Course Chapter Map

1. INTRODUCTION
2. MySQL CLIENT/SERVER MODEL
3. DATABASE BASICS
4. DATABASE DESIGN
5. DATABASE CREATION
6. BASIC QUERIES
7. DATABASE MAINTENANCE
8. DATA MANIPULATION
9. FUNCTIONS
10. JOINING TABLES
11. EXPORTING AND IMPORTING DATA
12. SUBQUERIES
13. SUPPLEMENTAL INFORMATION
14. CONCLUSION

Learning Objectives

- Delete and modify table data
- Use the **INSERT** and **REPLACE** statements to add new data to a table
- Use the **UPDATE** statement to modify data in a table
- Use the **DELETE** statement to remove table rows

Functions in MySQL Expressions

- Several categories of functions
- Invoked within an expression
- General function syntax

```
function_name(<arg1> [, <arg2>, …, <argn>])
```

- Examples

```
SELECT NOW()

SELECT VERSION()
```

Function Tips

- **Functions can be used anywhere a value expression is accepted**
- **Columns can be used as arguments with the correct data type**
- **A function output can be the input of another function**
- **An expression with NULL always produces a NULL**
- **Mathematical functions will return NULL on error (i.e., division by zero)**

String Functions

- Perform operations on strings
- Two categories
  - Returning numbers
  - Returning strings

Returning Numbers

- Examples

```
mysql> SELECT CHAR_LENGTH('MySQL');
+----------------------+
| CHAR_LENGTH('MySQL') |
+----------------------+
|                    5 |
+----------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT INSTR('MySQL', 'SQL');
+----------------------+
| INSTR('MySQL', 'SQL') |
+----------------------+
|                    3 |
+----------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT STRCMP('abc','def'),
    -> STRCMP('def','def'),
    -> STRCMP('def','abc');
+---------------------+---------------------+---------------------+
| STRCMP('abc','def') | STRCMP('def','def') | STRCMP('def','abc') |
+---------------------+---------------------+---------------------+
|                  -1 |                   0 |                   1 |
+---------------------+---------------------+---------------------+
1 row in set (0.00 sec)
```

Returning Strings (1/5)

- Examples

```
mysql> SELECT CONCAT('A', '-', 'Z');
+----------------------+
| CONCAT('A', '-', 'Z') |
+----------------------+
| A-Z                  |
+----------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT RIGHT('MySQL', 3);
+------------------+
| RIGHT('MySQL', 3) |
+------------------+
| SQL              |
+------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT REVERSE('MySQL');
+-----------------+
| REVERSE('MySQL') |
+-----------------+
| LQSyM           |
+-----------------+
1 row in set (0.01 sec)
```

```
mysql> SELECT LEFT('MySQL', 3);
+-----------------+
| LEFT ('MySQL', 3) |
+-----------------+
| MyS             |
+-----------------+
1 row in set (0.00 sec)
```

Returning Strings (2/5)

- Examples *(continued)*

```
mysql> SELECT LOWER('MySQL');
+----------------+
| LOWER('MySQL') |
+----------------+
| mysql          |
+----------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT UPPER('MySQL');
+----------------+
| UPPER('MySQL') |
+----------------+
| MYSQL          |
+----------------+
1 row in set (0.01 sec)
```

```
mysql> SELECT LPAD('MySQL', 15, '.');
+-----------------------+
| LPAD('MySQL', 15, '.') |
+-----------------------+
| .........MySQL        |
+-----------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT RPAD('MySQL', 14,
    '_.');
+-----------------------+
| RPAD('MySQL', 14, '_.') |
+-----------------------+
| MySQL_._._._._        |
+-----------------------+
1 row in set (0.00 sec)
```

Returning Strings (3/5)

- Examples *(continued)*

```
mysql> SELECT TRIM('   MySQL    ') AS str;
+-------+
| str   |
+-------+
| MySQL |
+-------+
1 row in set (0.00 sec)


mysql> SELECT TRIM(LEADING 'Q' FROM 'QQQMySQLQQQ');
+---------------------------------------+
| TRIM(LEADING 'Q' FROM 'QQQMySQLQQQ') |
+---------------------------------------+
| MySQLQQQ                              |
+---------------------------------------+
1 row in set (0.00 sec)
```

Returning Strings (4/5)

- Examples *(continued)*

```
mysql> SELECT SUBSTRING('MySQL', 3);
+-----------------------+
| SUBSTRING('MySQL', 3) |
+-----------------------+
| SQL                   |
+-----------------------+
1 row in set (0.00 sec)


mysql> SELECT SUBSTRING('MySQL', 2, 2);
+--------------------------+
| SUBSTRING('MySQL', 2, 2) |
+--------------------------+
| yS                       |
+--------------------------+
1 row in set (0.00 sec)
```

Returning Strings (5/5)

- Examples *(continued)*

```
mysql> SELECT SUBSTRING_INDEX('training@mysql.com', '@', 1);
+-----------------------------------------------+
| SUBSTRING_INDEX('training@mysql.com', '@', 1) |
+-----------------------------------------------+
| training                                      |
+-----------------------------------------------+
1 row in set (0.00 sec)


mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', -2);
+-------------------------------------------+
| SUBSTRING_INDEX('www.mysql.com', '.', -2) |
+-------------------------------------------+
| mysql.com                                 |
+-------------------------------------------+
1 row in set (0.00 sec)
```

