Database Systems
Lecture03 – Introduction to SQL

Beomseok Nam (남범석)
bnam@skku.edu

MariaDB (MySQL) Login

- mysql –u username –p [password] [database_name]
  - Your MariaDB accounts have been created in in-ui-ye-ji cluster.
  - Your database names are db${your_student_ID}
- Example

```
% mysql –p db20171234
Enter password: changethis
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 59
Server version: 10.0.36-MariaDB-0ubuntu0.16.04.1 Ubuntu 16.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

  - Your initial password is “changethis”.
  - Change your password after logging in
    - MariaDB [(none)]> SET PASSWORD FOR ‘20171202'@'localhost’ = PASSWORD(‘new_passwd_for_20171202');

Using a MySQL Database

- To get started on your own database, first check which databases currently exist.
- Use the SHOW statement to find out which databases currently exist on the server:

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| ……                 |
| db20171234         |
| information_schema |
| mysql              |
| performance_schema |
| slurm_acct_db      |
+--------------------+
56 rows in set (0.01 sec)
```

Using a Database

- To create a new database, DB admin runs “create database” command:
  - **`MariaDB [(none)]> create database mydb;`**
- To select a database, users run “use” command:
  - **`MariaDB [(none)]> use mydb;`**
  - **`MariaDB [mydb]>`**

Creating a Table

- Once you select a database, you can access database tables in the database:

```
MariaDB [mydb]> show tables;
Empty set (0.02 sec)
```

- An empty set indicates that you have not created any table yet.

Loading Sample Data

- You can create a text file `pet.txt' containing one record per line.
- Values must be separated by tabs, and given in the order in which the columns were listed in the CREATE TABLE statement.
- Then load the data via the LOAD DATA Command.

Sample Data File

Fluffy	Harold	cat	f	1993-02-04	\N
Claws	Gwen	cat	m	1994-03-17	\N
Buffy	Harold	dog	f	1989-05-13	\N
Fang	Benny	dog	m	1990-08-27	\N
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	\N
Whistler	Gwen	bird	\N	1997-12-09	\N
Slim	Benny	snake	m	1996-04-29	\N

To Load pet.txt:

MariaDB [mydb]> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;

Run SQL source code

- You can create an SQL source file `test.sql' containing SQL statements using a text editor and run it in MariaDB shell.

To Run test.sql:

MariaDB [mydb]> source test.sql;

University Database Example

- in ~swe3003/sample_db

To Create and populate University Database:

```
MariaDB [mydb]>  source DDL.sql;
MariaDB [mydb]>  source data.sql;
```

- Once you run the SQL files, run ‘pager less’ to see the outputs page by page.

To Apply Pagination:

```
MariaDB [mydb]>  pager less;

MariaDB [mydb]> select * from instructor;
```

Chapter 3: Introduction to SQL (Cont'd)

The Rename Operation

- The SQL allows renaming relations and attributes using the **as** clause:

  old-name **as** *new-name*

- E.g.
  - **select** *ID, name, salary/12* **as** *monthly_salary*
    from *instructor*

- Keyword **as** is optional and may be omitted
  instructor **as** *T* ≡ *instructor T*
  - Keyword **as** must be omitted in Oracle DBMS

String Operations

- SQL provides a string-matching operator for comparisons on character strings.
  - percent (%). The % character matches any substring.
  - underscore (_). The _ character matches any single character.
- The operator "**like**" uses patterns that are described using special characters.

- Find the names of all instructors whose name includes the substring "dar".

```
select name
from instructor
where name like '%dar%'
```

- Match the string "100 %"

```
like '100 \%'  escape  '\'
```

String Operations (Cont.)

