UNIT 13 QUERY LANGUAGE

Objectives

After going through this unit, you should be able to :

- appreciate the nature of Query languages and their ease of use
- use the correct syntax for query processing, involving variants of SELECT command
- use feafures of embedded SQL in a host programming language.

Structure

13.1 Introduction
13.2 Query processing
13.3 Running queries on multiple tables
13.4 Managing data with SQL
13.5 Embedded SQL
13.6 Summary
13.7 Self assessment exercises
13.8 Further Readings

13.1 Introduction

SQL, previously called SEQUEL, is one of the most important relational data manipulation languages developed based on the principles of relational calculus. A prototype implementation of the original version of SQL was developed by IBM at their San Jose, California Research Laboratory as a data definition language and a data manipulation language. Its statements can be issued interactively with a terminal or embedded in a host programming language. The interactive commands are processed by the ISQL (interactive SQL) processor.

In the following sections, various SQL expressions for database retrievals and updates will be illustrated. All the examples given below will be based on the relations in referred to in the earlier unit.

An SQL query expression consists of one or more **retrieval blocks** called SELECT - FROM - WHERE blocks. Each block has the following structure :

SELECT fields

FROM relations

(WHERE logical Conditions)

The SELECT clause specifies one or more target columns to be retrieved, and the FROM clause specifies source relations (either base table or view) from which desired columns may be obtained. The optional WHERE clause specifies conditions for selecting rows. For example, the following SQL statement will retrieve INV-NO and QTY for transactions made by customer CI.

SELECT INV_NO, QTY

```
FROM        TRANSACTION

WHERE       CUST-ID = 'C1'
```

13.2 QUERY PROCESSING

The simplest form of SQL expressions involve operations on one relation only.

1) Selecting columns

Get INV-NAME and UNIT - PRICE of all inventory items.

```
SELECT          INV NAME,      UNIT PRICE

FROM            INVENTORY
```

Since the WHERE clause is omitted, all rows are selected from the INVENTORY relation. The response to the query is :

```
INV-NO      UNIT-PRICE

CHAIR       75.00

TABLE       259.15

DESK        399.00
```

2) Selecting rows

Get all INVENTORY records whose UNIT-PRICE<100.00

```
SELECT

FROM        INVENTORY

WHERE       UNIT-PRICE<100.00
```

The response is :

```
INV-NO   INV-NAME    UNIT-PRICE
  11        CHAIR      75.00
```

The asterisk represents a selection of all columns from the specified relation.

3) WHERE clause involving more than one condition

Get TRANS-NO and INV-NO for all transactions made by customer C1 where the quantity of each transaction is greater than 2.

```
SELECT          TRANS-NO, INV-NO
```

```
FROM            TRANSACTION

WHERE           CUST-ID = 'C1' and QTY>2
```

The answer, as you can verify for yourself, is :

```
TRANS-NO    INV-NO
T1          11
```

This last example specifies two conditions in the WHERE clause for a horizontal subset of the TRANSACTION relation.

4) Eliminating duplicate responses

Get INV-NO of inventory items sold after July 1.

```
SELECT          DISTINCT   INV-NO

FROM            TRANSACTION

WHERE           DATE-OF-TRANS 01/07
```

The response is :

```
INV-NO
13
12
```

Referring to the sample data in section 12.2, there are two TRANSACTION records in which the inventory item, 13, was sold after July 1. However, duplicated responses will be suppressed due to the specification of the option DISTINCT in the SELECT clause. In SYSTEM R, the key word UNIQUE is used to replace the key word DISTINCT for suppressing duplicate answers.

In general, duplicate responses are not eliminated automatically by the system because of the cost of that operation. Most systems leave it up to the users to decide whether a suppression of duplicate responses is required.

5) Ordering retrieved records

The following SQL expression will answer the same query given in example (4) except that retrieved records will be ordered in the ascending order of INV-NO, and duplicate responses will not be suppressed.

```
SELECT          INV-NO

FROM            TRANSACTION

WHERE           DATE-OF-TRANS 01/07

ORDER           BY INV-NO  ASC
```

the answer is as follows :

INV-NO
12
13
13

13.3 RUNNING QUERIES ON MULTIPLE TABLES.

A query may involve retrieval of data from more than one table. This may be accomplished by joining tables based on a commond field. Here are some examples of retrieving data from more than one relation.

1) Retrieving data from two relations

Get names of customers who bought inventory item 13.

```
SELECT      CUST-NAME
FROM        CUSTOMER, TRANSACTION
WHERE       CUSTOMER-CUST-ID = TRANSACTION. CUST-ID
            AND INV-NO = '13'
```

Response :

CUST-NAME
DAVE
JANE

Conceptually, the above SQL retrieval may be accomplised with the following equivalent relational algebraic operations :

STEP 1 Join the two relations by matching equal values in the CUST-ID field of each relation

STEP 2 Select from the joined relation those rows containing 13 in INV-NO.

STEP 3 Select the desired columns, CUST-NAME, from the joined relation,

2) Retrieving data by joining a relation with itself

Find pairs of customers who bought the same inventory items.

```
SELECT      A. CUST.ID, B.CUST-ID
FROM        TRANSACTION A, TRANSACTION B
WHERE       A-INV-NO = B.INV-NO
            AND A.TRANS-NO<B,TRANS-NO
```

Response :

A. CUSTID B. CUST-ID
C1 C3

To find the answer, two identical TRANSACTION relation as idenified by A and B are joined. In other words, the TRANSACTION relation joins with itself. The WHERE clause specifies that records in the two identical tables are joined over INV-NO except that the same record will not join with itself.

