col-1-fld-1>, .. <db2-array-1-table-1-col-n-fld-n>)
END-EXEC.
IF SQLCODE = 0
THEN
        EXEC SQL
                INSERT INTO <array-n-table-2>(<array-n-table-2-col-1> ... <array-n-table-
2-col-n>)
                VALUES (<db2-array-1-table-2-col-1-fld-1>, .. <db2-array-1-table-2-col-n-
fld-n>)
        END-EXEC
END-IF
...
IF SQLCODE = 0
THEN
        EXEC SQL
                INSERT INTO <array-n-table-n>(<array-n-table-n-col-1> ... <array-n-table-
n-col-n>)
                VALUES (<db2-array-1-table-n-col-1-fld-1>, .. <db2-array-1-table-n-col-n-
fld-n>)
        END-EXEC
END-IF
COMPUTE <db2-count> = <db2-count> + 1
```

5.1.19 UPDATE-PARA-1

```
EXEC-SQL
        FETCH
        INTO    <db2-array-1-table-1-col-1-fld-1>, .. <db2-array-1-table-1-col-n-fld-n>
                ...
                <db2-array-1-table-n-col-1-fld-1>, .. <db2-array-1-table-n-col-n-fld-n>
END-EXEC
MOVE <array-1>(<db2-count>) TO <db2-array-1>
EXEC SQL
        UPDATE <array-1-table-1>
        SET     <array-1-table-1-col-1> = <db2-array-1-table-1-col-1-fld-1>,
                ...
                <array-1-table-1-col-n> = <db2-array-1-table-1-col-n-fld-n>
WHERE CURRENT OF
END-EXEC
COMPUTE <db2-count> = <db2-count> + 1
```

5.1.20 UPDATE-PARA-n

```
EXEC-SQL
        FETCH
        INTO    <db2-array-1-table-1-col-1-fld-1>, .. <db2-array-1-table-1-col-n-fld-n>
                ...
                <db2-array-1-table-n-col-1-fld-1>, .. <db2-array-1-table-n-col-n-fld-n>
END-EXEC
MOVE <array-1>(<db2-count>) TO <db2-array-1>
EXEC SQL
        UPDATE <array-1-table-1>
        SET     <array-1-table-1-col-1> = <db2-array-1-table-1-col-1-fld-1>,
                ...
                <array-1-table-1-col-n> = <db2-array-1-table-1-col-n-fld-n>
WHERE CURRENT OF
END-EXEC
COMPUTE <db2-count> = <db2-count> + 1
```