Introduction to Database Systems(Lab)

Engr. Madeha Mushtaq
Department of Computer Science
Iqra National University

Structured Query Language(SQL)

- SQL stands for Structured Query Language.
- SQL is a standard language for storing, manipulating and retrieving data in databases.
- Softwares:
- MySQL,
- SQL Server,
- Oracle,
- Sybase,
- Informix,
- Postgres, and other database systems.

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

SQL Statements

- Most of the actions you need to perform on a database are done with SQL statements.
- **SELECT Statement:**
- The SELECT statement is used to select data from a database.
- The data returned is stored in a result table, called the result-set.
- **SELECT Syntax**
- SELECT *column1*, *column2, ...*
  FROM *table_name*;
- Here, column1, column2, ... are the field names of the table you want to select data from.

SELECT Statement

- If you want to select all the fields available in the table, use the following syntax:
- SELECT * FROM *table_name*;
- **SELECT Statement Example:**
- The following SQL statement selects the "CustomerName" and "City" columns from the "Customers" table:

- SELECT CustomerName,City FROM Customers;

- The following SQL statement selects all the columns from the "Customers" table:

SELECT DISTINCT Statement

- The SELECT DISTINCT statement is used to return only distinct (different) values.
- Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.
- **SELECT DISTINCT Syntax**
- SELECT DISTINCT *column1*, *column2, ...*
  FROM *table_name*;

SELECT DISTINCT Statement

- **SELECT DISTINCT Examples**
- The following SQL statement selects only the DISTINCT values from the "Country" column in the "Customers" table:
- SELECT DISTINCT Country FROM Customers;

- The following SQL statement lists the number of different (distinct) customer countries:

- SELECT COUNT(DISTINCT Country) FROM Customers;

The SQL WHERE Clause

- The WHERE clause is used to filter records.
- The WHERE clause is used to extract only those records that fulfill a specified condition.
- The WHERE clause is not only used in SELECT statement, it is also used in UPDATE, DELETE statement, etc.

- **WHERE Syntax**
- SELECT *column1*, *column2, ...*
  FROM *table_name*
  WHERE *condition*;

The SQL WHERE Clause

- Example:
- The following SQL statement selects all the customers from the country "Mexico", in the "Customers" table:
- SELECT * FROM Customers
  WHERE Country='Mexico';

The SQL WHERE Clause

- **Text Fields vs. Numeric Fields:**
- SQL requires single quotes around text values (most database systems will also allow double quotes).
- However, numeric fields should not be enclosed in quotes:

- SELECT * FROM Customers
  WHERE CustomerID=1;

SQL AND, OR and NOT Operators

- The WHERE clause can be combined with AND, OR, and NOT operators.
- The AND and OR operators are used to filter records based on more than one condition:
- The AND operator displays a record if all the conditions separated by AND are TRUE.
- The OR operator displays a record if any of the conditions separated by OR is TRUE.
- The NOT operator displays a record if the condition(s) is NOT TRUE.

SQL AND, OR and NOT Operators

- **AND Syntax**
- SELECT *column1*, *column2, ...*
  FROM *table_name*
  WHERE *condition1* AND *condition2* AND *condition3 ...*;
- **OR Syntax**
- SELECT *column1*, *column2, ...*
  FROM *table_name*
  WHERE *condition1* OR *condition2* OR *condition3 ...*;
- **NOT Syntax**
- SELECT *column1*, *column2, ...*
  FROM *table_name*
  WHERE NOT *condition*;

SQL AND, OR and NOT Operators

- **AND Example**
- The following SQL statement selects all fields from "Customers" where country is "Germany" AND city is "Berlin":
- SELECT * FROM Customers
  WHERE Country='Germany' AND City='Berlin';
- **OR Example**
- The following SQL statement selects all fields from "Customers" where country is "Germany" OR "Spain":
- Example
- SELECT * FROM Customers
  WHERE Country='Germany' OR Country='Spain';

SQL AND, OR and NOT Operators

- **NOT Example**
- The following SQL statement selects all fields from "Customers" where country is NOT "Germany":
- Example
- SELECT * FROM Customers
  WHERE NOT Country='Germany';

SQL AND, OR and NOT Operators

- Combining AND, OR and NOT
- You can also combine the AND, OR and NOT operators.
- The following SQL statement selects all fields from "Customers" where country is "Germany" AND city must be "Berlin" OR "München" (use parenthesis to form complex expressions):
- Example
- SELECT * FROM Customers
  WHERE Country='Germany' AND (City='Berlin' OR City='Münc
  hen');

SQL AND, OR and NOT Operators

- The following SQL statement selects all fields from "Customers" where country is NOT "Germany" and NOT "USA":
- Example
- SELECT * FROM Customers
  WHERE NOT Country='Germany' AND NOT Country='USA';

The SQL ORDER BY Keyword

- The ORDER BY keyword is used to sort the result-set in ascending or descending order.
- The ORDER BY keyword sorts the records in ascending order by default.
- To sort the records in descending order, use the DESC keyword.
- **ORDER BY Syntax**
- SELECT *column1*, *column2, ...*
  FROM *table_name*
  ORDER BY *column1, column2, ...* ASC|DESC;

The SQL ORDER BY Keyword

- Example
- The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" column:
- SELECT * FROM Customers
  ORDER BY Country;
- DESC Example
- The following SQL statement selects all customers from the "Customers" table, sorted DESCENDING by the "Country" column:
- Example
- SELECT * FROM Customers
  ORDER BY Country DESC;

The SQL ORDER BY Keyword

- The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" and the "CustomerName" column.
- This means that it orders by Country, but if some rows have the same Country, it orders them by CustomerName:

- SELECT * FROM Customers
  ORDER BY Country, CustomerName;

SQL INSERT INTO Statement

- The INSERT INTO statement is used to insert new records in a table.
- **INSERT INTO Syntax**
- It is possible to write the INSERT INTO statement in two ways.
- The first way specifies both the column names and the values to be inserted:

- INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...) VALUES (*value1*, *value2*, *value3*, ...);

SQL INSERT INTO Statement

- If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query.
- However, make sure the order of the values is in the same order as the columns in the table.
- The INSERT INTO syntax would be as follows:

- INSERT INTO *table_name*
  VALUES (*value1*, *value2*, *value3*, ...);

SQL INSERT INTO Statement

- Example:
- The following SQL statement inserts a new record in the "Customers" table:

- INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
  VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

SQL INSERT INTO Statement

- Insert Data Only in Specified Columns
- It is also possible to only insert data in specific columns.
- The following SQL statement will insert a new record, but only insert data in the "CustomerName", "City", and "Country" columns (CustomerID will be updated automatically):

- INSERT INTO Customers (CustomerName, City, Country)
  VALUES ('Cardinal', 'Stavanger', 'Norway');

End of Slides