# Unit 2 RDBMS and SQL

**Structure:**



## 2.1 Introduction

In previous unit, you studied the the advantages and disadvantages of those database systems.

The interaction level of database depends on its usage. If the user uses the database at higher level than the interaction level will also be high. Hence,

each database systems should give several methods, languages and group of software. So that users can submit a request, process the request; and can get the output of the request. This unit introduces some of the database query languages and tools.

Now as we are clear about various types of DBMS, let us start this unit, where you will learn about query languages and you will also study SQL features and queries.

**Objectives:**

After studying this unit, you should be able to:

- create relational database objects using SQL
- formulate tables and data residing in them
- create and manipulate views
- describe transaction processing
- discuss the concept of embedded SQL and dynamic SQL

## 2.2 Relational Query Languages

Modern RDBMSs supports several query languages for user interaction. There are two most common query languages available are SQL (Structured Query Language) and QBE (Query by Example) with RDBMS.

- ***Others are Information System Base Language*** (ISBL) from the Peterlee Relational Test Vehicle (PRTV) system and QUEL (Query Language) from INGRES (Interactive Graphics Retrieval System). ISBL (Information System Base Language) is based on relational algebra, Query Language and SQL is like tuple calculus and QBE is like a domain calculus.
- In this section, we will focus on QBE and in the forthcoming sections (section 2.3 onwards) you will study about SQL in detail.
- ***Query by Example (QBE):*** QBE was developed in mid 70s at IBM research simultaneously with the development of SQL. M.M Zloof had designed the Query by Example (QBE) which is a relational database query language. It is the first graphical query language. QBE is used for visual representation of tables where the user gives commands for defining what is to be done, instances for defining how it is done and conditions in which records should be admitted into the processing.

**Self Assessment Questions**

1. QBE stands for _____.
2. SQL is supported by RDBMS. (True/False)

## 2.3 SQL

**SQL (Structured Query Language)** is a standard relational database language used for creation, deletion and modification of database tables.

(***Note*:*** The SQL Keywords are case-insensitive (SELECT, FROM, WHERE, etc); we have used caps words where we want to put emphasis on the word. Table names, column names etc are case-insensitive in Windows OS but it is case sensitive in UNIX OS)

***Features:*** SQL has a very rich set of features which are given in Table 2.1 below:

**Table 2.1: SQL Features**

| | |
|---|---|
| The Data Manipulation Language (DML): | As, the name says this language is used for manipulating. The data is stored in database objects. DML uses SELECT, INSERT, DELETE and UPDATE command to modify the data. |
| The Data Definition Language (DDL): | This language is used to define the structure of the table. With CREATE, ALTER and DROP commands the structure of the table can be modified, it can also be deleted and created as well. |
| Specifications of Triggers and Complex Integrating Constraints: | SQL provides the features of the triggers and complex integrity constraints (ICs) to be applied on queries. |
| Triggers: | Triggers are the actions that are run by DBMS whenever some event related to the database occurs. |
| Run-time (Dynamic) and Embedded SQL: | With run-time feature of SQL, users can execute the queries at run-time. With embedded SQL, users can retrieve the SQL statements that are the part of some other host language (such as C or Cobol). |
| Execution of the Client-Server Application and Accessing Remote | This feature allows a client program to establish a connection with the server database. This feature |

| | |
|---|---|
| Database: | also allows the user to access remote database. |
| Managing, Transaction: | SQL command specifies the actions to be taken in order to control the execution of the transaction. |
| Security: | It controls the access to the tables and views thereby protecting the database. |
| Advanced Features provided by the SQL: | Many features such as recursive and decision-support queries, object-oriented features etc are provided by the SQL |

**Self Assessment Questions**

3. SELECT, INSERT, DELETE and UPDATE commands are used by _____ to modify the data.
4. SQL commands defines the actions to be taken to control _____ .

## 2.4 Integrity Constraints

DBMS maintains the data integrity to avoid the wrong information to enter in database.

The condition of integrity constraints are defined on database schema. An integrity constraint limits the data that could be stored in database instance. When database instance fulfils all the integrity constraints -defined on the database schema, it is then known as legal instance. A DBMS implements integrity constraints; therefore it permits only legal instances to be stored in the database.

The major relational constraints are *Domain constraints*, *Key constraints* and *constraints on null*, *Entity integrity* and *Referential integrity and foreign keys.*

### 2.4.1 Entity integrity

When all rows in the column have unique identifier it means each row is different from other it is known as entity integrity. Entity integrity is placing a primary key (PK) constraint on particular column. This ensures all values inserted into the column (s) should be unique. In PK constraints you cannot enter duplicate values and null values in column (s) because it results to failure.