Temporal Functions

- Time and date
- Several forms of description
- Generated in many ways
  - Copying existing
  - Execute built-in function
  - Build a string representation
- Date components

Type:	Default Format:
DATE	YYYY-MM-DD
TIME	HH:MM:SS
DATETIME	YYYY-MM-DD HH:MI:SS
TIMESTAMP	YYYY-MM-DD HH:MI:SS
YEAR	YYYY

Temporal Function Types

- NOW()
- CURDATE()
- CURTIME()
- YEAR(<*date_expression*>)
- MONTH(<*date_expression*>)
- DAYOFMONTH(<*date_expression*>), DAY(<*date_expression*>)
- DAYNAME(<*date_expression*>)
- HOUR(<*time_expression*>)
- MINUTE(<*time_expression*>)
- SECOND(<*time_expression*>)

Extracting Temporal Data (1/2)

- Examples

```
mysql> SELECT CURDATE(), CURTIME(), DAYNAME(NOW());
+------------+-----------+----------------+
| CURDATE()  | CURTIME() | DAYNAME(NOW()) |
+------------+-----------+----------------+
| 2006-06-09 | 17:55:40  | Friday         |
+------------+-----------+----------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT NOW(), NOW() + INTERVAL 5 DAY;
+---------------------+-----------------------+
| NOW()               | NOW() + INTERVAL 5 DAY |
+---------------------+-----------------------+
| 2006-06-09 18:02:35 | 2006-06-14 18:02:35    |
+---------------------+-----------------------+
1 row in set (0.00 sec)
```

Extracting Temporal Data (2/2)

- Examples

```
mysql> SELECT NOW(), DATE_FORMAT(NOW(), '%W the %D of %M');
+---------------------+----------------------------------------+
| NOW()               | DATE_FORMAT(NOW(), '%W the %D of %M') |
+---------------------+----------------------------------------+
| 2004-04-30 11:59:15 | Friday the 30th of April               |
+---------------------+----------------------------------------+
1 row in set (0.00 sec)
```

Numeric Functions (1/2)

- Mathematical operations
- Examples

```
mysql> SELECT ABS(-42), ABS(42);
+----------+---------+
| ABS(-42) | ABS(42) |
+----------+---------+
|       42 |      42 |
+----------+---------+
1 row in set (0.00 sec)
```

```
mysql> SELECT SIGN(-42), SIGN(-1), SIGN(0), SIGN(1), SIGN(42);
+-----------+----------+---------+---------+----------+
| SIGN(-42) | SIGN(-1) | SIGN(0) | SIGN(1) | SIGN(42) |
+-----------+----------+---------+---------+----------+
|        -1 |       -1 |       0 |       1 |        1 |
+-----------+----------+---------+---------+----------+
1 row in set (0.06 sec)
```

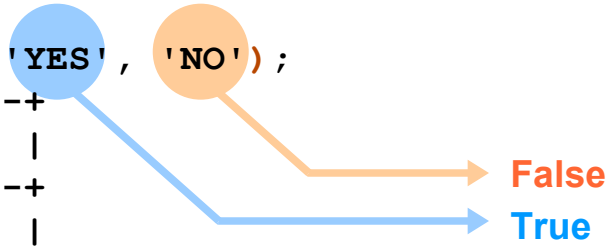
Numeric Functions (2/2)

- More numeric functions
  - **TRUNCATE(<*number*>, <*decimals*>)**
  - **FLOOR(<*number*>)**
  - **CEILING(<*number*>)**
  - **ROUND(<*number*>)**
- Mathematical functions
  - Geometry
  - Trigonometry
  - Other

Flow Control Functions (1/3)

- Choose between different values based on the result of an expression

```
mysql> SELECT IF(1 > 0, 'YES', 'NO');
+-------------------------+
| IF(1 > 0, 'YES', 'NO') |
+-------------------------+
| YES                     |
+-------------------------+
1 row in set (0.03 sec)
```



Flow Control Functions (2/3)

- **CASE/WHEN** provides branching flow control
- General syntax

```
CASE value
    WHEN <compare_value> THEN <result>
    [WHEN <compare_value> THEN <result> ...]
    [ELSE <result>]
END
```

...and...

```
CASE
    WHEN <condition> THEN <result>
    [WHEN <condition> THEN <result> ...]
    [ELSE <result>]
END
```

Flow Control Functions (3/3)

- Examples

```
mysql> SELECT CASE 3 WHEN 1 THEN 'one'
    -> WHEN 2 THEN 'two' ELSE 'more' END;
    'more'

mysql> SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false' END;
    'true'

mysql> SELECT CASE BINARY 'B'
    -> WHEN 'a' THEN 1 WHEN 'b' THEN 2 END;
    NULL

SELECT pet_name, owner, pet_gender,
CASE
    WHEN pet_gender = 'm' THEN 'Boy'
    WHEN pet_gender = 'f' THEN 'Girl'
    ELSE 'Unknown'
END
FROM pet_info
ORDER BY pet_gender;
```

