SQL Programming
DDL and DML Languages
BootCamp for Database Technologies

Lecturer: Dr. Muhammad Iqbal

CCT College Dublin

Email: miqbal@cct.ie

Outline

- **Purpose and importance of SQL.**
- **How to write an SQL command.**
- **How to retrieve data from database using SELECT and:**
  - **Use simple and compound WHERE conditions.**
  - **Sort query results using ORDER BY.**
  - **Use AGGREGATE functions.**
  - **Group data using GROUP BY and HAVING.**
- **How to update database using INSERT, UPDATE, and DELETE.**

Objectives of SQL

- **Ideally, database language should allow user to**
    1. **Create the database and relation structures**
    2. **Perform insertion, modification, deletion of data from the relations**
    3. **Perform simple and complex queries**
- **A database language must perform these tasks with minimal user effort, and its command structure and syntax must be relatively easy to learn.**
- **It must be portable (capable of being used on different computer systems).**

Objectives of SQL

- **SQL is relatively easy to learn**
  - **it is non-procedural**
  - **we can specify *what* information requires for a specific task, rather than *how* to get it**
  - **it is essentially free-format**

- **SQL is a transform-oriented language with two major components:**

  i. **A DDL (Data Definition Language) for defining database structure. For example, CREATE tables, indexes, views, DROP OR ALTER tables.**

SQL Commands

mysql> CREATE Schema/ Database TestingSQL

- **Consists of standard English words:**

```
1) CREATE TABLE Staff (    Sno VARCHAR(5),
                           fName VARCHAR(15),
                           lName VARCHAR(15),
                           StartDate DATE,
                           salary DECIMAL(7,2),
                           Primary Key (Sno));
2) INSERT INTO Staff VALUES ('SG16', 'Peter', 'Brown', '2000-01-17', 8300);
3) UPDATE staff  SET salary = 9000 WHERE sno = 'SG16’;
4) SELECT Sno, lName, salary   FROM     Staff
                               WHERE salary > 10000;
```

Datatypes in SQL

- **MySQL supports a number of SQL standard data types in various categories. Following Data types are based on MySQL community server 5.6.**

1. **String**
2. **Numeric**
3. **DATETIME**

Data Type Syntax	Maximum Size	Explanation
DATE	Values range from '1000-01-01' to '9999-12-31'.	Displayed as 'YYYY-MM-DD'.
DATETIME	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIMESTAMP(*m*)	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC.	Displayed as 'YYYY-MM-DD HH:MM:SS'.
TIME	Values range from '-838:59:59' to '838:59:59'.	Displayed as 'HH:MM:SS'.
YEAR[(2|4)]	Year value as 2 digits or 4 digits.	Default is 4 digits.

Writing SQL Commands

SQL statement consists of *reserved words* and *user-defined words*.

- **Reserved words are a fixed part of SQL and must be spelt exactly as required and cannot be split across lines.**
- **User-defined words are made up by user and represent names of various database objects such as relations, columns and views.**

- **Most components of an SQL statement are *case insensitive*, except for literal character data. It is more readable with indentation and lineation**
  - **Each clause should begin on a new line.**
  - **Start of a clause should line up with start of other clauses.**
  - **If the clause has several parts, should each appear on a separate line and be indented under start of clause.**

SELECT Statement

```
SELECT [DISTINCT | ALL]
    {* | [columnExpression [AS newName]] [,...] }
FROM        TableName [alias] [, ...]
[WHERE      condition]
[GROUP BY   columnList]
[HAVING     condition]
[ORDER BY   columnList]
```

Optional Component

Required Element

SELECT	Specifies which columns are to appear in the output.
FROM	Specifies table(s) to be used.
WHERE	Filters rows (Checking condition).
GROUP BY	Forms groups of rows with same column value.
HAVING	Filters groups subject to some condition.
ORDER BY	Specifies the order of the output.

- Order of the clauses cannot be changed.
- Only SELECT and FROM are mandatory.

All and Specific Columns

List full details of all staff.

SELECT Sno, fName, lName, address, position, sex, DOB, salary, Bno FROM Staff;

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

- **Can also use * as an abbreviation for 'all columns':**

**SELECT *
FROM Staff;**

Specify an asterisk (*)

Produce a list of salaries for all staff, showing only = staff number, first and last names, and salary.