The primary key of a relational table uniquely identifies each record in the table. It can either be a normal attribute that is guaranteed to be unique or it

can be generated by the DBMS (such as a globally unique identifier in Microsoft SQL Server). Primary keys may consist of a single attribute or multiple attributes in combination. Intelligent key is the utilisation of genuine data as a PK. Only one PK is assigned to a table. A composite PK does not contain only one column. We can utilise the composite PK when not even one column has unique composite key.

Hence we can say that a table can contain only one PK but a PK can contain more than one column. If we have to apply uniqueness on more than one column, we need to utilize a PK constraint on single column and UNIQUE constraint or IDENTITY property on other columns that does not contains duplicate values.

### 2.4.2 Domain integrity

In database language a domain is group of allowed values for a column (domain cannot be confuse between different types of domain for example Internet domain or Windows NT Domain.

Domain integrity is also called '*attribute*' integrity for example allowed size values, right data type, null status etc. Implementation of data integrity can be with DEFAULT constraint, FOREIGN KEY, CHECK constraint and data types. Data types restrict the fields in different ways. A default can be defined as a value to be inserted into column; a rule is defined as acceptable values to be inserted into column. Rules and defaults are same as constraints but not similar to ANSI standard; their continued utilisation is not promoted.

### 2.4.3 Referential integrity

Referential integrity is formed with the combination of Primary Key (PK) and Foreign Key (FK).

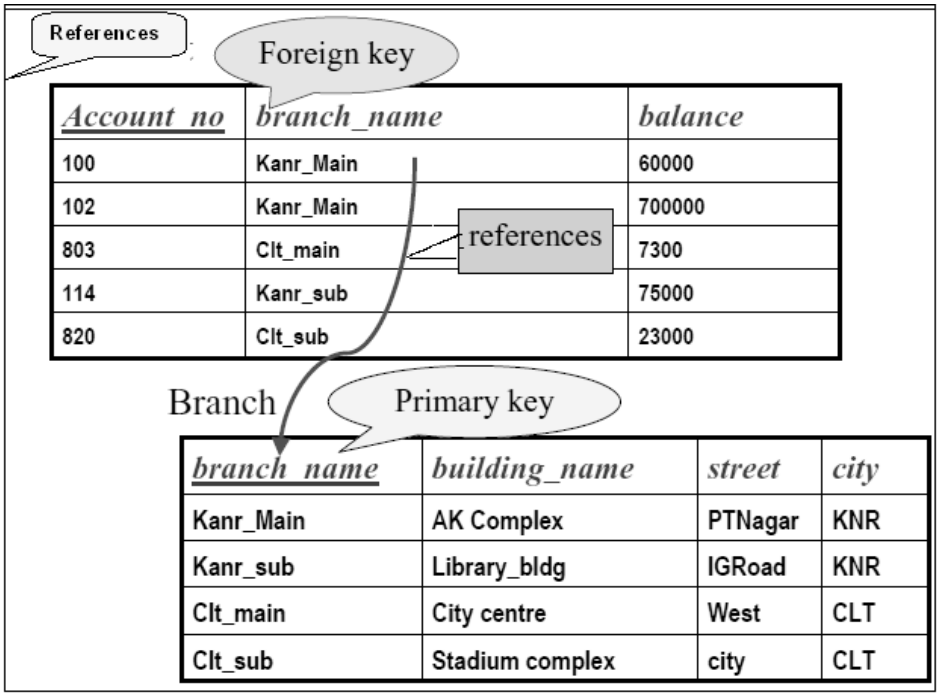
***Primary key:*** As explained above it is a key that uniquely recognises a record in a field(s) of a table. Hence a particular record can be tracked without confusion.

***Foreign Key:*** Foreign key is a column or even a group of columns in a table (as also called 'child table') that accept its values from the primary key (PK) from another table (also called 'parent table'). To preserve the referential integrity, the foreign key in the 'child' table can only take values that are in the primary key of 'parent' table. The main aim of referential integrity is to

avoid 'orphans'. These orphans are records in ‘child table’ that cannot be linked to a record in the ‘parent table’.

Implementing **referential integrity** means when the records go through the operations like insertion, deletion and updation on that time the relationship should be maintained between the tables. PK-FK combination also has the referential integrity. An example of primary key and foreign key is represented in Figure 2.1.

