ITCS393 Database Systems Lab

Data Type and Function

Dr. Petch Sajjacholapunt
Dr. Wudhichart Sawangphol
Dr. Jidapa Kraisangka

petch.saj@mahidol.ac.th

Learning Outcomes

After this class, students should be able to

- explain about different data types that MySQL uses to store data.
- Useful function to manage Date and Time.
- Use CAST and CONVERT functions in MySQL to change between the data types,

Data Types

- **Lesson 1**: Categories of Data Types
  - **Numeric:** are used for storing numbers.
  - **String:** are used for storing text data.
  - **Date and time:** are used for storing dates and times.
  - **Miscellaneous:** other then above e.g., flag data
- **Lesson 2**: Cast and Conversion

Definition of Data Types

- **A data type** is a classification that determines the type of value a variable or column can hold.
- Data types help ensure data integrity and improve query performance.
- In databases, data types are essential for accurately storing and processing data.

Example of Creating a Table in MySQL

```
CREATE TABLE inventory (
    item_id             INT NOT NULL,
    item_name           VARCHAR(50) NOT NULL,
    quantity            INT NOT NULL,
    last_restock_date   DATE,
    last_restock_time   TIME,
    in_stock            BOOLEAN,
    PRIMARY KEY (item_id)
);
```

Table: inventory

item_id	item_name	quantity	last_restock_date	last_restock_time	in_stock
1001	Speaker	50	2022-02-15	14:30:00	true
1002	Mobile Phones	25	2022-02-20	10:45:00	false
1003	Laptop	75	2022-02-25	09:15:00	true
1004	Pen	100	2022-03-01	16:00:00	true

Lesson 1: Categories of Data Types

- MySQL has a wide range of data types.
  - **Numeric:** are used for storing numbers.
  - **String:** are used for storing text data.
  - **Date and time:** are used for storing dates and times.
  - **Miscellaneous:** other then above e.g., flag data

Lesson 1: Categories of Data Type – (1)

Numeric	String	Date and Time	Miscellaneous data types
INTEGER/INT(size)	CHAR(size)	DATE	BOOLEAN/BOOL
TINYINT(size)	VARCHAR(size)	DATETIME	ENUM
SMALLINT(size)	TINYTEXT	TIMESTAMP	SET
MEDIUMINT(size)	TEXT	TIME	
BIGINT(size)	MEDIUMTEXT	YEAR	
FLOAT(size,d)	LONGTEXT		
DOUBLE(size,d)			
DECIMAL(size,d)/DEC(size,d)			

Numeric Data Types

Numeric

Data Type	Range	Storage
TINYINT	Signed: -128 to 127 Unsigned: 0 to 255	1 byte
SMALLINT	Signed: -32768 to 32767 Unsigned: 0 to 65535	2 bytes
MEDIUMINT	Signed: -8388608 to 8388607 Unsigned: 0 to 16777215	3 bytes
INT	Signed: -2147483648 to 2147483647 Unsigned: 0 to 4294967295	4 bytes
BIGINT	Signed: -9223372036854775808 to 9223372036854775807 Unsigned: 0 to 18446744073709551615	8 bytes

Questions

- If you want to store weight of a person, which data type would be sufficient?

  TinyInt

- How about keeping distance value in km between earth to other planets?
  (*note* distance between earth and Neptune is 4.6218 billion km)

  BigInt

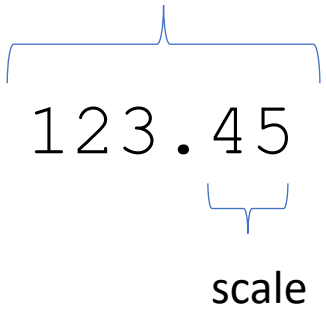
- What will happen if you try to keep value outside its range?
  (e.g., `SELECT 9223372036854775807 + 1;`)

  Arithmetic overflow error

Numeric

1.2 Exact Numeric (fixed precision): decimal, numeric

- `decimal(p[,s])]` and `numeric[(p[,s])]`
  - decimal and numeric are *functionlly equivalent*
- `p` (*precision*): maximum total number of digits to be stored up to
  - Range from 1 to 38, the default value is 18
- `s` (*scale*): the number of decimal digits stored to the right of the decimal point.
  - Range from 0 through *precision*, the default value is 0



Precision	Storage bytes
1 - 9	5
10 - 19	9
20 - 28	13
29 - 38	17

Numeric

Data Type	Range	Storage
FLOAT	-3.402823466E+38 to -1.175494351E-38 0 1.175494351E-38 to 3.402823466E+38	4 bytes
DOUBLE	-1.7976931348623157E+308 to -2.225074e-308 0 2.225074e-308 to 1.7976931348623157E+308	8 bytes
DECIMAL	-10^38 + 1 to 10^38 - 1	Variable

- The FLOAT and DOUBLE data types are used to store approximate numeric values, while the DECIMAL data type is used to **store exact numeric values**.

