DATABASE MANGEMENT SYSTEMS
2015-2016 FALL SEMESTER
LABORATORY MANUAL

Experiment 1

What is SQL?

The Structured Query Language (SQL) comprises one of the fundamental building blocks of modern database architecture. SQL defines the methods used to create and manipulate relational databases on all major platforms.

SQL comes in many flavors. Oracle databases utilize their proprietary PL/SQL. Microsoft SQL Server makes use of Transact-SQL. However, all of these variations are based upon the industry standard ANSI SQL.

SQL commands can be divided into two main sublanguages. The Data Definition Language (DDL) contains the commands used to create and destroy databases and database objects. After the database structure is defined with DDL, database administrators and users can utilize the Data Manipulation Language to insert, retrieve and modify the data contained within it.

Data Definition Language (DDL)

The Data Definition Language (DDL) is used to create and destroy databases and database objects. These commands will primarily be used by database administrators during the setup and removal phases of a database project.

CREATE

Installing a database management system (DBMS) on a computer allows you to create and manage many independent databases. For example, you may want to maintain a database of customer contacts for your sales department and a personnel database for your HR department. The CREATE command can be used to establish each of these databases on your platform.

```
CREATE DATABASE employees
```

creates an empty database named "employees" on your DBMS. After creating the database, your next step is to create tables that will contain data.

```
CREATE TABLE personal_info (first_name nvarchar(20) not null, last_name nvarchar(20)
not null, employee_id int not null)
```

establishes a table titled "personal_info" in the current database. In our example, the table contains three attributes: first_name, last_name and employee_id.

USE

The USE command allows you to specify the database you wish to work with within your DBMS. For example, if we're currently working in the sales database and want to issue some commands that will affect the employees database.

```
USE employees
```

It's important to always be conscious of the database you are working in before issuing SQL commands that manipulate data.

ALTER

Once you've created a table within a database, you may wish to modify the definition of it. The ALTER command allows you to make changes to the structure of a table without deleting and recreating it.

```
ALTER TABLE personal_info
ADD salary money null
```

This example adds a new attribute to the personal_info table -- an employee's salary. The "money" argument specifies that an employee's salary will be stored using a dollars and cents format. Finally, the "null" keyword tells the database that it's OK for this field to contain no value for any given employee.

DROP

The final command of the Data Definition Language, DROP, allows us to remove entire database objects from our DBMS. For example, if we want to permanently remove the personal_info table that we created, we'd use the following command:

```
DROP TABLE personal_info
```

Similarly, the command below would be used to remove the entire employees database:

```
DROP DATABASE employees
```

<u>Exercises :</u>

1. A database named as db_Lab1 needs to store information about customers, orders and products. Create three tables for this database named as tbl_Customer, tbl_Order, and tbl_Product. tbl_Customer is identified by CustomerID, IdentityNumber, Name, and Surname. tbl_Order is identified by OrderID, OrderCount, and OrderAddress. Lastly, tbl_Product keeps ProductID, ProductName, Price, and ProductCount information. For each situation, draw an ER diagram that describes the given database (assuming no further constraints hold).
   Hint: You can use Visio for drawing ER diagram.

2. Create a database named as “db_Lab1” using SQL query language.

3. Create the tables described in question 1 using SQL query language.

   Hint: Use below figures to create each table.

Name	**Type**
CustomerID	int
IdentityNumber	varchar(11)
Name	varchar(20)

Fig. 1. tbl_Customer

Name	**Type**
OrderID	int
OrderCount	int
OrderAddress	varchar(20)
CustomerID	int
ProductID	int

Fig. 2. tbl_Order

Name	**Type**
ProductID	int
ProductName	varchar(100)
Price	decimal(10,2)
ProductCount	int

Fig. 3. tbl_Product

   Allow NULL for all columns except CustomerID, OrderID, and ProductID.

4. Add a column Surname to the tbl_Customer table with data type varchar(50).

5. Modify the column width of the OrderAddress field of tbl_Order.

6. Drop a column Surname from the tbl_Customer table.

7. Drop the tbl_Product table.