SQL Course

The Select Statement

- Introduce the **SELECT** statement and describe its use.
- Describe fully the syntax of the **SELECT** statement.

Here it is shown how the SQL **SELECT** can be used to implement the relational algebra **PROJECT** operator.
The **SELECT** statement lets you find and view your data in a variety of ways.You can use it to answer questions based on your data i.e. how many, where, what kind of, even what if. Once you become comfortable with its sometimes dauntingly complex syntax, you'll be amazed at what the **SELECT** statement can do.

- All SQL retrievals are made using the **SELECT** statement.
- The **SELECT** statement requires a **FROM** clause to specify the table(s) to be searched.
- Data retrieval is relationally complete. This means you can :
  - Retrieve any atomic piece of data.
  - Retrieve all data.
  - Retrieve any subset of data
    (any column or row subset combination).
  - Retrieve any set of subsets of data.
  - Data can be retrieved in a sorted order.

The **SELECT** statement retrieves data From a database and returns it to you in the form of query results.
Later modules will look at depth at the variations and possibilities capable when using the **SELECT** statement.

The full **SELECT** statement is specified as follows :

> **SELECT** [**ALL**|**DISTINCT**] *{select_list* | **}* **FROM** *{table_name* | *view_name}* [, *{table_name* | *view_name}*] [**WHERE** *search_conditions*] [**GROUP BY** *column_name* [,*column_name*] ...] [**HAVING** *search_conditions*] [**ORDER BY** {*column_name* | *select_list_number*} [**ASC** | **DESC**] [,{*column_name* | *select_list_number*} [**ASC** | **DESC**]] ...]**;**

Although SQL is a free form language, you do have to keep the clauses in the **SELECT** statement in syntactical order (i.e. a **GROUP BY** clause must come before an **ORDER BY** clause). Otherwise you'll get syntax errors when you try to execute the query.

You may also need to qualify the names of database objects if there is any ambiguity about which object you mean. The most you would ever have to qualify a table or column name is by :

- database.
- owner.
- table_name.
- column_name.

This would result in: **database.owner.table_name.column_name**

Qualifiers are usually omitted in most books, articles, and reference manuals on SQL because the short forms make the **SELECT** statements more readable. However it's never wrong to include them.

The **SELECT** and **FROM** clauses are required. The remaining four clauses are optional. You can use them when you want to use the functionality they provide. The function of each clause is summarised below :

Select

This clause lists the data items to be retrieved by the **SELECT** statement. The items may be columns from the database, or columns to be calculated by SQL as it performs the query.

From

This clause lists the tables that contain the data to be retrieved by the query.

Where

The **WHERE** clause tells SQL to include only certain rows of data in the query results. A search condition is used to specify the desired rows.

Group By

This specifies a summary query. A summary query groups together similar rows and then produces one summary row of query results for each group.

Having

This tells SQL to include only certain groups produced by the **GROUP BY** clause. It also uses a search condition to specify the desired groups.

Order By

This clause sorts the query results based on the data in one or more columns.

The query results will always have the same row/column format as the actual tables in the database that are being queried. Some queries may return no rows. Even in this situation the returned result is still regarded as a table with zero rows. Query results may also contain **NULL** values if the column or columns they were retrieved from also contained **NULL** values. The advantage of the query results being in tabular form is that these results may in turn be queried or used in conjunction with other SQL statements.

A **SELECT** statement can include :

- Expressions of column values and constants :
  e.g. *c_name || c_address, 1.5 * p_economy.*
- Functions of column values and constants :
  e.g. *round (salary), upper (c_name).*
- Group functions of sets of rows :
  e.g. *avg(salary).*