1.0 Introduction

Structured Query Language (SQL) is a database computer language designed for managing data in relational database management systems (RDBMS). Its scope includes data query and update, schema creation and modification, and data access control.

SQL was developed at IBM by Andrew Richardson, Donald C. Messerly and Raymond F. Boyce in the early 1970s. This version, initially called **SEQUEL**, was designed to manipulate and retrieve data stored in IBM's original relational database product, System R.

SQL has two major parts:

a. Data Definition Language (DDL) Used to create (define) data structures such as tables, indexes, clusters
b. Data Manipulation Language (DML) is used to store, retrieve and update data from tables.

2.0 Objectives

By the end of this unit, you should be able to:

n. Know what relational algebra is all about

3.0 What Can SQL do?

a. SQL can execute queries against a database
b. SQL can retrieve data from a database
c. SQL can insert records in a database
d. SQL can update records in a database
e. SQL can delete records from a database
f. SQL can create new databases
g. SQL can create new tables in a database
h. SQL can create stored procedures in a database
i. SQL can create views in a database
j. SQL can set permissions on tables, procedures, and views

3.1 Database Tables

A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.

Below is an example of a table called "Persons":

Table 7.1: Persons Table
Source: http://www.w3schools.com/

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The table above contains three records (one for each person) and five columns (P_Id, LastName, FirstName, Address, and City).

3.2 SQL Data Types

Each implementation of SQL uses slightly different names for the data types.

3.2.1 Numeric Data Types

- Integers: `INTEGER`, `INT` or `SMALLINT`
- Real Numbers: `FLOAT`, `REAL`, `DOUBLE`, `PRECISION`
- Formatted Numbers: `DECIMAL(i,j)`, `NUMERIC(i,j)`

3.2.2 Character Strings

- Two main types: Fixed length and variable length.
- Fixed length of *n* characters: `CHAR(n)` or `CHARACTER(n)`
- Variable length up to size *n*: VARCHAR(n)

3.2.3 Date and Time

- Note: Implementations vary widely for these data types.
- `DATE`

Has 10 positions in the format: `YYYY-MM-DD`

- `TIME`

Has 8 positions in the format: `HH:MM:SS`

- `TIME(i)`

Defines the TIME data type with an additional *i* positions for fractions of a second. For example: `HH:MM:SS:dd`

- Offset from UTZ. `+/- HH:MM`
- `TIMESTAMP`
- `INTERVAL`

Used to specify some span of time measured in days or minutes, etc.

- Other ways of expressing dates:
    - Store as characters or integers with Year, Month Day: `19972011`
    - Store as Julian date: `1997283`
- Both MS Access and Oracle store date and time information together in a `DATE` data type.

3.2.5 MySQL Data Types

MySQL is another powerful RDBMS in use today. In MySQL there are three main data types: text, number, and Date/Time types (see figure 7.3 for more detail).

Table 7.3: Microsoft Access Data Types
Source: http://www.w3schools.com/

Text types:

Data type	Description
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data

ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. **Note:** The values are sorted in the order you enter them. You enter the possible values in this format: ENUM('X','Y','Z')
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

Number types:

Data type	**Description**
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

*The integer types have an extra option called UNSIGNED. Normally, the integer goes from an negative to positive value. Adding the UNSIGNED attribute will move that range up so it starts at zero instead of a negative number.

Date types:

Data type	Description
DATE()	A date. Format: YYYY-MM-DD **Note:** The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MM:SS **Note:** The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MM:SS **Note:** The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MM:SS **Note:** The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format.

	Note: Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

*Even if DATETIME and TIMESTAMP return the same format, they work very differently. In an INSERT or UPDATE query, the TIMESTAMP automatically set itself to the current date and time. TIMESTAMP also accepts various formats, like YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD, or YYMMDD.