String Data Types

String

Data Type	Range	Storage
CHAR(n)	**Fixed-length** string with a maximum length of n characters	n bytes
VARCHAR(n)	**Variable-length** string with a maximum length of n characters	1 or 2 byte + length of the string (up to a maximum of n bytes)
TINYTEXT	**Variable-length** string with a maximum length of 255 characters	1 byte + length of the string (up to a maximum of 255 bytes)
TEXT	**Variable-length** string with a maximum length of 65,535 characters	2 bytes + length of the string (up to a maximum of 65,533 bytes)
MEDIUMTEXT	**Variable-length** string with a maximum length of 16,777,215 characters	3 bytes + length of the string (up to a maximum of 16,777,212 bytes)
LONGTEXT	**Variable-length** string with a maximum length of 4,294,967,295 characters	4 bytes + length of the string (up to a maximum of 4,294,967,291 bytes)

String

- **Fixed-Length String** or *CHAR(n)*
- It *will always require exactly n bytes of storage.*
  - If the string value is shorter than n characters, the remaining bytes will be filled with blank spaces.
  - If the string value is longer than n characters, mysql will not allow to insert data. (unless the restrict mode is disabled)

String

- Example of using CHAR(n)

Try: Create the following table with char(5) column and try insert with different string values:

```
CREATE TABLE example_table (
    id INT PRIMARY KEY,
    char_column CHAR(5)
);
```

What will happened after insert the following value ?

```
INSERT INTO example_table (id, char_column) VALUES (1, 'Hello');
INSERT INTO example_table (id, char_column) VALUES (2, 'World');
INSERT INTO example_table (id, char_column) VALUES (3, 'This is longer than 5 character');
```

String

- **Variable-Length String** e.g., VARCHAR(n), TEXT.
- It will use a storage as necessary to store the actual string value PLUS 1 byte (overhead) storing an actual size value of string.

For example:

- A VARCHAR(10) column can store a string value of up to 10 characters in length.
- If we store a shorter string value like 'Hello' in the column.
- 6 bytes of storage will be used.
- 1 byte for the length of the string and 5 bytes for the actual string value.

String

- Example of using VARCHAR(n)

Try: Create the following table with varchar(10) column and try insert with different string values:

```
CREATE TABLE example_table2 (
    id INT PRIMARY KEY,
    varchar_column VARCHAR(10)
);
```

What will happened after insert the following value ?

```
INSERT INTO example_table2 (id, varchar_column) VALUES (1, 'Hello');
INSERT INTO example_table2 (id, varchar_column) VALUES (2, 'World');
INSERT INTO example_table2 (id, varchar_column) VALUES (3, 'This is a longer string');
```

String

- Example of using TEXT

Try: Create the following table with TEXT column and try insert with different string values:

```
CREATE TABLE example_table3 (
  id INT PRIMARY KEY,
  title VARCHAR(50),
  content TEXT
);
```

How different between **VARCHAR(n)** and **TEXT ?**

```
INSERT INTO example_table3 (id, title, content) VALUES
(1, 'Example Title', 'This is an example of a very large text string that can be stored in a TEXT column.');
```

String

- Key different between VARCHAR and TEXT.
  - TEXT is forced the maximum value to be 65,535 bytes.
  - VARCHAR is based on the actual length value which is use less storage for the shorter string.
  - TEXT is good for store large String value that maximum exceed the maximum length limit of VARCHAR.
  - For larger String, MEDIUMTEXT and LONGTEXT are considered.

Date and Time Data Types

Date and Time

Data Type	Range	Storage
DATE	January 1, 1000, to December 31, 9999*	3 bytes
TIME	'-838:59:59' to '838:59:59'	3 bytes
DATETIME	January 1, 1000, to December 31, 9999,* with a precision of up to 6 decimal places for fractional seconds	8 bytes
TIMESTAMP	January 1, 1970, to December 31, 2037, * with a precision of up to 6 decimal places for fractional seconds	4 bytes

***Note that**: if you want to store dates apart from this range you can use other non-temporal formats such as INT.

Date and Time

- Example of using DATE, TIME, DATETIME, TIMESTAMP

Try: Create the following table

```
CREATE TABLE my_table (
      id INT PRIMARY KEY AUTO_INCREMENT,
      event_name VARCHAR(255),
      event_date DATE,
      event_time TIME,
      event_datetime DATETIME,
      event_timestamp TIMESTAMP
);
```