- Patterns are case sensitive.
- Pattern matching examples:
  - 'Intro%' matches any string beginning with "Intro".
  - '%Comp%' matches any string containing "Comp" as a substring.
  - '_ _ _' matches any string of exactly three characters.
  - '_ _ _ %' matches any string of at least three characters.
- SQL supports a variety of string operations such as
  - concatenation (using "||")
  - converting from upper to lower case (and vice versa)
    - UPPER(), LOWER()
  - finding string length, extracting substrings, etc.
    - LENGTH(), SUBSTRING(str, position, length)

Ordering the Display of Tuples

- List in alphabetic order the names of all instructors

  select distinct *name*
  from *instructor*
  order by *name*

- We may specify **desc** for descending order or **asc** for ascending order, for each attribute; ascending order is the default.
  - Example: **order by** *name* **desc**
- Can sort on multiple attributes (lexicographic sorting)
  - Example: **order by** *dept_name, name*

Where Clause Predicates

- SQL includes a **between** comparison operator
- Example: Find the names of all instructors with salary between $90,000 and $100,000 (that is, ≥ $90,000 and ≤ $100,000)
  - **select** *name*
    from *instructor*
    where *salary* **between** 90000 **and** 100000
- Tuple comparison
  - **select** *name*, *course_id*
    from *instructor*, *teaches*
    where (*instructor.ID*, *dept_name*) = (*teaches.ID*, 'Biology');

Set Operations

- Find courses that were offered in Fall 2009 or in Spring 2010

(**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2009)
union
(**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2010)

- Find courses that were offered in Fall 2009 and in Spring 2010

(**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2009)
intersect
(**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2010)

- Find courses that were offerd in Fall 2009 but not in Spring 2010

(**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2009)
except
(**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2010)

Set Operations

- Set operations **union**, **intersect**, and **except**
  - Each of the above operations automatically eliminates duplicates

- To retain all duplicates use the corresponding multiset operations **union all**, **intersect all** and **except all**.

Null Values

- It is possible for tuples to have a null value, denoted by *null*, for some of their attributes
- *null* signifies an unknown value or that a value does not exist.
- The result of any arithmetic expression involving *null* is *null*
  - Example: 5 + *null* returns null
- The predicate **is null** can be used to check for null values.
  - Example: Find all instructors whose salary is null.

select *name*
from *instructor*
where *salary* **is null**

Null Values and Three Valued Logic

- Any comparison with *null* returns *unknown*
  - Example*: 5 < null or null <> null or null = null*
- Three-valued logic using the truth value *unknown*:
  - OR: (*unknown* **or** *true*) = *true*,
    (*unknown* **or** *false*) = *unknown*
    (*unknown* **or** *unknown) = unknown*
  - AND: *(true* **and** *unknown) = unknown,*
    (false **and** *unknown) = false,*
    (unknown **and** *unknown) = unknown*
  - NOT*: (***not** *unknown) = unknown*
  - "*P* **is unknown**" evaluates to true if predicate *P* evaluates to *unknown*
- Result of **where** clause predicate is treated as *false* if it evaluates to *unknown*

Aggregate Functions

- These functions operate on the multiset of values of a column of a relation, and return a value

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values

Aggregate Functions (Cont.)

- Find the average salary of instructors in the Computer Science department
  - **select avg** (*salary*)
    from *instructor*
    where *dept_name*= 'Comp. Sci.';
- Find the total number of instructors who teach a course in the Spring 2014 semester
  - **select count** (**distinct** *ID*)
    from *teaches*
    where *semester* = 'Spring' **and** *year* = 2014
- Find the number of tuples in the *course* relation
  - **select count** (*)
    from *course*;

Aggregate Functions – Group By

- Find the average salary of instructors in each department
  - **select** *dept_name*, **avg** (*salary*)
    from *instructor*
    group by *dept_name*;
  - Note: departments with no instructor will not appear in result

ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

Aggregation (Cont.)

- Attributes in **select** clause outside of aggregate functions must appear in **group by** list
  - /* erroneous query */
    select *dept_name*, *ID*, **avg** (*salary*)
    from *instructor*
    group by *dept_name*;

Aggregate Functions – Having Clause

- Find the names and average salaries of all departments whose average salary is greater than 42000

```
select dept_name, avg (salary)
from instructor
group by dept_name
having avg (salary) > 42000;
```

Note: predicates in the **having** clause are applied after the formation of groups whereas predicates in the **where** clause are applied before forming groups

Null Values and Aggregates

- Total all salaries

  select sum (*salary*)
  from *instructor*

  - Above statement ignores null amounts
  - Result is *null* if there is no non-null amount

- All aggregate operations except **count(*)** ignore tuples with null values on the aggregated attributes