In the 1st table, first column (Account Number) is the PK and in the same table branch name is the FK. To connect 1st and 2nd table the FK has become PK.




| *Account_no* | *branch_name* | *balance* |
|---|---|---|
| 100 | Kanr_Main | 60000 |
| 102 | Kanr_Main | 700000 |
| 803 | Clt_main | 7300 |
| 114 | Kanr_sub | 75000 |
| 820 | Clt_sub | 23000 |

| *branch_name* | *building_name* | *street* | *city* |
|---|---|---|---|
| Kanr_Main | AK Complex | PTNagar | KNR |
| Kanr_sub | Library_bldg | IGRoad | KNR |
| Clt_main | City centre | West | CLT |
| Clt_sub | Stadium complex | city | CLT |

**Figure 2.1: Instance of PK and FK**

**Self Assessment Questions**

5. _____ is formed with the combination of PK and FK.
6. Domain integrity is also called as ‘_____' integrity.

## 2.5 Data Definition Statements

Data Definition Language (DDL) permits user for the creation or modification of database objects. Specifically, they perform the task of creating objects, altering or modifying objects, dropping or deleting objects, etc.

### 2.5.1 Creating relations in SQL

We define an SQL relation using the CREATE TABLE command to create a TABLE Structure. CREATE TABLE syntax is given below.

```
CREATE TABLE <tablename>
    (
    Column1 data type (size) [null/not null]
    Column2 data type (size), …….
    )
```

For example, to create the table EMP enter the following query.

```
CREATE TABLE EMP
        (
        EMPNO NUMBER (4) NOT NULL
        ENAME VARCHAR 2 (10),
        JOB VARCHAR2 (9),
        DOJ DATE,
        SAL NUMBER (7,2),
        COMM NUMBER (7,2),
        DEPTNO NUMBER (2 NOT NULL)
        )
```

In the above program we have created a table name “EMP”. In which there are 7 columns EMPNO (Employee Number), ENAME (Employee Name), JOB, DOJ (Date of Joining), SAL (Salary), COMM (Communication Number), DEPTNO (Department Number).

### 2.5.2 Adding and deleting tuples

***Adding a tuple/record/row:*** INSERT command is used to insert record in table. The syntax is:

INSERT INTO <tablename>

VALUES (value1, value 2, ……);

For example

INSERT INTO EMP

VALUES (‘101’, ‘Nandi’, ‘President’, ‘17-NOV-88’, 5000, null, ‘10’);

The above example shows the insertion of record into the EMP table.

To insert values into only EMPNO, DEPTNO and ENAME fields, enter the following query.

INSERT INTO EMP (EMPNO, DEPTNO, ENAME)

VALUES (‘101’, ‘29’, ‘Sujit’);

***Deleting tuple:*** DELETE command is used to delete row from the table. The syntax is:

DELETE FROM <tablename>

WHERE<condition>

For example,

DELETE FROM EMP

WHERE SAL > 1000;

The above example shows the deletion of all the employees whose salary is more than 1000. If we delete the WHERE clause then all rows of the table will be deleted but a part of the row cannot be deleted.

### 2.5.3 Destroying and altering relations

DROP TABLE command is used to deletes all the information of a dropped relation from database. Syntax is *DROP TABLE<Table Name>.*

Example: **DROP TABLE** details

There are two types of DROP command: CASCADE and RESTRICT.

CASCADE command deletes the complete database schema which contains tables, domains and other elements.

RESTRICT command deletes the database schema if it does not contain any element or else the command will be terminated.

***Alter table command:*** ALTER TABLE command adds attributes to an existing relation. The Null values are assigned to all the tuples as a new attribute. The syntax is

ALTER TABLE d ADD I, D

Where *d* is existing relation, *I* is added attribute, and *D* is the domain of the added attribute. .

ALTER TABLE d DROP I

This statement can drop attributes from a relation. Where d is existing relation and I is the attribute of the relation.

**Example:** ALTER TABLE details ADD Parents_Name VARCHAR (20);

The above example will add an attribute: Parents_Name to the table details.

**Self Assessment Questions**

7. There are two types of DROP commands: CASCADE and RISTRICT (True/False)
8. _____ command helps for the creation of SQL relations.

## 2.6 Data Manipulation Language

Data Manipulation Language (DML) contains commands which manipulate data in existent database schema objects. Current transactions are committed by these statements. You can find these commands in Table 2.2.

