Ch 7

Database Management System

Sumayyea Salahuddin (Lecturer)
Dept. of Computer Systems Eng.
UET Peshawar

Objectives

- Advance SQL (Part I)
  - Use of relational set operators UNION, UNION ALL, INTERSECT (Equivalent), and MINUS (Equivalent)
  - How to use the advanced SQL JOIN operator syntax
  - About the different types of sub-queries and correlated queries
  - How to use SQL functions to manipulate dates, strings, and other data

Relational Set Operators

- UNION
- INTERSECT
- MINUS
- Work properly if relations are **union-compatible**
  - Names of relation attributes must be the same and their data types must be identical

UNION

- Combines rows from two or more queries without including duplicate rows
  - Example:

```
SELECT    CUS_LNAME, CUS_FNAME,
          CUS_INITIAL, CUS_AREACODE,
FROM      CUSTOMER
UNION
SELECT    CUS_LNAME, CUS_FNAME,
          CUS_INITIAL, CUS_AREACODE,
FROM      CUSTOMER_2;
```

- Can be used to unite more than two queries

UNION ALL

- Produces a relation that retains duplicate rows
  - Example query:

```
SELECT      CUS_LNAME, CUS_FNAME,
            CUS_INITIAL, CUS_AREACODE,
FROM        CUSTOMER
UNION ALL
SELECT      CUS_LNAME, CUS_FNAME,
            CUS_INITIAL, CUS_AREACODE,
FROM        CUSTOMER_2;
```

- Can be used to unite more than two queries

Intersect

- Combines rows from two queries, returning only the rows that appear in both sets
- Syntax: *query* INTERSECT *query*
  - Example query:

```
SELECT      CUS_CODE
FROM        CUSTOMER
WHERE       CUS_AREACODE='615'
INTERSECT
SELECT      DISTINCT CUS_CODE
FROM        INVOICE;
```

Minus

- Combines rows from two queries
  - Returns only the rows that appear in the first set but not in the second
- Syntax: *query* MINUS *query*
  - Example:

```
SELECT      CUS_CODE
FROM        CUSTOMER
WHERE       CUS_AREACODE='615'
MINUS
SELECT      DISTINCT CUS_CODE
FROM        INVOICE;
```

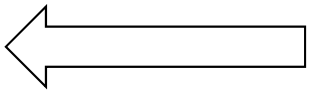
Syntax Alternatives

- IN and NOT IN subqueries can be used in place of INTERSECT and MINUS
- Example:

```
SELECT          CUS_CODE
FROM            CUSTOMER
WHERE           CUS_AREACODE = '615'
WHERE  CUS_CODE IN (
SELECT DISTINCT CUS_CODE
FROM            INVOICE);
```

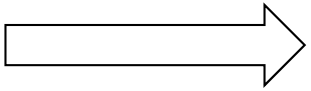
Example

```
mysql> select FirstName, LastName, Team from member
    -> where memberID in (
    -> select memberID from member where Gender = 'M');
+-----------+----------+--------+
| FirstName | LastName | Team   |
+-----------+----------+--------+
| Michael   | Stone    | NULL   |
| Thomas    | Spence   | NULL   |
| William   | Cooper   | Team B |
| Robert    | Pollard  | Team B |
+-----------+----------+--------+
4 rows in set (0.00 sec)
```



Intersect

Minus



```
mysql> select FirstName, LastName, Team from member
    -> where memberID not in (
    -> select memberID from member where Gender = 'M');
+-----------+----------+--------+
| FirstName | LastName | Team   |
+-----------+----------+--------+
| Melissa   | McKenzie | NULL   |
| Brenda    | Nolan    | Team B |
| Helen     | Branch   | NULL   |
| Sarah     | Beck     | NULL   |
| Sandra    | Burton   | NULL   |
| Barbara   | Olson    | NULL   |
+-----------+----------+--------+
6 rows in set (0.00 sec)
```

SQL Join Operators

- Join operation merges rows from two tables and returns the rows with one of the following:
  - Have common values in common columns
    - Natural join
  - Meet a given join condition
    - Equality or inequality
  - Have common values in common columns or have no matching values
    - Outer join
- **Inner join**: only return rows meeting criteria

Cross Join

- Performs relational product of two tables
  - Also called Cartesian product
- Syntax:
  - SELECT *column-list* FROM *table1* CROSS JOIN *table2*
- Perform a cross join that yields specified attributes

Cross Join (Example)

```
mysql> select employee.emp_name, department.dep_Name
    -> from employee
    -> cross join
    -> department;
```

