CS105.3
Database Management Systems
Structured Query Language

By: Chalani Oruthotaarachchi

What is SQL?

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987
- There are different versions of the SQL language.
- Free from www.mysql.com

Query

- When a user wants to get some information from a database file, he can issue a query.
- A query is a user request to retrieve data or information with a certain condition.
- The result of the query will then be stored in form of a table.
- The program will go through all the records in the database file and select those records that satisfy the condition(searching).

SQL query types

- SQL is divided into three types of primary language statements.
  - **DML (Data Manipulation Language)**
    - How the data should reside in the database.
      - E.g.: SELECT, UPDATE, INSERT
  - **DDL (Data Definition Language)**
    - Used to store, modify, retrieve, delete and update data in database.
      - E.g.: CREATE, ALTER, DROP
  - **DCL (Data Control Language)**
    - Concerned with rights, permissions and other controls of the database
      - E.g.: GRANT, REVOKE

SQL syntax

- SQL keywords are NOT case sensitive
  - “select” is the same as “SELECT”.

- Some database systems require a semicolon at the end of each SQL statement.
  - Semicolon is the standard way to separate each SQL statement in database systems. That allow more than one SQL statement to be executed in the same call to the server.

Basic MySQL Operations

- Create database
- Create table
- Insert records
- Load data
- Retrieve records
- Update records
- Delete records
- Modify table
- Join table
- Drop table
- Optimize table
- Count, Like, Order by, Group by
- More advanced ones (sub-queries, stored procedures, triggers, views …)

SQL DATABASE

- The CREATE DATABASE statement is used to create a new SQL database.

  CREATE DATABASE *databasename*;

- The DROP DATABASE statement is used to drop an existing SQL database.

  DROP DATABASE *databasename*;

- The BACKUP DATABASE statement is used in SQL Server to create a full back up of an existing SQL database.

  BACKUP DATABASE *databasename*;

Data Types

- MySQL uses many different data types broken into three categories
  - Numeric
    - INT (4 bytes), BIT (0 or 1),
      TINYINT (0 to 255 -1 byte), SMALLINT(2 bytes),
      BIGINT (8 bytes)
  - Date and Time
    - DATETIME, SMALLDATETIME, DATE, TIME
  - String Types
    - CHAR(L), VARCHAR(L), TEXT

Read More:
https://dev.mysql.com/doc/refman/8.0/en/data-types.html

Field Size

- Maximum number of characters that can be saved in a particular field as attribute value.

- Field size is define with the data type.

E.g.
–CHAR (4)
–VARCHAR (20)

No need to specify the field sizes for INTEGER and DATE data types.

CREATE TABLE

- The table creation command requires the following details
  - Name of the table
  - Name of the fields
  - Definitions for each field

```
CREATE TABLE table_name
(field_name1  data_type(field_size),
field_name2  data_type(field_size),
….
….
field_nameN  data_type(field_size)
);
```

CREATE TABLE contd.

CREATE TABLE **table_name** (column_name column_type);

```
create table Book(
   Book_id INT NOT NULL AUTO_INCREMENT,
   Book_title VARCHAR(100) NOT NULL,
   Book_author VARCHAR(40) NOT NULL,
   Submission_date DATE,
   PRIMARY KEY (book_id )
);
```

Table named Book

Book_id	Book_title	Book_author	Submission_date

- Field Attribute **NOT NULL** is being used because we do not want this field to be NULL. So, if a user will try to create a record with a NULL value, then MySQL will raise an error.

- Field Attribute **AUTO_INCREMENT** tells MySQL to go ahead and add the next available number to the id field.

- Keyword **PRIMARY KEY** is used to define a column as a primary key. You can use multiple columns separated by a comma to define a primary key.

CREATE A TABLE TO STORE FOLLOWING DETAILS. DECIDE ON THE FIELD DATA TYPE.

Fields of the table- **STUDENT**:

–Student ID
–Student Name
–Address
–DOB
–Contact no
–Gender

Sample :

(1, ‘Anne Perera’, ‘105, Cinnoman Gardens, Colombo 7’, ‘13/10/1995’, ‘0115446000’, ‘F’);

DELETE TABLE

DROP TABLE <Table_Name>;

Exercise:

Delete the Student table that you have created earlier.