3.3 Data Definition Language (DDL)

The Data Definition Language (DDL) is used to create and destroy databases and database objects. Let us take a look at the structure and usage of basic DDL commands:

3.3.1 The CREATE DATABASE Statement

The CREATE DATABASE statement is used to create a database. SQL CREATE DATABASE Syntax is:

```
CREATE DATABASE database_name
```

Example: Let us create a database called "my_db". We use the following CREATE DATABASE statement:

```
CREATE DATABASE my_db
```

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

3.3.2 The CREATE TABLE Statement

The CREATE TABLE statement is used to create a table in a database. SQL CREATE TABLE Syntax is:

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
.
.
.
.
)
```

The data type specifies what type of data the column can hold. See tables 7.2 to 7.4 for a complete reference of all the data types available in MS Access, MySQL, and SQL Server.

Example: Let us create a table called "Persons" that contains five columns: P_Id, LastName, FirstName, Address, and City. We use the following CREATE TABLE statement:

```
CREATE TABLE Persons
(
P_Id int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
```

```
City varchar(255)
)
```

The P_Id column is of type int and will hold a number. The LastName, FirstName, Address, and City columns are of type varchar with a maximum length of 255 characters.

3.3.3 SQL Constraints

Constraints are used to limit the type of data that can go into a table. Constraints can be specified when a table is created (with the CREATE TABLE statement) or after the table is created (with the ALTER TABLE statement).

In this section, we will focus on the following constraints:

a. NOT NULL
b. UNIQUE
c. PRIMARY KEY
d. FOREIGN KEY

3.3.4 NOT NULL Constraint

The NOT NULL constraint enforces a column to NOT accept NULL values.

The NOT NULL constraint enforces a field to always contain a value. This means that you cannot insert a new record, or update a record without adding a value to this field.

The following SQL enforces the "P_Id" column and the "LastName" column to not accept NULL values:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

3.3.5 SQL UNIQUE Constraint

The UNIQUE constraint uniquely identifies each record in a database table.

The UNIQUE and PRIMARY KEY constraints both provide a guarantee for uniqueness for a column or set of columns.

A PRIMARY KEY constraint automatically has a UNIQUE constraint defined on it.

Note that you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

Example1: The following SQL creates a UNIQUE constraint on the "P_Id" column when the "Persons" table is created:

```
CREATE TABLE Persons
(
P_Id int NOT NULL UNIQUE,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

Example 2: To create a UNIQUE constraint on the "P_Id" column when the table is already created, use the following SQL:

```
ALTER TABLE Persons
ADD CONSTRAINT uc_PersonID UNIQUE (P_Id,LastName)
```

Example 3: To drop a UNIQUE constraint, use the following SQL:

```
ALTER TABLE Persons
DROP CONSTRAINT uc_PersonID
```

3.3.6 PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a database table. Primary keys must contain unique values. A primary key column cannot contain NULL values.

Each table should have a primary key, and each table can have only one primary key.

Example 1:The following SQL creates a PRIMARY KEY on the "P_Id" column when the "Persons" table is created:

```
CREATE TABLE Persons
(
P_Id int NOT NULL PRIMARY KEY,
```

```
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

Example 2: To create a PRIMARY KEY constraint on the "P_Id" column when the table is already created, use the following SQL:

```
ALTER TABLE Persons
ADD CONSTRAINT pk_PersonID PRIMARY KEY (P_Id,LastName)
```

3.3.7 SQL FOREIGN KEY Constraint

A FOREIGN KEY in one table points to a PRIMARY KEY in another table.

Let us illustrate the foreign key with an example. Look at table 7.1 above and table 7.5:

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Table 7.5: Orders Table
Source: http://www.w3schools.com/

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Note that the "P_Id" column in the "Orders" table points to the "P_Id" column in the "Persons" table.

The "P_Id" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

The "P_Id" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

The FOREIGN KEY constraint is used to prevent actions that would destroy link between tables.

The FOREIGN KEY constraint also prevents that invalid data is inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

Example 1:The following SQL creates a FOREIGN KEY on the "P_Id" column when the "Orders" table is created:

```
CREATE TABLE Orders
(
O_Id int NOT NULL PRIMARY KEY,
OrderNo int NOT NULL,
P_Id int FOREIGN KEY REFERENCES Persons(P_Id)
)
```

Example 2: To create a FOREIGN KEY constraint on the "P_Id" column when the "Orders" table is already created, use the following SQL:

```
ALTER TABLE Orders
ADD FOREIGN KEY (P_Id)
REFERENCES Persons(P_Id)
```

3.3.7 USE

The USE command allows you to specify the database you wish to work with within your DBMS.

```
USE employees
```

3.3.8 ALTER

Once you have 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. Take a look at the following command:

```
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.

3.3.9 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

Use this command with care! Remember that the DROP command removes entire data structures from your database. If you want to remove individual records, use the DELETE command of the Data Manipulation Language.

Activity A

1a. what do you understand by Data Definition Language? Then list some of the available DDL commands

b. Write briefly on the following commands:

i. Create
ii. Use
iii. Alter iv. Drop