Cartesian Product

emp_name	dep_Name
BCA	Computer Systems Engineering
BCA	Mining Engineering
BCA	Civil Engineering
BCA	Chemical Engineering
BCA	Mechanical Engineering
BCA	Electrical Engineering
ABC	Computer Systems Engineering
ABC	Mining Engineering
ABC	Civil Engineering
ABC	Chemical Engineering
ABC	Mechanical Engineering
ABC	Electrical Engineering
XYZ	Computer Systems Engineering
XYZ	Mining Engineering
XYZ	Civil Engineering
XYZ	Chemical Engineering
XYZ	Mechanical Engineering
XYZ	Electrical Engineering
CVA	Computer Systems Engineering
CVA	Mining Engineering
CVA	Civil Engineering
CVA	Chemical Engineering
CVA	Mechanical Engineering
CVA	Electrical Engineering
VAC	Computer Systems Engineering
VAC	Mining Engineering
VAC	Civil Engineering
VAC	Chemical Engineering
VAC	Mechanical Engineering
VAC	Electrical Engineering
ACB	Computer Systems Engineering
ACB	Mining Engineering
ACB	Civil Engineering
ACB	Chemical Engineering
ACB	Mechanical Engineering
ACB	Electrical Engineering
XZY	Computer Systems Engineering
XZY	Mining Engineering
XZY	Civil Engineering
XZY	Chemical Engineering
XZY	Mechanical Engineering
XZY	Electrical Engineering
ABS	Computer Systems Engineering
ABS	Mining Engineering
ABS	Civil Engineering
ABS	Chemical Engineering
ABS	Mechanical Engineering
ABS	Electrical Engineering
FGJ	Computer Systems Engineering
FGJ	Mining Engineering
FGJ	Civil Engineering
FGJ	Chemical Engineering
FGJ	Mechanical Engineering
FGJ	Electrical Engineering
JKF	Computer Systems Engineering
JKF	Mining Engineering
JKF	Civil Engineering
JKF	Chemical Engineering
JKF	Mechanical Engineering
JKF	Electrical Engineering

60 rows in set (0.00 sec)

Natural Join

- Returns all rows with matching values in the matching columns
  - Eliminates duplicate columns
- Used when tables share one or more common attributes with common names
- Syntax:

  SELECT *column-list* FROM *table1* NATURAL JOIN *table2*

Natural Join (Example)

```
mysql> select employee.emp_name, department.dep_Name
    -> from employee
    -> natural join
    -> department;
+----------+-------------------------------+
| emp_name | dep_Name                      |
+----------+-------------------------------+
| BCA      | Computer Systems Engineering  |
| XYZ      | Computer Systems Engineering  |
| ACB      | Civil Engineering             |
| XZY      | Civil Engineering             |
| ABS      | Civil Engineering             |
| FGJ      | Civil Engineering             |
| ABC      | Electrical Engineering        |
| CVA      | Electrical Engineering        |
| VAC      | Electrical Engineering        |
| JKF      | Mechanical Engineering        |
+----------+-------------------------------+
10 rows in set (0.14 sec)
```

Join USING Clause

- Returns only rows with matching values in the column indicated in the USING clause
- Syntax:

  SELECT *column-list* FROM *table1* JOIN *table2* USING *(common-column)*
- JOIN USING operand does not require table qualifiers

Join Using (Example)

```
mysql> select employee.emp_name, department.dep_Name
    -> from employee
    -> join
    -> department
    -> using (dep_ID);
+----------+-----------------------------+
| emp_name | dep_Name                    |
+----------+-----------------------------+
| BCA      | Computer Systems Engineering |
| XYZ      | Computer Systems Engineering |
| ACB      | Civil Engineering           |
| XZY      | Civil Engineering           |
| ABS      | Civil Engineering           |
| FGJ      | Civil Engineering           |
| ABC      | Electrical Engineering      |
| CVA      | Electrical Engineering      |
| VAC      | Electrical Engineering      |
| JKF      | Mechanical Engineering      |
+----------+-----------------------------+
10 rows in set (0.05 sec)
```

JOIN ON Clause

- Used when tables have no common attributes
- Returns only rows that meet the join condition
  - Typically includes equality comparison expression of two columns
- Syntax: SELECT *column-list* FROM *table1* JOIN *table2* ON *join-condition*

Outer Joins

- Returns rows matching the join condition
- Also returns rows with unmatched attribute values for tables to be joined
- Three types
  - Left
  - Right
  - Full
- Left and right designate order in which tables are processed

Outer Joins (continued)