13.4 MANAGING DATA WITH SQL

The SQL includes UPDATE, INSERT and DELETE statements to alter the database content. The following examples are based on the relations in used for illustration in the preceding sections.

1) INSERT

The following INSERT command is used to insert a record, T5 13 C1.1, into the TRANSACTION relation.

INSERT INTO TRANSACTION (TRANS-NO, INV-NO, CUST-ID, QTY) VALUES ('T5,13,C1,1)

When the new record is inserted into the relation, the system will automatically establish necessary pointers or indexes to accommodate the new record. Since the DATE-OF-TRANS fields is omitted in the INSERT statement, its content will be initialized to null.

An insert statement may be issued without specifying any field names. In this case, the list of data fields originally defined in the CREATE TABLE statement in the left-to-right order will be used.

The second type of INSERT statement involves a retrieval of a number of records from the database and a subsequent insertion of retrieved records into an employ relation. Suppose that an empty table, TEMP (TRANS-NO, INV-NO), has been defined with a CREATE TABLE statement. The following INSERT statement will first retrieve the transaction number and the inventory number of all the transactions made by customer C1 and then insert the retrieved data into the empty table TEMP.

```
INSERT INTO TEMP
SELECT TRANS - NO, INV - NO
FROM TRANSACTION
WHERE CUST - NO = 'C1'
```

2) Delete

A record in a relation may be deleted with a DELETE command by specifying its unique primary key. For example, the following statement deletes a record with the TRANS-NO T1 from the TRANSACTION relation.

```
DELETE TRANSACTION
WHERE TRANS-NO - 'T1'
```

If a secondary key value is specified in the WHERE Clause, one or more records that satisfy the condtion will be deleted from the specified relation. For example :

```
DELETE TRANSACTION
```

```
WHERE  CUST-ID = 'C1'
```

The DELETE statement will delete all those transactions made by customer C1.

3) UPDATE

The SQL UPDATE command is used to change the content of one or more existing records in the database. For exapmle, the unit price of an inventory item 13 can be increased by 2.00 as follows :

```
UPDATE   INVENTORY
SET  UNIT-PRICE  = UNIT-PRICE  + 2

WHERE   INV-NO  = '13'
```

After the expression is executed, the unit price of 13 in the INVENTORY relation is changed from 399.00 to 401.00. However, if the field to be updated is a primary key or a foreign key, updates of the field may cause data inconsistency if the corresponding field in its parent or child record is not simultaneously updated. Current relational systems provide no automatic procedures to enforce referential integrity. Thus, when a connection-field value is a relation is to be updated, the user will be responsible for updating the corresponding field in all its parent or child records. For example, if the inventory number, 13, in the INVENTORY relation is to be changed to 15, then records containing 13 in the TRANSACTION relation must all be updated to 15 :

```
UPDATE          INVENTORY
SET             INV-NO = '15'
WHERE           INV-NO = '13'
UPDATE          TRANSACTION
SET             INV-NO = '15'
WHERE           INV-NO = '13'
```

13.5 EMBEDDED SQL

In SYSTEM R, SQL/DS and DB2, the SQL statement can be issued interactively or embedded in a host programming language such as PL/1 or COBOL. This characteristic is generally not available in network or hierarchical database management systems.

The SQL statements embedded in an application program are first preprocessed by the DBMS, so that the SQL statements will be translated into CALL statements to invoke appropriate routines in the DBMS. An application embedded with SQL statements is first passed through a preprocessor and then through a normal compiler for converting host language statements into machine code.

One of the problems with embedding SQL retrieval statements in an application program is that the host procedural language is equipped to handle I/O only one record at a time. However, an SQL SELECT statement may retrieve multiple records from the database. To resolve such conflicting orientation in the code of I/O operations between SQL and the conventional pogramming language, some relational systems (e.g. SYSTEM R, SQL/DS and DB2) proved a cursor mechanism to fetch records one by one from a table retrieved by an SQL command.

13.6 SUMMARY

Structured Query Language (SQL) is a DML derived fom relational calculus. However, an SQL statement can be translated into equivalent relational algebric steps.

The basic consrtuct of an SQL retrieval statement involves a SELECT _ FROM-WHERE block with which object fields and a predicate for selection criteria are specified. An SQL statement may select one or more records at a time. Nested SELECT-FROM-WHERE blocks may be coded within an SQL expression.

Embedding SQL statements in a host programming language for batch processing is described. The I/O operation of a conventional host programming language is record oriented. while the execution of an SQL statement may retrieve an entire table. To accommodate such differences in the I/O processing mode, SYSTEM R,DB2 and SQL / DS all provide a cursor mechanism for application programs to process the table retrieved by an SQL statement one record at a time.

13.7 SELF - ASSESSMENT EXERCISES

1. In the specific context of your own organisation, name one programming language you would need in which SQL statement can be embedded. Similarly find one which is not important for being embedded.

2. Taking the example given in the inventory table, how would you use the UPDATE command to increase the price of each item by 20%

3. Why does a system, of its own accessed not automatically eliminatate duplicate responses when eliminate duplicate responses when executing a SELECT command ?

13.8 FURTHER READINGS

1. Atre S. *Database Structural Techniques for Design, Performance & Management,* John Wiley & Sons, 1980.

2. Date C.J. *A Guide to DB2* Addison-Wesley. 1984

3. Date C.J *An Introduction to Database Systems,* Addison-Wesley, 1981

4. Hawry Stkiewycz I.T. *Database Analysis and Design,* SRA. 1984

5. Kroenke D.M. *Database Processing : Fundamentals, Design, Implementation* 2nd Edition, SRA, 1983.