Insert Data into a Table

Insert Records to a Table

```
INSERT INTO table_name
VALUES (… , …. , …..);
```

INSERT INTO Student
VALUES (1001, ‘Ann Perera’, ‘105, Cinnoman Gardens, Colombo 7’, ‘1995/10/13’, ‘0115446000’, ‘F’);

Insert Data Selected field in a Table

```
INSERT INTO table_name (field1,field2,...fieldN )
  VALUES (value1, value2,...valueN );
```

INSERT INTO Student (StudentID, ContactNo, Age, StudentName)
VALUES (1001, ‘0115446000’, 22, ‘Ann Perera’);

Filter Data from a Table

Show all columns

SELECT * from *Table Name;*

SELECT * from Student;

Show selected columns in a Table

SELECT field 1, field 2,….. from *Table Name*;

SELECT Student_no, Student_name from Student;

Exercise

Write SQL commands to

1. Create below table.
2. Add first record into the table.
3. Display all the details of students.
4. Display ID and name of all students.

StudentID	FirstName	Surname	DOB	Gender
1001	Kate	West	12/10/1994	F
1002	Julie	McLain	3/7/1995	F
1003	Tom	Smith	24/12/1994	M
1004	Mark	Foster	5/11/1996	M
1005	Jane	Knight	17/6/1995	F
1006	Matt	Smith	24/12/1995	M
1007	Karen	Edwards	3/7/1995	M
1008	John	Williams	15/11/1996	M
1009	Allison	Cambell	10/10/1994	M
1010	Shirley	Thomas	15/4/1995	F

Filter Data: Where

Used to filter records.

It is used to extract only the records that fulfill a specified condition.

It is mostly used in statements like:
–SELECT
–UPDATE
–DELETE

SELECT field1, field2,...fieldN
FROM table_name1, table_name2...
[WHERE Clause]

Select the records which satisfy a condition with all the fields of the table:

```
SELECT * FROM <Table_Name>
WHERE <Condition>;
```

Select the records which satisfy a condition with specific fields in a table:

```
SELECT <Field names>
FROM <Table_Name>
WHERE <Condition>;
```

Basic SELECT statements with Operators:

Operator	Description
=	Equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

```
SELECT * from Book
WHERE book_author = ‘Elmasri';
```

```
SELECT * from Book
WHERE book_ID < 1000;
```

```
SELECT * FROM Book
WHERE Book_Price BETWEEN 30.00 AND 50.00;
```

```
SELECT * FROM Book
WHERE Country IN ('Germany', 'France', 'UK');
```

```
SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'France', 'UK');
```

Delete records from table

```
DELETE from table_name
Where <condition>;
```

E.g.:

Delete from Customers

where CustomerID = 92;

Exercise

1. Display all the details of female students.
2. Display ID and first name of all students whose surname is Smith.
3. Display all the details of students with StudentID less than 1005.
4. Display DOB of students with StudentID between 1005 and 1010.
5. Delete Student record of Student ID is 1010.

StudentID	FirstName	Surname	DOB	Gender
1001	Kate	West	12/10/1994	F
1002	Julie	McLain	3/7/1995	F
1003	Tom	Smith	24/12/1994	M
1004	Mark	Foster	5/11/1996	M
1005	Jane	Knight	17/6/1995	F
1006	Matt	Smith	24/12/1995	M
1007	Karen	Edwards	3/7/1995	M
1008	John	Williams	15/11/1996	M
1009	Allison	Cambell	10/10/1994	M
1010	Shirley	Thomas	15/4/1995	F

Logical Operators

- SQL allows you to have multiple conditions in a query through the use of logical operators: AND, OR and NOT.
- NOT has the highest precedence, followed by AND, and then followed by OR.
- However, you are strongly recommended to use parentheses to clarify the intended meaning of the query.

Logical Operator : AND

- This logical AND connective is used to set up a query where there are two conditions which must be met for the query to return the required row(s)

- Syntax: **AND , &&**

- Exercise:

  Write a query to display the employee number (EMP_NUM) and the attraction number (ATTRACT_NUM) for which the numbers of hours worked (HOURS_PER_ATTRACT) by the employee is greater than 3 and the date worked (DATE_WORKED) is after 18th May 2007 from the HOURS table.

```
SELECT EMP_NUM, ATTRACT_NO
FROM HOURS
WHERE HOURS_PER_ATTRACT > 3  AND
DATE_WORKED > '18-MAY-07';
```

```
SELECT EMP_NUM, ATTRACT_NO
FROM HOURS
WHERE HOURS_PER_ATTRACT > 3  &&
DATE_WORKED > '18-MAY-07';
```

Logical Operator : NOT

- The logical operator **NOT** is used to negate the result of a conditional expression.