**Table 2.2: Data Manipulation Language Commands**

| Command | Purpose |
|---|---|
| DELETE | To remove rows from a table |
| EXPLAIN PLAN | To return the execution plan for a SQL statement |
| INSERT | To add new rows to a table |
| LOCK TABLE | To lock a table or view, limiting access to it by other users |
| SELECT | To select data in rows and columns from one or more tables |
| UPDATE | To change data in a table |

The SELECT statement is used for retrieving information from the database.

### 2.6.1 SELECT statement

Apart from information retrieval, this statement also gives query capability. It means that when select command runs the information present in table will be displayed on the screen.

***Syntax:*** The three common elements of SELECT command is SELECT, FROM and WHERE. These elements can retrieve information from more than one table. The syntax is:

SELECT <column_list>

FROM <table_list>

WHERE <search_criteria>

*Where*

- <column_list> defines the list attributes whose value is to be extracted
- <table list> defines the list of relation names
- <Condition> defines the conditional expression that recognises the tuple.

In SQL, basic logical comparison operators are used on the WHERE clause. Comparison operators and their meanings are given in Table 2.3:

**Table 2.3: Logical Comparison Operators and their Meaning**

| Operator | Meaning |
|---|---|
| = | equal to |
| > | greater than |
| >= | greater than equal to |
| < | less than |
| <= | less than equal to |
| <> | not equal to |
| ! = | not equal to |
| ! > | not greater than |
| ! < | not less than |
| ( ) | order of precedence |

***Let us first begin with a very basic SQL query.***

***Example:*** Assume a table, whose name is EMPLOYEE

| EMPNO | ENAME | DESIGNATION | DEPTNO | PAY | INCENTIVES |
|---|---|---|---|---|---|
| 1821 | JOHN | PRESIDENT | 1 | 60000 | 8000 |
| 1858 | AINA | MANAGER | 3 | 30000 | 6000 |
| 1875 | KRIPSI | MANAGER | 1 | 20000 | 4000 |
| 1877 | ARICA | MANAGER | 2 | 15000 | 1000 |

SELECT EMPNO, ENAME, DEPTNO

FROM EMPLOYEE

WHERE DEPTNO =2;

This query will display 3 columns, i.e., EMPNO, ENAME, AND DEPTNO of all rows of the EMPLOYEE table, whose DEPTNO is 2.

| EMPNO | ENAME | DEPTNO |
|---|---|---|
| 1877 | ARICA | 2 |

Example:

SELECT *

FROM EMPLOYEE

WHERE DESIGNATION = ‘MANAGER’;

This query will display all 5 columns, i.e., EMPNO, ENAME, DESIGNATION, DEPTNO and PAY of all rows of the EMPLOYEE table whose DESGINATION stores MANAGER.

| EMPNO | ENAME | DESIGNATION | DEPTNO | PAY | INCENTIVES |
|---|---|---|---|---|---|
| 1858 | AINA | MANAGER | 3 | 30000 | 6000 |
| 1875 | KRIPSI | MANAGER | 1 | 20000 | 4000 |
| 1877 | ARICA | MANAGER | 2 | 15000 | 1000 |

***Note:*** An asterisk (*) is used to retrieve all columns from the table.

The Complete syntax for the SELECT statement is as follows:

SELECT [ALL/DISTINCT] [TOPn] [PERCENT] [WITH TIES]]

select - list

[INTO new_Table]

[FROM table_Sources]

[WHERE search_Condition]

[GROUP By [ALL] Group_by_expression [,.....n]

[WITH {CUBE ¦ ROLLUP}]]

[HAVING search_Condition]

[ORDER BY {column_name [ASC / DESC]} [,....n]]

[COMPUTE {{Column_Name [ASC / DESC]} [,....n]]

[COMPUTE {{AVG | COUNT | MAX | MIN | SUM} (expression)} [,....n] [By expression [,...n]]

[FOR BROWSE] [OPTION (query_hint [,...n])]

### 2.6.2 Subquery

With the help of WHERE and HAVING commands it is possible to embed a SQL statement into another. In this situation the query is known as sub query and the entire select statement is known as nested query.

The structure is:

SELECT “column_name1”

FROM “table_name1”

WHERE “column_name2” [Comparison Operator]

(SELECT “column_name3”

FROM “table_name2”

WHERE [Condition])

***Example:*** Take the table EMPLOYEE mentioned above

EMPLOYEE

| EMPNO | ENAME | DESIGNATION | DEPTNO | PAY | INCENTIVES |
|---|---|---|---|---|---|
| 1821 | JOHN | PRESIDENT | 1 | 60000 | 8000 |
| 1858 | AINA | MANAGER | 3 | 30000 | 6000 |
| 1875 | KRIPSI | MANAGER | 1 | 20000 | 4000 |
| 1877 | ARICA | MANAGER | 2 | 15000 | 1000 |

Display the employees whose DEPTNO is the same as that of employee 1821

Select ENAME, DEPTNO

FROM EMP

Where DEPTNO =

(SELECT DEPTNO

FROM EMP

WHERE EMPNO = 1821);

In the example above you can see that the inner query is executed firstly and then the result is followed by the outer query.