Date and Time

- Example of using DATE, TIME, DATETIME, TIMESTAMP

Try: insert with different data values:

```
INSERT INTO my_table VALUES
('My Event 1', '2023-03-20', '09:30:00', '2023-03-20 09:30:00');
INSERT INTO my_table VALUES
('My Event 2', '2023-03-21', '15:45:00', '2023-03-21 15:45:00');
INSERT INTO my_table (event_name, event_date) VALUES
('My Event 3', '2023-03-22');
INSERT INTO my_table (event_name, event_time) VALUES
('My Event 4', '20:15:00');
INSERT INTO my_table (event_name, event_datetime) VALUES
('My Event 5', '2023-03-23 12:00:00');
INSERT INTO my_table (event_name) VALUES ('My Event 6');
```

Date and Time

Working with Date and Time..

Function	Description	Example
NOW()	Returns the current date and time in MySQL server's timezone	SELECT NOW();
DATE()	Extracts the date part of a datetime expression	SELECT DATE('2023-03-19 12:34:56');
YEAR()	Extracts the year part of a datetime expression	SELECT YEAR('2023-03-19 12:34:56');
MONTH()	Extracts the month part of a datetime expression	SELECT MONTH('2023-03-19 12:34:56');
DAY()	Extracts the day part of a datetime expression	SELECT DAY('2023-03-19 12:34:56');
HOUR()	Extracts the hour part of a datetime expression	SELECT HOUR('2023-03-19 12:34:56');
MINUTE()	Extracts the minute part of a datetime expression	SELECT MINUTE('2023-03-19 12:34:56');
SECOND()	Extracts the second part of a datetime expression	SELECT SECOND('2023-03-19 12:34:56');
DATE_FORMAT()	Formats a datetime expression into a specified format	SELECT DATE_FORMAT ('2023-03-19 12:34:56', '%Y-%m-%d %H:%i:%s');

Date and Time

Working with Date and Time.. (Con’t)

Function	Description	Example
ADDDATE()	Adds a specified interval to a date or datetime expression	SELECT ADDDATE('2022-09-30', INTERVAL 1 MONTH);
DATEDIFF()	Calculates the difference between two dates	SELECT DATEDIFF('2023-01-01', ‘2022-09-30');
TIMEDIFF()	Calculates the difference between two times	SELECT TIMEDIFF('12:00:00', '09:00:00');
TIME_TO_SEC()	Converts a time expression to seconds	SELECT TIME_TO_SEC('01:30:00');
SEC_TO_TIME()	Converts a number of seconds to a time expression	SELECT SEC_TO_TIME(5400);
LAST_DAY()	Returns the date of the last day of the month for a given date	SELECT LAST_DAY(‘2022-09-15');

Miscellaneous Data Types

Lesson 1: Categories of Data Type – (1) Miscellaneous

Data Type	Range	Storage
BOOLEAN	True/False or 1/0	1 byte
ENUM	A list of up to 65,535 distinct values	1 or 2 bytes, depending on the number of distinct values
SET	A list of up to 64 distinct values	1, 2, 3, 4, or 8 bytes, depending on the number of distinct values

Miscellaneous

- Example of using Boolean

Try: Create the following table with `BOOLEAN` column and try insert data.

```
CREATE TABLE my_table (
  id INT(11)          NOT NULL AUTO_INCREMENT,
  name VARCHAR(50)    NOT NULL,
  is_active BOOLEAN NOT NULL DEFAULT FALSE,
  PRIMARY KEY (id)
);
```

```
INSERT INTO my_table (name, is_active) VALUES ('My Item', TRUE);
INSERT INTO my_table (name, is_active) VALUES ('My Item', 1);
```

Miscellaneous

- ENUM - is used to define a column that can contain a finite set of values.
- Use to store value, which are defined when the table is created.
- It is useful when you want to restrict the possible values

Example of using ENUM

```
CREATE TABLE employees (
        id INT(11) PRIMARY KEY NOT NULL AUTO_INCREMENT,
        name VARCHAR(50) NOT NULL,
        department ENUM('Sales', 'Marketing', 'Finance', 'IT') NOT NULL,
);
```

Miscellaneous

- Example of using ENUM

What will happened after insert the following value ?

```
INSERT INTO employees VALUES (1, 'John Doe', 'Marketing');
INSERT INTO employees VALUES (2, 'Jane Smith', 'Marketing');
INSERT INTO employees VALUES (3, 'Bob Johnson', 'ICT');
```

Note that when inserting data into an ENUM column, the value must be one of the predefined values. If you try to insert a value that is not in the list, MySQL will throw an error.

Miscellaneous

- SET - is used to define a column that can contain a set of values.
- Use to store zero or more values from a predefined set.