- Syntax: **NOT , !**

- Exercise:

  Write a query to display the a listing of all rows for which EMP_NUM is not 106 from EMPLOYEE table

```
SELECT *
FROM EMPLOYEE
WHERE NOT (EMP_NUM = 106);
```

```
SELECT *
FROM EMPLOYEE
WHERE  ! (EMP_NUM = 106);
```

Note: that the condition is enclosed in parentheses; that practice is optional, but it is highly recommended for clarity

Logical Operator : OR

- The logical operator **NOT** is used to negate the result of a conditional expression.

- Syntax: **OR , ||**

- Exercise:

  Write a query to display list the names and countries of all Theme parks where PARK_COUNTRY = 'FR' OR PARK_COUNTRY = 'UK' from PARK table.

```
SELECT PARK_NAME, PARK_COUNTRY
FROM PARK
WHERE PARK_COUNTRY = 'FR'  OR PARK_COUNTRY =
'UK';
```

```
SELECT PARK_NAME, PARK_COUNTRY
FROM PARK
WHERE PARK_COUNTRY = 'FR'  || PARK_COUNTRY =
'UK';
```

Exercise

Write query to display customer name for customers who are located in the USA or France and have credit limit greater than 10000 from the Customer table.

Sorting Data

- The **ORDER BY** clause is especially useful when the listing order of the query is important.

- Although you have the option of declaring the order type—ascending (**ASC**) or descending (**DESC**) —the default order is ascending.

For example, if you want to display all employees listed by EMP_HIRE_DATE in descending order you would write the following query

```
SELECT *
FROM EMPLOYEE
ORDER BY EMP_HIRE_DATE DESC;
```

Describe in your own words what this query is actually doing.

```
SELECT TICKET_TYPE, PARK_CODE
FROM TICKET
WHERE (TICKET_PRICE > 15 AND TICKET_TYPE =
'Child')
ORDER BY TICKET_NO DESC;
```

Basic SQL Aggregate Functions

Function Name	Purpose	Example
COUNT	The number of rows containing non-null values	SELECT COUNT(PARK_CODE) FROM ATTRACTION;
DISTINCT	Produce a list of only those values that are different from one another **Not a Aggregate function	SELECT DISTINCT(PARK_CODE) FROM ATTRACTION; SELECT COUNT(DISTINCT(PARK_CODE)) FROM ATTRACTION;

Basic SQL Aggregate Functions

Function Name	Purpose	Example
MIN	The minimum attribute value encountered in a given column	SELECT MIN(TICKET_PRICE) FROM TICKET;
MAX	The maximum attribute value encountered in a given column	SELECT max(TICKET_PRICE) FROM TICKET;

Basic SQL Aggregate Functions

Function Name	Purpose	Example
SUM	The sum of all values for a given column	SELECT SUM(LINE_QTY) FROM SALES_LINE;
AVG	The arithmetic mean (average) for a specified column	SELECT AVG(LINE_PRICE) FROM SALES_LINE;

GROUP BY

The GROUP BY clause is generally used when you have attribute columns combined with aggregate functions in the SELECT statement.

It is valid only when used in conjunction with one of the SQL aggregate functions, such as COUNT, MIN, MAX, AVG and SUM.

The GROUP BY clause appears after the WHERE statement.

When using GROUP BY you should include all the attributes that are in the SELECT statement that do not use an aggregate function.

```
SELECT column_name(s)
FROM table_name
WHERE <condition>
GROUP BY column_name(s);
```

```
SELECT PARK_CODE,
MIN(TICKET_PRICE),MAX(TICKET_PRICE)
FROM TICKET
GROUP BY PARK_CODE;
```

Exercise

Write query to display

1. Maximum Customer ID
2. Minimum customer ID
3. Count of the customers
4. The count of the customers in each country.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

End of the session