***Result:***

| ENAME | DEPTNO |
|---|---|
| JOHN | 1 |
| KRIPSI | 1 |

### 2.6.3 Querying multiple relations

SQL have various set operators for instance in, any, all, exists, not exists, union, minus, intersects. These operators are utilised for the processes like testing and membership of value in

- a set of values or
- the values in a set of values or
- membership of a tuple in a set of topples

***Example:*** Make the list of all employees working for a department located in NEW YORK

| EMPNO | ENAME | DESIGNATION | DEPTNO | PAY | | DEPTNO | DEPTNAME | LOCATION |
|---|---|---|---|---|---|---|---|---|
| 1821 | JOHN | President | 1 | 60000 | | 1 | Accounting | New York |
| 1858 | AINA | Manager | 3 | 30000 | | 2 | Research | Dallas |
| 1875 | KRIPSI | Manager | 1 | 20000 | | 3 | Sales | Chicago |
| 1877 | ARICA | Manager | 2 | 15000 | | 4 | Operations | Boston |

SELECT * FROM EMP WHERE DEPTNO IN

(SELECT DEPTNO FROM DEPT WHERE LOC = ‘NEW YORK’);

***Result:***

| EMPNO | ENAME | DESIGNATION | DEPTNO |
|---|---|---|---|
| 1821 | JOHN | President | 1 |
| 1875 | KRIPSI | Manager | 1 |

### 2.6.4 Functions

A Subprogram that returns a value is known as functions. SQL supports various aggregate functions shown below.

(a) ***Count:*** COUNT function contains a column name and returns the count of tupple in that column. When DISTINCT command is used then it will return only the COUNT of unique tupple or distinct values of the column. If the column name and DISTINCT command is not used then it will return the count of all tupples including duplicates also COUNT (*) displays all the tuples of the column.

Example: Write a query to List the number of employee in the company from a table employee

SELECT COUNT (*)

FROM EMPLOYEE

(b) ***SUM:*** SUM function is written with column name and gives the sum of all tupples present in that column.

(c) ***AVG:*** AVG function or Average function is written with column name and returns the AVG value of that column.

(d) ***MAX:*** MAX function or Maximum value function written with column name returns the maximum value present in that column.

(e) ***MIN:*** MIN function or Minimum value function written with column name returns the minimum value present in that column.

**Examples of Queries Based on Aggregate Functions Queries**

Find the sum of salaries of all the employees and also the minimum, maximum and average salary.

**Solution:**

SELECT SUM (E.ESAL) AS SUM_SALARY,

MAX (E.ESAL) AS MAX_SALARY,

MIN (E.ESAL) AS MIN_SALARY,

AVG ([DISTINCT] E.ESAL) AS AVERAGE_SALARY

FROM EMPLOYEE

This query calculates the total, minimum, maximum and average salaries and also renames the column names.

### 2.6.5 GROUP BY

GROUP BY clause is utilised with the group functions for retrieving the data which is grouped according to one or more columns.

Example: Calculate the total number of salary spent on each department. What would be the query?

SELECT DEPT, SUM (SALARY)

FROM EMPLOYEE

GROUP BY DEPT;

The output would be like:

| dept | salary |
|---|---|
| ---------- | ---------- |
| Electrical | 25000 |
| Electronics | 55000 |
| Aeronautics | 35000 |
| InfoTech | 30000 |

### 2.6.6 Updating the database

UPDATE command is used for updating a single value without updating all the values in tupple. Syntax is

Update table_name set attribute = newvalue where condition;

Suppose we wish to change the house name of the student ‘Simran’ stored in the relation *ST_DATA*. The following statement will serve the purpose.

UPDATE ST_DATA

SET ST_HNAME =’pranavam’

WHERE ST_NAME=’meenu’;

**Self Assessment Questions**

9. With the help of WHERE and _____ commands it is possible to embed a SQL statement into another.
10. It is not possible to query multiple relations in SQL. (True/ False)

**Activity 1**
Generally, there are numerous ways to specify the same query in SQL. In your opinion, what are the main advantages and disadvantages of this flexibility?