SELECT Sno, fName, lName, salary

FROM Staff;

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG37	Ann	Beech	12000.00
SG14	David	Ford	18000.00
SA9	Mary	Howe	9000.00
SG5	Susan	Brand	24000.00
SL41	Julie	Lee	9000.00

Use of DISTINCT

List the property numbers of all properties that have been viewed.

```
SELECT Pno
FROM Viewing;
```

propertyNo
PA14
PG4
PG4
PA14
PG36

- **Use DISTINCT to eliminate duplicates:**

```
SELECT DISTINCT Pno
FROM Viewing;
```

propertyNo
PA14
PG4
PG36

Calculation on Data Fields

Produce a list of monthly salaries for all staff, showing staff number, first/last name, and salary.

```
SELECT Sno, fName, lName, salary/12
FROM Staff;
```

- **To name column, use AS clause:**

```
SELECT Sno, fName, lName,
salary/12 AS monthlySalary
FROM Staff;
```

staffNo	fName	lName	salary/12
SL21	John	White	2500.00
SG37	Ann	Beech	1000.00
SG14	David	Ford	1500.00
SA9	Mary	Howe	750.00
SG5	Susan	Brand	2000.00
SL41	Julie	Lee	750.00

Row selection (WHERE clause)

- **Restriction of rows is possible with the WHERE clause, which consists of the keyword WHERE followed by a search condition that specifies the rows to be retrieved. For example,**

1. ***Comparison:*** **Compare the value of one expression to the value of another expression. (Single and multiple values/ points of comparison)**

2. ***Range:*** **Test whether the value of an expression falls within a specified range of values. (called as Range query)**

3. ***Set Membership:*** **Test whether the value of an expression equals to one of a set of values.**

4. ***Pattern Match:*** **Test whether a string matches to a specified pattern. (Pre and post fix match)**

5. ***Null:*** **Test whether a column has a null (unknown) value.**

1. Comparison Search Condition

List all staff with a salary greater than 10,000.

```
SELECT Sno, fName, lName, position, salary
        FROM Staff
        WHERE salary > 10000;
```

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG37	Ann	Beech	Assistant	12000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

Logical comparison operators

=, <, <=, >, >=, and <>

1. Compound Comparison Search Condition

List addresses of all branch offices in London or Glasgow.

```
SELECT *
FROM Branch
WHERE city = 'London' OR city = 'Glasgow';
```

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B003	163 Main St	Glasgow	G11 9QX
B002	56 Clover Dr	London	NW10 6EU

2. Range Search Condition

List all staff with a salary between 20,000 and 30,000.

```
SELECT Sno, fName, lName, position, salary
FROM Staff
WHERE salary BETWEEN 20000 AND 30000;
```

- **BETWEEN test includes the endpoints of range.**

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG5	Susan	Brand	Manager	24000.00

3. Set Membership

List all managers and supervisors.

```
SELECT Sno, fName, lName, position
FROM Staff
WHERE position IN ('Manager', 'Supervisor');
```

staffNo	fName	lName	position
SL21	John	White	Manager
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

4. Pattern Matching

Find all owners with the string 'Glasgow' in their address.

```
SELECT Ono, fName, lName, address, Tel_No
FROM Owner
WHERE address LIKE '%Glasgow%';
```

ownerNo	fName	lName	address	telNo
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

- **The LIKE keyword used in a WHERE operator with a wildcard (% or _) allows you to search for patterns in character-based fields.**

4. Pattern Matching

- **For example:** Address **LIKE 'H%'** means the first character must be **H**, but the rest of the string can be anything.
- Address **LIKE 'H_ _ _'** means that there must be exactly four characters in the string, the first of which must be an **H**.
- Address **LIKE '%e'** means any sequence of characters, of length at least **1**, with the last character an **'e'**.
- Address **LIKE '%Glasgow%'** means a sequence of characters of any length containing Glasgow.
- Address **NOT LIKE 'H%'** means first character cannot be an **H**.

5. NULL Search Condition

List details of all viewings on property PG04 where a comment has not been supplied.

- **There are 2 viewings for property PG04, one with and one without a comment.**
- **Have to test for null explicitly using special keyword IS NULL:**

SELECT Rno, Date

FROM Viewing

WHERE Pno = 'PG04' AND comment IS NULL;

Rno	Date
CR56	26-May-04

Single Column Ordering

List salaries for all staff, arranged in descending order of salary.

```
SELECT Sno, fName, lName, salary
  FROM Staff
  ORDER BY salary DESC;
```

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG5	Susan	Brand	24000.00
SG14	David	Ford	18000.00
SG37	Ann	Beech	12000.00
SA9	Mary	Howe	9000.00
SL41	Julie	Lee	9000.00

Keyword DESC to see result in a descending order of values Keyword ASC to specify ascending order explicitly

Produce abbreviated list of properties in order of property type.

```
SELECT Pno, type, rooms, rent
  FROM Property_For_Rent
  ORDER BY type;
```

propertyNo	type	rooms	rent
PL94	Flat	4	400
PG4	Flat	3	350
PG36	Flat	3	375
PG16	Flat	4	450
PA14	House	6	650
PG21	House	5	600

SELECT Statement - Aggregates

ISO standard defines five aggregate functions:

1. **COUNT** returns number of values in a specified column.
2. **SUM** returns sum of values in a specified column.
3. **AVG** returns average of values in a specified column.
4. **MIN** returns smallest value in a specified column.
5. **MAX** returns largest value in a specified column.

