THE INSERT INTO STATEMENT- Review

Use the INSERT INTO statement to insert records into tables.

In example a) below, the column names for each value is specified.

In example b) below, the column names are not specified, so the values will be inserted into the columns in the order in which the columns were listed in the DDL statement that was used to create the table.

It's good practice to specify the names of the columns (as in example a), because you will always know which fields the values are being inserted into just from looking at the INSERT statement. This is important because you won't remember the field names 6 months after writing the INSERT statements, and someone may have altered the table structure. So, be explicit in order to avoid potential errors.

- Examples

a) INSERT INTO Student(sid, name, login, age, gpa) VALUES (53688, 'SMITH', 'SMITH@EE', 18, 3.2);

b) INSERT INTO project VALUES(1234, 'PERFECT PROJECT', NULL, 'JOHN');

THE UPDATE STATEMENT

ALWAYS try to use the WHERE clause when updating records in a table! Otherwise, you may accidentally change all of the records when you only intended to update one or a few records.

Notice that the examples include a calculation for the new value to which the field value should be updated

- **Syntax of UPDATE STATEMENT**

UPDATE tablename SET fieldname=VALUE

[WHERE fieldname comparisonOperator VALUE];

  - If the WHERE clause is specified, then only those rows that satisfy the condition will be updated
  - If WHERE clause is NOT specified, then ALL rows will be updated

Examples:

UPDATE student s SET s.age=s.age +1 WHERE sid=53688;

UPDATE project SET budget=1.1*budget WHERE projno>1000;

THE DELETE STATEMENT

ALWAYS try to use the WHERE clause when deleting records in a table! Otherwise, you may accidentally delete all of the records when you only intended to delete one or a few records.

- **Syntax of DELETE STATEMENT**

DELETE FROM tablename

[WHERE fieldname comparisonOperator VALUE];

  - If WHERE clause is specified, then only those rows that satisfy the condition will be deleted
  - If WHERE clause is NOT specified, then ALL rows will be deleted

- Examples:
- DELETE FROM project WHERE manager ='JOHN';
- DELETE FROM student;
- NOTE: You use the DELETE FROM statement to delete an entire row from a table. The table structure will remain intact.

Both the Update and Delete statements can use the where clause with the identical switches and modifiers as are used when using it with the SELECT statement. (See the Select Statement PDF File)

ADDITIONAL SQL FACTS (FYI)

IMPORTANT NOTES ABOUT THE SQL SELECT TABLES:

- The result of a SQL query is a relation, but…
  - Does not automatically eliminate redundant tuples
  - To remove redundant records from the result set, then use the keyword DISTINCT immediately after the keyword SELECT
  - Set operations (UNION, INTERSECTION and MINUS) do remove redundant tuples

- Attributes are stored by the sequence in which they are listed in the CREATE command
- Tuples are not ordered
- SELECT in SQL is the same as PROJECT in the relational algebra

IMPORTANT NOTE ABOUT DROPPING TABLES: ORDER OF OPERATIONS

When foreign key constraints are involved, the order in which you create the tables, insert data into the tables and drop the tables matters!!!!

Remember that when foreign keys are involved, you must create the tables that contain foreign keys AFTER the tables that they reference. You MUST also insert data into the tables in the same order in which you created the tables. You MUST, however, drop tables in the opposite order that you created the tables. In other words, you must drop the tables that have the foreign keys BEFORE you drop the tables that they are referencing. Otherwise, you will get errors because you are attempting to drop a table whose data is being referenced by other tables.

To avoid those errors, you must list the DROP TABLE statements for the tables with the foreign keys before the tables that they are referencing.

```
CREATE TABLE Major (
majorID                                 INT,
majorName                               VARCHAR(15)  NOT NULL,
CONSTRAINT major_PK PRIMARY KEY (majorID))
CREATE TABLE Student (
sid                                     INT,
firstName                               VARCHAR(15)  NOT NULL,
lastName                                VARCHAR(20)  NOT NULL,
login                                   VARCHAR(7),
dob                                     DATE,
majorID                                 INT,
CONSTRAINT student_PK PRIMARY KEY (sid),
CONSTRAINT student_FK1 FOREIGN KEY (majorID) REFERENCES Major(majorID));
```

Because of foreign keys, the above two tables MUST be created in the order listed above. They must also be populated with data in the order listed above. However, they must be dropped in the opposite order as follows:

DROP TABLE student;

DROP TABLE major;

You can imagine that it would be quite cumbersome to arrange the DROP TABLE statements in the correct order if you have a lot of tables. So, there is an alternative form of the DROP TABLE statement that will drop the tables in the correct order to avoid foreign key violations (irrespective of the order in which the drop table statements are listed).

If you add the keywords CASCADE CONSTRAINTS to the end of the DROP TABLE statements, then MySQL will drop the tables in the correct order. So, the following would work:

DROP TABLE major CASCADE CONSTRAINTS;

DROP TABLE student CASCADE CONSTRAINTS;