## 2.7 Views

A view is a subschema in which logical tables are generated from more than one base table. For example Windows; windows are similar to a created view where user can see the stored information in tables. View is stored as a query as it does not contain its own data. During the query execution contents are taken from other tables. When the table content gets modified or changed then the view will change dynamically,

The syntax to create a view is given below.

```
CREATE VIEW <view name>
        AS <query>;
```

In a single table if query does not have GROUP BY clause and DISTINCT clause then user can UPDATE and DELETE rows in a view. And if query have columns defined by expressions then user can INSERT rows.

Example: In order to create a view of EMP table named DEPT20, to show the employees in department 20 and their annual salary, use the following command.

```
CREATE VIEW DEPT20
    AS SELECT ENAME, SAL *12 FROM EMP WHERE DEPARTNO= 20;
```

Once the VIEW is created, it can be treated like any other table. Thus the following is a valid command.

```
SELECT * FROM DEPT20;
```

**Self Assessment Questions**

11. A _____ is a subschema in which logical tables are generated from more than one base table.
12. During the query execution contents are taken from other tables. (True/False)

## 2.8 Embedding SQL Statements

SQL statements can be embed into various types of programming languages such as C, Cobol, Pascal, Fortaran etc. Host language is the language in which the SQl queries are embedded. Therefore, C, FORTRAN, Pascal etc are the host languages. The SQl structure, which is embedded in the host language, is termed as embedded SQl. Therefore, the programmers can make use of the various SQL commands to access and update any data, which is stored in the database.

The use of embedded statements makes it easier to make any amendments in the database. It also enhances the programmer's capability to modify the database largely. Database system is responsible for all query execution. The database then returns the result (one tuple at a time) to the program.

Before compiling the program, the embedded SQL statement is processed by using a special pre-processor. To allow the embedded SQL program to be processed at runtime, they are replaced with the declarations and procedure calls of the host-language. After doing so the resultant program is sent for compilation. For easily recognising the embedded SQL statements to pre-processor, you may use the EXEC SQL statement. It has the following syntax:

```
EXEC SQL <embedded SQL statement > END-EXEC
```

The syntax given above is a generalised form, however the syntax may differ somewhat depending upon the host language for which it is being used.

***Declaring Variables and Exceptions:*** SQL INCLUDE can be used in the host program to determine the place for inserting the special variables (variables which are used in communication within the database and program) by the pre-processor. Host language variables can also used inside the embedded SQL statements. It is a good practice to append a

colon before the host variables to differentiate them from other variables used in SQL. Declare cursor statement is used for writing a embedded SQL query within a host program. It does not runs the query. Separate command is used to fetch the result of the embedded query.

Let us take an example of banking schema. Suppose you have a host-language variable termed as “amount”, and you want to determine the names and residing cities of all the bank customers who currently have balance more than a particular amount in any of their accounts. The query for finding this can be written as shown below:

***EXEC SQL***

*declare c cursor for*

*select customer-name, customer-city*

*from deposit, customer*

*where deposit.customer-name = customer.customer-name and*

*deposit.balance > : amount*

***END-EXEC***

The variable c that is used in the above query statement is termed as the ‘cursor’. This cursor is used for identifying a query in an open statement and also helps in query evaluation.

This cursor variable is also used in the fetch statement. It places the values of a tuple/row in the host language variable. Given below is an example of this.

```
EXEC SQL open c END-EXCE
```

When any error occurs in the execution of SQL query then the error report is stored inside a special variable. These special variables are called as SQL communication-area (SQLCA) variables. The declarations for the SQLCA variables are contained inside SQL INCLUDE statement.

**Fetch Statement**

A sequence of fetch statements are used to make tuples of the result available to the program. One host language variable is required for each attribute of the result relation. Therefore in the banking schema example we require two separate variables i.e. one for storing customer name and the other for storing customer resident city. Let us assume we take a variable en

for storing the customer name and cc for storing customer city. Then the tuple of the result relation can be obtained by using the following statement:

EXEC SQL fetch c into: en,: cc END-EXEC

After this the programmer can modify the values of the two variables en and cc by using the host language commands and features.

**Close statement**

The close statement is another embedded SQL statement, which is used for deleting the temporary relation that stores the query result. Given below is the use of close statement in our example:

```
EXEC SQL close c END-EXEC
```

**Embedded SQL statements for database modifications**

The Embedded SQL statements which are used for database modification such as update, insert, & delete return no result. Therefore, they are simple and easy to use. For example, a database-modification statement in Embedded SQL has the following syntax:

```
EXEC SQL < any valid update, insert, or delete> END-EXEC
```

A SQL database modification expression may also contain the host-language variables, that is preceded by a colon. In case of an error during statement execution, SQLCA comes into picture.