Why Use Aggregate Functions? (1/2)

- Summary functions
  - Perform summary operations on a set of values
- Returns Single Value Based on Group of Values
- Only **NON NULL**

Why Use Aggregate Functions? (2/2)

- Aggregate function types
  - **MIN()**
  - **MAX()**
  - **SUM()**
  - **AVG()**
  - **COUNT()**
  - **GROUP_CONCAT()**
- Examples

```
mysql> SELECT COUNT(*)
    -> FROM Country;
+----------+
| COUNT(*) |
+----------+
|      239 |
+----------+
1 row in set (0.00 sec)
```

AND

```
mysql> SELECT COUNT(Capital)
    -> FROM Country;
+----------------+
| COUNT(Capital) |
+----------------+
|            232 |
+----------------+
1 row in set (0.00 sec)
```

Grouping with SELECT...GROUP BY (1/2)

- Use **GROUP BY** for sub-group
- Based on values on one or more columns of rows
- Example

```
mysql> SELECT Continent, AVG(Population)
    -> FROM Country
    -> GROUP BY Continent;
+---------------+------------------+
| Continent     | AVG(Population)  |
+---------------+------------------+
| Asia          |   72647562.7451  |
| Europe        |   15871186.9565  |
| North America |   13053864.8649  |
| Africa        |   13525431.0345  |
| Oceania       |    1085755.3571  |
| Antarctica    |          0.0000  |
| South America |   24698571.4286  |
+---------------+------------------+
7 rows in set (0.00 sec)
```

Grouping with SELECT...GROUP BY (2/2)

- Use **GROUP_CONCAT()** to concatenate results
- Example

```
mysql> SELECT GovernmentForm, GROUP_CONCAT(Name) AS Countries
    -> FROM Country WHERE Continent = 'South America'
    -> GROUP BY GovernmentForm\G
*************************** 1. row ***************************
GovernmentForm: Dependent Territory of the UK
      Countries: Falkland Islands
*************************** 2. row ***************************
GovernmentForm: Federal Republic
      Countries: Argentina,Venezuela,Brazil
*************************** 3. row ***************************
GovernmentForm: Overseas Department of France
      Countries: French Guiana
*************************** 4. row ***************************
GovernmentForm: Republic
Countries:Chile,Uruguay,Suriname,Peru,Paraguay,Bolivia,Guyana,
Ecuador,Colombia
4 rows in set (0.00 sec)
```

GROUP BY...HAVING

- Eliminates rows based on aggregate values
- Only for comparisons against aggregated values
- Example

```
mysql> SELECT Continent, SUM(Population) AS pop
    -> FROM Country
    -> GROUP BY Continent
    -> HAVING SUM(Population) > 100000000;
+---------------+------------+
| Continent     | pop        |
+---------------+------------+
| Asia          | 3705025700 |
| Europe        |  730074600 |
| North America |  482993000 |
| Africa        |  784475000 |
| South America |  345780000 |
+---------------+------------+
5 rows in set (0.00 sec)
```

GROUP BY...WITH ROLLUP (1/2)

- Multiple levels of summary values
- Example

```
mysql> SELECT Continent, SUM(Population) AS pop
    -> FROM Country
    -> GROUP BY Continent WITH ROLLUP;
+---------------+------------+
| Continent     | pop        |
+---------------+------------+
| Asia          | 3705025700 |
| Europe        |  730074600 |
| North America |  482993000 |
| Africa        |  784475000 |
| Oceania       |   30401150 |
| Antarctica    |          0 |
| South America |  345780000 |
| NULL          | 6078749450 |
+---------------+------------+
8 rows in set (0.53 sec)
```

GROUP BY...WITH ROLLUP (2/2)

- Super aggregate operation
- Example

```
mysql> SELECT Continent, AVG(Population) AS avg_pop
    -> FROM Country
    -> GROUP BY Continent WITH ROLLUP;
+---------------+---------------+
| Continent     | avg_pop       |
+---------------+---------------+
| Asia          | 72647562.7451 |
| Europe        | 15871186.9565 |
| North America | 13053864.8649 |
| Africa        | 13525431.0345 |
| Oceania       |  1085755.3571 |
| Antarctica    |        0.0000 |
| South America | 24698571.4286 |
| NULL          | 25434098.1172 |
+---------------+---------------+
8 rows in set (0.06 sec)
```

IGNORE SPACE

- Example

```
mysql> SELECT PI ();
ERROR 1305 (42000): FUNCTION world.PI does not exist

mysql> SET sql_mode = 'IGNORE_SPACE';
Query OK, 0 rows affected (0.06 sec)

mysql> SELECT PI ();
+----------+
| PI()     |
+----------+
| 3.141593 |
+----------+
1 row in set (0.05 sec)
```

Further Practice: Chapter 9

- Comprehensive exercises

Chapter Summary

- Delete and modify table data
- Use the **INSERT** and **REPLACE** statements to add new data to a table
- Use the **UPDATE** statement to modify data in a table
- Use the **DELETE** statement to remove table rows