Example of using SET

```
CREATE TABLE customers (
        id INT(11)              NOT NULL AUTO_INCREMENT,
        name VARCHAR(50)        NOT NULL,
        email VARCHAR(50)       NOT NULL,
        interests SET('books', 'movies', 'music', 'sports', 'travel'),
        PRIMARY KEY (id)
);
```

Miscellaneous

- Example of using SET

```
INSERT INTO customers (name, email, interests) VALUES
  ('John Doe', 'john@example.com', 'books,movies'),
  ('Jane Smith', 'jane@example.com', 'music,sports,travel'),
  ('Bob Johnson', 'bob@example.com', 'travel'),
  ('Alice Brown', 'alice@example.com', 'sports');
```

Can we select only the customer who play sport ?

Miscellaneous

FIND_IN_SET function

- ***Syntax**: FIND_IN_SET(string, set)*
  - It will return the position of a string within a set of strings.
  - It is used to determine if a string value is present in the set of values stored in the column, if 'yes' return position (which will always greater than zero).
- Try the following statement:

```
SELECT * FROM customers WHERE FIND_IN_SET('sports', interests) > 0;
```

Miscellaneous

Useful Function

Function	Description	Example
UUID()	Generates a universally unique identifier (UUID) using the standard UUID algorithm.	UUID()
UUID_SHORT()	Generates a 64-bit UUID using a non-standard algorithm.	UUID_SHORT()
MD5(str)	Calculates the MD5 hash of the input string str.	MD5(str)
SHA1(str)	Calculates the SHA-1 hash of the input string str.	SHA1(str)
SHA2(str, len)	Calculates the SHA-2 hash of the input string str, where len specifies the length of the hash output (either 256 or 512 bits).	SHA2(str, len)
PASSWORD(str)	Calculates the hash value of the input string str using the old MySQL password hashing algorithm.	PASSWORD(str)
ENCRYPT(str[, salt])	Calculates the hash value of the input string str using the Unix crypt() system call. An optional salt argument can be provided to specify the salt to use for the hashing.	ENCRYPT(str[, salt])
MD5SUM(str)	Calculates the MD5 checksum of the input string str, which is the same as the output of the Unix md5sum command.	MD5SUM(str)

UUID is a Universally Unique Identifier. It's a 128-bit number used as a unique identifier for digital objects.

Cast and Conversion

CAST function

- Cast is to convert a value to an int datatype.
- Syntax: CAST(*expression* AS *datatype(length)*)

For example:

```
SELECT CAST(25.65 AS SIGNED INTEGER);
```

Output is 25

Note that: in mysql CAST() does not support all datatypes conversion.

- In **MYSQL** you can convert between different data types using the CAST and CONVERT functions.
  - The CAST function allows you to convert a value of one data type to another data type.
  - The CONVERT function allows you to convert a value of one data type to another data type with a specific character set.

CAST function

- **Syntax:** CAST(value AS data_type).
  - The *value* is the value that you want to convert.
  - The *data_type* the data type to which you want to convert the value.
- The data types that you can cast to depend on the original data type of the value you want to convert.
- Some examples of data types that you can cast to include INT, FLOAT, DECIMAL, CHAR, VARCHAR, DATE, TIME, and DATETIME.

- Example of CAST numeric value
  - SELECT CAST(12345 AS DECIMAL(7,2)) AS decimal_col;
  - SELECT CAST(123.45 AS SIGNED) AS signed_col;
  - SELECT CAST('123.12' AS UNSIGNED INTEGER) AS unsigned_col;
  - SELECT CAST(2.718 AS CHAR) AS char_col;
  - SELECT CAST('2023-03-19' AS CHAR) as string_col;
  - SELECT CAST('2023-03-19' AS DATE) as date_col;

CONVERT function

- **Syntax:**

  CONVERT(value, data_type *CHARACTER SET character_set*) *Optional

  CONVERT(value, data_type USING character_set)

  - The *value* is the value that you want to convert.
  - The *data_type* the data type to which you want to convert the value.
  - the character_set is the character set that you want to use for the conversion.

- Some examples of data types that you can convert to include CHAR, VARCHAR, TEXT, BINARY, VARBINARY, BLOB, DATE, TIME, and DATETIME.

CONVERT function (con't)

- In MySQL, character sets are used to define how character data is stored and manipulated.
- MySQL supports many different character sets, including ASCII, Unicode, and various national character sets.
- CONVERT thus is typically used for character data where it is important to specify the correct character set.

- Example of CONVERT character value
  - SELECT CONVERT('hello', CHAR(10) CHARACTER SET utf8) as char_value;
  - SELECT CONVERT('convert char set' USING utf8) as char_set_col;
  - SELECT CONVERT('2023-03-19', DATETIME) as datetime_col;

THANKS
FOR YOUR
ATTENTION
ANY
QUESTIONS?