**Self Assessment Questions**

13. To recognise embedded SQL requests to the pre-processor, we use the _____ statement.
14. It is a good practice to append a colon before the host variables to differentiate them from other variables used in SQL. (True/False)

## 2.9 Transaction Processing

The logical unit of database processing is defined by mechanism provided by transaction processing. Transaction processing systems consists of immense databases and lakhs of users concurrently executing database transaction. Transaction is a logical unit of data manipulation related tasks wherein either all the component tasks must be completed or none of them is executed in order to keep the database consistent. When many

transactions proceed in the database environment it is imperative that a strict control is applied on them failing which the consistency of the database cannot be ensured.

***ACID Properties:*** Unwanted inconsistencies can easily occur in the database particularly when various transactions are executing simultaneously. The term ACID defines those properties that must be related with transactions in order that the reliability of the database is assured. The term ACID when extended can be read as the following:

**A** Atomicity
C Consistency
**I** Isolation
**D** Durability

- ***Atomicity:*** A transaction usually includes various database operations. This property of a transaction makes sure that either every operation is executed in a successful manner or none of them is executed at all.
- ***Consistency:*** This property requires that the database integrity rules must be obeyed properly.
- ***Isolation:*** In case of multi-transaction environment various transactions may be carrying out simultaneously on a single database. This property provides assurance that all transactions are executed independently.
- ***Durability:*** When a transaction is completed successfully, this property makes sure that the changes performed in the database are saved in the physical database.

**Transaction support in SQL**

SQL offers the concurrency control for the execution of a transaction via a Data Control Language which can also be called as (SQL DCL). When a transaction begins, we use the statement BEGIN TRANSACTION offered by SQL DCL whereas when a transaction ends, we use the statement END TRANSACTION.

There are two statements provided by SQL that makes the process of concurrent transaction control easy.

- ***COMMIT:*** On the execution of this statement, every modification done by the related transaction until now is made constant.
- ***ROLLBACK:*** On the execution of this statement, every change performed since the preceding COMMIT statement is rejected.

There are some conditions into which transactions may occur. These conditions are shown in Table 2.4 below:

**Table 2.4: Conditions into which Transactions may occur**

| Conditions | Features |
|---|---|
| Dirty read | This condition arises when a transaction reads data written by a concurrent uncommitted transaction. |
| Non-repeatable read | This condition is caused by a transaction which reads data again and finds that data has been modified by committed write operation of some other transaction. |
| Phantom read | This condition arises when a transaction executes a query again it had previously executed and gets rows different from what it got earlier. |

Depending upon the conditions given above some levels of transaction isolation are defined by SQL. These levels are discussed as below:

- ***Read uncommitted isolation:*** Here the transactions are permitted to perform the execution of all non-repeatable, dirty, and phantom reads.
- ***Read committed isolation:*** In this level when the execution of a transaction takes place, the data committed before the beginning of a query is obtained by a SELECT query.
- ***Repeatable read:*** This level does not allow dirty and non-repeatable reads. It provides permission for only phantom read.
- ***Serialisable isolation:*** From all the levels of isolation, this level is considered as the most rigid one. Here the transactions are forced to execute sequentially. Thus a transaction can start only after the completion of the existing transaction. As serialisation failures in this level can take place often, it must assure the withdrawal of a transaction.

**Self Assessment Questions**

15. SQL offers _____ statements that make easy the process of concurrent transaction control.
16. In transaction processing, the integrity rules of a database are maintained by _____ property.

**Activity 2**

Create a list of all Transaction Control commands in SQL and explain then with there uses.

## 2.10 Dynamic SQL

Dynamic SQL permits to create and submit SQL queries dynamically or run time. But, the embedded SQL statements should be entirely there at compile time, and are executed by the embedded SQL pre-processor.

Dynamic SQL is used for creating SQL queries as strings at run time depending on the input from the users. As well as it can be compiled immediately or have them prepared for later use.

Below example shows the use of Dynamic SQL in C program

```
char * sqlprog = “UPDATE account SET balance = balance * 1.05
    WHERE account-number=?”
    EXEC SQL PREPARE dynprog from: sqlprog;
    char account [10] = “A-101”;
    EXEC SQL EXECUTE dynprog using: account;
```

A “?” denotes a place holder for any value in a dynamic SQL program query.