- Left outer join
  - Returns rows matching the join condition
  - Returns rows in left side table with unmatched values
  - Syntax: SELECT *column-list* FROM *table1* LEFT [OUTER] JOIN *table2* ON *join-condition*
- Right outer join
  - Returns rows matching join condition
  - Returns rows in right side table with unmatched values

Left Join (Example)

```
mysql> select employee.emp_name, department.dep_Name
    -> from employee
    -> left join
    -> department
    -> on
    -> department.dep_ID = employee.dep_ID;
+----------+------------------------------+
| emp_name | dep_Name                     |
+----------+------------------------------+
| BCA      | Computer Systems Engineering |
| XYZ      | Computer Systems Engineering |
| ACB      | Civil Engineering            |
| XZY      | Civil Engineering            |
| ABS      | Civil Engineering            |
| FGJ      | Civil Engineering            |
| JKF      | Mechanical Engineering       |
| ABC      | NULL                         |
| CVA      | NULL                         |
| VAC      | NULL                         |
+----------+------------------------------+
10 rows in set (0.00 sec)
```

Right Join (Example)

```
mysql> select employee.emp_name, department.dep_Name
    -> from employee
    -> right join
    -> department
    -> on
    -> department.dep_ID = employee.dep_ID;
+----------+-----------------------------+
| emp_name | dep_Name                    |
+----------+-----------------------------+
| BCA      | Computer Systems Engineering |
| XYZ      | Computer Systems Engineering |
| NULL     | Mining Engineering          |
| ACB      | Civil Engineering           |
| XZY      | Civil Engineering           |
| ABS      | Civil Engineering           |
| FGJ      | Civil Engineering           |
| NULL     | Chemical Engineering        |
| JKF      | Mechanical Engineering      |
| NULL     | Electrical Engineering      |
+----------+-----------------------------+
10 rows in set (0.00 sec)
```

Outer Joins (continued)

- Full outer join
  - Returns rows matching join condition
  - Returns all rows with unmatched values in either side table
  - Syntax:

```
SELECT      column-list
FROM        table1 FULL [OUTER] JOIN table2
            ON join-condition
```

Subqueries and Correlated Queries

- Often necessary to process data based on other processed data
- Subquery is a query inside a query, normally inside parentheses
- First query is the outer query
  - Inside query is the inner query
- Inner query executed first
- Output of inner query used as input for outer query
- Sometimes referred to as a nested query
- Example on slide 9

SQL Functions

- Generating information from data often requires many data manipulations
- SQL functions similar to functions in programming languages
- Functions always use numerical, date, or string value
- Value may be part of a command or attribute in a table
- Function may appear anywhere in an SQL statement

Date and Time Functions

- All SQL-standard DBMSs support date and time functions
- Date functions take one parameter and return a value
- Date/time data types implemented differently by different DBMS vendors
- ANSI SQL standard defines date data types, but not how data types are stored
- For more information, visit Table 12.13 @ https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html

Date and Time Functions Examples

```
mysql> SELECT CURDATE();
+------------+
| CURDATE()  |
+------------+
| 2018-05-17 |
+------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT WEEK('2018-05-18');
+--------------------+
| WEEK('2018-05-18') |
+--------------------+
|                 19 |
+--------------------+
1 row in set (0.10 sec)
```

```
mysql> SELECT DATE_FORMAT(CURDATE(), '%W, %dth %M %Y');
+-----------------------------------------+
| DATE_FORMAT(CURDATE(), '%W, %dth %M %Y') |
+-----------------------------------------+
| Thursday, 17th May 2018                 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT QUARTER('2018-05-18');
+-----------------------+
| QUARTER('2018-05-18') |
+-----------------------+
|                     2 |
+-----------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2018-05-17 09:55:20 |
+---------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT SYSDATE(), SLEEP(2), SYSDATE();
+---------------------+----------+---------------------+
| SYSDATE()           | SLEEP(2) | SYSDATE()           |
+---------------------+----------+---------------------+
| 2018-05-18 07:59:43 |        0 | 2018-05-18 07:59:45 |
+---------------------+----------+---------------------+
1 row in set (2.00 sec)
```

```
mysql> SELECT DATE_FORMAT(curdate(),GET_FORMAT(DATE,'EUR'));
+-----------------------------------------------+
| DATE_FORMAT(curdate(),GET_FORMAT(DATE,'EUR')) |
+-----------------------------------------------+
| 18.05.2018                                    |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

Numeric Functions

- Grouped in different ways
  - Algebraic, trigonometric, logarithmic, etc.
- Do not confuse with aggregate functions
  - Aggregate functions operate over sets
  - Numeric functions operate over single row
- Numeric functions take one numeric parameter and return one value
- For example: abs, round, ceil, floor, etc
- For more information, visit Table 12.10 @ https://dev.mysql.com/doc/refman/8.0/en/numeric-functions.html

Numeric Functions Examples

