SQL Course

Insert Statement

- To introduce the syntax used to add data to a table.
- To note any restrictions the SQL standard places on the use of the **INSERT** statement.
- To illustrate the use of the **INSERT** statement using examples.

- The **INSERT** statement adds new row(s) to a table.
- The **INSERT** statement can specify new values using the **VALUES** clause.
- If an **INSERT** statement only inserts some values in a row, then other values of that row are set to **NULL**.

To insert data into a table that you do not own you must use a qualified table name. The ANSI/ISO standard mandates unqualified column names in the column list, but many implementations allow qualified names.
The insert command has the form:

> **INSERT INTO** *tablename* [*column_list*] **VALUES** (*constant1, constant2, ...*) | **DEFAULT VALUES;**

If the column list is omitted then the **INSERT** statement will add a new row to the table, giving a value for every column in the row. In the **CREATE TABLE** statement the **DEFAULT** value is a value which will be automatically used by the database when a value is not specified by the **INSERT** statement for that particular column.
It is important that you type the data values in the same order as the column names in the original **CREATE TABLE** statement. Most systems require single or double quotes around character and date data types and commas to separate the values.

When you add data in some, but not all, of the columns in a row, you need to specify those columns. The order in which you list the column names does not matter as long as the order in which you list the data values matches it. The columns that were missing from the statement would have **NULL** values put into the columns in the row. If one of these columns had been assigned a **NOT NULL** status by the **CREATE TABLE** command then the **INSERT** statement would return an error.
When you omit the column list, the **NULL** keyword must be used in the values list to explicitly assign **NULL** values to columns. The **DEFAULT VALUES** is a special option that specifies a single row consisting entirely of the default values for each column.

The insert command also may be used in the form:

> **INSERT INTO** *tablename* [*column_list*] **SELECT** statement**;**

In this form, multiple rows can be added to a table. The data values for the rows are not explicitly specified within the statement text. Instead the source of the new rows are a database query, specified in the statement. This is known as a multi row **INSERT** statement.
The ANSI/ISO standard specifies several logical restrictions on the query that appears within the multi row **INSERT** statement:

- The query cannot contain an **ORDER BY** clause. This can be seen to be implicit as any

data inserted into a table is unordered anyway.
- The query results must contain the same number of columns as the column list in the **INSERT** statement (or the entire target table, if the column list is omitted), and the data types must be compatible, column by column.

Two restriction that have been relaxed since the SQL1 standard are that the query can now be the union of several **SELECT** statements and the **INSERT** statement now permits self-insertion. This is where the table itself is being referenced in the query.

The following example enters one new record into the *Airport* table.

INSERT INTO *airport* **VALUES** (*'MAD', 'Madrid', 'Spain', 1*)**;**

ID	Location	Country	Time Difference
IE	Dublin	Ireland	0
BA	London	England	0
QA	Sydney	Australia	9

Table after insert statement

ID	Location	Country	Time Difference
IE	Dublin	Ireland	0
BA	London	England	0
QA	Sydney	Australia	9
MAD	Madrid	Spain	1

The following example enters one new record into the *Aircraft* table. Two of the columns are not given values, so after the insertion they will contain their default values or be NULL.

INSERT INTO *aircraft (call_sign, model, no_club_seats)* **VALUES**(*'C171', 'Cesena', 10*)**;**