PREPARE and EXECUTE are two important commands illustrated below:

```
char c_sqlstring[] = {“DELETE FROM Sailors WHERE rating>5”};
        EXEC SQL PREPARE readytogo FROM: c_sqlstring;
        EXEC SQL EXECUTE readytogo;
```

This example shows the declaration of C variable “c_sqlstring” and initialisation of values to the string representation of SQL commands in the first statement. The second statement results in this string being parsed and compiled as an SQL command, with the resulting executable bound to the SQL variable ready to go. (Since ready to go is an SQL variable, just like a cursor name, it is not prefixed by a colon.) The third statement executes the command.

**Self Assessment Questions**

17. _____ permits to create and submit SQL queries dynamically or run time
    a) Miscellaneous SQL
    b) Dynamic SQL
    c) Data Definition Language
    d) SQL Preprocessor
18. Using dynamic SQL, programs cannot create SQL queries as strings at run time. (True/ False)

## 2.11 Summary

Let us recapitulate the important concepts discussed in this unit:

- SQL and QBE are the main types of relational query languages.
- DBMS maintains the data integrity to avoid the wrong information to enter in database.

A DBMS implements integrity constraints; therefore it permits only legal instances to be stored in the database.

- A PK is known as a ' surrogate/alternate key' for those who does not contain genuine data.
- A subquery is simply a query within another Query.
- SQL supports various functions such as max, min, avg, count etc.
- Transaction Control commands manages changes made by Data Manipulation Language commands.
- Dynamic SQL permits to create and submit SQL queries dynamically or run time.

## 2.12 Glossary

- ***DDL:*** Data Definition Language
- ***DML:*** Data Manipulation Language
- ***Domain constraints:*** The set of all the values that an attribute can attain.
- ***Dynamic SQL:*** Dynamic SQL allows a query to be compiled at run-time.
- ***Embedded SQL:*** Embedded SQL allows SQL code [program] to be used in a host language i.e., general programming languages, such as C, COBOL, PASCAL, Fortran.

- ***ISB:*** Information Systems Base Language.
- ***PRTV:*** Peterlee Relational Test Vehicle
- ***QBE:*** Query-By-Example is a relational data manipulation language.
- ***QUEL:*** QUEry Language
- ***INGRES*** (INteractive Graphics and REtrieval System)
- ***View:*** A customised presentation of the data from one or more tables.

## 2.13 Terminal Questions

1. Explain SQL and its features.
2. Explain with examples different SQL commands used for creating and deleting relations.
3. What are the three basic components of select statement? Explain with an example.
4. What are the uses of Insert, Delete and Update commands?
5. What is the function of Create, Alter commands?
6. What do you understand by DDL? Make a list of commands used in DDL.
7. Write a short note on ACID properties of transaction model.
8. What is primary key and candidate key?
9. Write a short note on Dynamic SQL.

## 2.14 Answers

### Self Assessment Questions

1. Query by example
2. False
3. DML
4. Transaction Execution
5. Referential Integrity
6. Attribute
7. True
8. Create
9. HAVING
10. False
11. View
12. True
13. EXEC SQL
14. True

15. Two
16. Consistency
17. Dynamic SQL
18. False

**Terminal Questions**

1. SQL refers to Structured Query language. Refer Section 2.3 for more details.
2. Relations can be created by use of Create command. Refer Section 2.5 for more details.
3. The three basic components of select statement are SELECT, FROM and WHERE. Refer Section 2.6 for more details.
4. These commands are the DML commands. Refer Section 2.6 for more details.
5. Create and Alter commands are used to create and alter database objects. Refer Section 2.5 for more details.
6. DDL refer to data definition language. Refer Section 2.5 for more details.
7. Every transaction must follow ACID property. Refer Section 2.9 for more details.
8. Primary key is used to uniquely identify a row. Refer Section 2.4 for more details.
9. Dynamic SQL allows a query to be constructed (and executed) at run-time. Refer Section 2.10 for more details.

**References:**


- Peter Rob, Carlos Coronel, "Database Systems: Design, Implementation, and Management", (7th Ed.), Thomson Learning
- Silberschatz, Korth, Sudarshan, "Database System Concepts", (4th Ed.), McGraw-Hill
- Elmasari Navathe, "Fundamentals of Database Systems",(3rd Ed.), Pearson Education Asia


**E-references:**


- http://docs.oracle.com/cd/B19306_01/server.102/b14200/functions001.htm
- http://msdn.microsoft.com/en-us/library/windows/desktop/ms714570%28v=vs.85%29.aspx
- http://beginner-sql-tutorial.com/sql-commands.htm