SELECT Statement - Aggregates

- **Each operates on a single column of a table and returns a single value.**
- **COUNT, MIN, and MAX apply to numeric and non-numeric fields, but SUM and AVG may be used on numeric fields only.**
- **Apart from COUNT(*), each function eliminates nulls first and operates only on remaining non-null values.**
- **COUNT(*) counts all rows of a table, regardless of whether nulls or duplicate values occur.**
- **Can use DISTINCT before column name to eliminate duplicates.**
- **DISTINCT has no effect with MIN/ MAX, but may have with SUM/ AVG.**

SELECT Statement - Aggregates

- **Aggregate functions can be used only in SELECT list and in HAVING clause.**
- **If SELECT list includes an aggregate function and there is no GROUP BY clause, SELECT list cannot reference a column out with an aggregate function. For example, the following is illegal**

```
SELECT COUNT(salary)
FROM Staff;
```

```
SELECT position, COUNT(salary)
FROM Staff
Group by position;
```

position	COUNT(salary)
Assistant	2
Deputy	1
Manager	2
Senior Assistant	1

Use of COUNT(*)

How many properties cost more than £350 per month to rent?

```
SELECT COUNT(*) AS myCount
 FROM Property_For_Rent
 WHERE rent > 350;
```

myCount
5

Use of COUNT(DISTINCT)

How many different properties viewed in May 13 1995?

```
SELECT COUNT(DISTINCT Pno) AS myCount
FROM Viewing
WHERE Date
BETWEEN '1995-04-01' AND '1995-04-31';
```

myCount
2

Format of Date is very important in comparison

help DATE

Use of COUNT and SUM

Find number of Managers and sum of their salaries.

```
SELECT COUNT(Sno) AS myCount,
SUM(salary) AS mySum
FROM Staff
WHERE position = 'Manager';
```

Aggregate Functions

myCount	mySum
2	54000.00

http://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html

Use of MIN, MAX, AVG

Find minimum, maximum, and average staff salary.

```
SELECT MIN(salary) AS myMin,
MAX(salary) AS myMax,
AVG(salary) AS myAvg
FROM Staff;
```

myMin	myMax	myAvg
9000.00	30000.00	17000.00

SELECT Statement - Grouping

- **Use GROUP BY clause to get sub-totals.**
- **SELECT and GROUP BY closely integrated: each item in the SELECT list must be *single-valued per group*, and SELECT clause may only contain**
  - **Column names**
  - **Aggregate functions**
  - **Constants**
  - **Expression involving combinations of the above**

SELECT Statement - Grouping

- **All column names in the SELECT list must appear in GROUP BY clause unless name is used only in an aggregate function.**

- **If WHERE is used with GROUP BY, WHERE is applied first, then groups are formed from remaining rows satisfying predicate.**

- **ISO considers two nulls to be equal for purposes of GROUP BY.**

Use of GROUP BY

Find number of staff in each branch and their total salaries.

```
SELECT Bno,
       COUNT(Sno) AS myCount,
       SUM(salary) AS mySum
FROM Staff
GROUP BY Bno
ORDER BY Bno;
```

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00
B007	1	9000.00

Restricted Groupings
HAVING clause

- **HAVING clause is designed for use with GROUP BY to restrict groups that appear in the final result table.**

- **Similar to WHERE, but WHERE filters individual rows whereas HAVING filters groups.**

- **Column names in HAVING clause must also appear in the GROUP BY list or be contained within an aggregate function.**

Use of HAVING

For each branch with more than 1 member of staff, find number of staff in each branch and sum of their salaries.

```
SELECT Bno,
COUNT(Sno) AS myCount,
SUM(salary) AS mySum
FROM Staff
GROUP BY Bno
HAVING COUNT(Sno) > 1
ORDER BY Bno;
```

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00

References/ Resources

- Thomas Connolly, Carolyn Begg 2014, Database Systems: A Practical Approach to Design, Implementation, and Management, 6th Edition Ed., Pearson Education [ISBN: 1292061189] [Present in our Library]
- https://www.mamp.info/en/mamp/windows/