```
mysql> SELECT CEILING(1.23);
+---------------+
| CEILING(1.23) |
+---------------+
|             2 |
+---------------+
1 row in set (0.09 sec)

mysql> SELECT COS(PI());
+-----------+
| COS(PI()) |
+-----------+
|        -1 |
+-----------+
1 row in set (0.03 sec)

mysql> SELECT DEGREES(PI());
+---------------+
| DEGREES(PI()) |
+---------------+
|           180 |
+---------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT LOG10(100);
+------------+
| LOG10(100) |
+------------+
|          2 |
+------------+
1 row in set (0.06 sec)

mysql> SELECT MOD(29,9);
+-----------+
| MOD(29,9) |
+-----------+
|         2 |
+-----------+
1 row in set (0.06 sec)

mysql> SELECT RADIANS(90);
+--------------------+
| RADIANS(90)        |
+--------------------+
| 1.5707963267948966 |
+--------------------+
1 row in set (0.00 sec)
```

```
mysql>  SELECT ROUND(1.58);
+-------------+
| ROUND(1.58) |
+-------------+
|           2 |
+-------------+
1 row in set (0.00 sec)

mysql> SELECT SQRT(20);
+------------------+
| SQRT(20)         |
+------------------+
| 4.47213595499958 |
+------------------+
1 row in set (0.00 sec)

mysql> SELECT TRUNCATE(1.223,1);
+-------------------+
| TRUNCATE(1.223,1) |
+-------------------+
|               1.2 |
+-------------------+
1 row in set (0.00 sec)
```

String Functions

- String manipulations most used functions in programming
- String manipulation function examples:
  - Printing in uppercase (i.e. upper, lower)
  - Finding length of an attribute (i.e. length)
  - Finding part of string (i.e. substr)
- For more information, visit Table 12.7 @ https://dev.mysql.com/doc/refman/8.0/en/string-functions.html

String Functions Examples

```
mysql> SELECT CONCAT('My', 'S', 'QL');
+-------------------------+
| CONCAT('My', 'S', 'QL') |
+-------------------------+
| MySQL                   |
+-------------------------+
1 row in set (0.07 sec)

mysql> select upper('dcse')
    -> ;
+---------------+
| upper('dcse') |
+---------------+
| DCSE          |
+---------------+
1 row in set (0.06 sec)

mysql> select lower('DCSE')
    -> ;
+---------------+
| lower('DCSE') |
+---------------+
| dcse          |
+---------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT REPEAT('MySQL', 3);
+--------------------+
| REPEAT('MySQL', 3) |
+--------------------+
| MySQLMySQLMySQL    |
+--------------------+
1 row in set (0.06 sec)

mysql> SELECT REVERSE('abc');
+----------------+
| REVERSE('abc') |
+----------------+
| cba            |
+----------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT 'a' REGEXP '^[a-d]';
+---------------------+
| 'a' REGEXP '^[a-d]' |
+---------------------+
|                   1 |
+---------------------+
1 row in set (0.00 sec)
```

```
mysql> select length('DCSE');
+----------------+
| length('DCSE') |
+----------------+
|              4 |
+----------------+
1 row in set (0.06 sec)

mysql> SELECT LTRIM('  barbar');
+-------------------+
| LTRIM('  barbar') |
+-------------------+
| barbar            |
+-------------------+
1 row in set (0.00 sec)

mysql> SELECT RTRIM('barbar
+-------------------------+
| RTRIM('barbar        ') |
+-------------------------+
| barbar                  |
+-------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT REPLACE('www.mysql.com', 'w', 'Ww');
+-------------------------------------+
| REPLACE('www.mysql.com', 'w', 'Ww') |
+-------------------------------------+
| WWWWWW.mysql.com                    |
+-------------------------------------+
1 row in set (0.06 sec)
```

Conversion Functions

- Take a value of given data type and convert it to the equivalent value in another data type
- MySql uses CAST and CONVERT functions
- For more information, visit Table 12.14 @ https://dev.mysql.com/doc/refman/8.0/en/cast-functions.html

Summary

- Relational set operators combine output of two queries to generate new relation
- Operations that join tables classified as inner joins and outer joins
- Natural join returns all rows with matching values in the matching columns
  - Eliminates duplicate columns
- Subqueries and correlated queries process data based on other processed data

Summary (continued)

- Most sub-queries are executed in serial fashion
- SQL functions are used to extract or transform data