Software Fundamentals Series

Workshop #2: Databases and SQL

IEEE Computer Society Ryerson Chapter

September 25th, 2018

IEEE computer society

IEEE Advancing Technology for Humanity

Data

- Data is everywhere

- Every application you use contains data related to various kinds of information

- E.g.
  - Facebook stores data such as users, user friends, user posts, user likes etc.
  - Banks stores data such as users, user money, user account etc.

- How is this data stored and associated with each other?
  - **Databases**

Databases

- Program structure that allows data be stored in a specific format

- Includes a Database Management System (DBMS) that provides the functionality to add to, modify, and query the data quickly

- **query**: instructions for retrieving data from a database

Types of Databases

- Relational
- Non-Relational (noSQL)
- Object Oriented
- Distributed
- Real Time
- And many more!

Relational Databases

- Most popular database model
- Popular relational DBMSs include: MySQL, Microsoft SQL Server, Oracle Database, PostgreSQL, Db2
- Table structure
- Uses multiple tables to relate data entries

Relational Databases

id	name
1	Davin
2	Daniel
3	Kirill

user_id	company_id
1	3
2	2
3	1

id	company
1	AMD
2	Celestica
3	IBM

SQL

- Structured Query Language

- Allows you to access databases and perform queries, create tables, edit data, etc.

- ANSI standard but there are many versions of SQL for each DBMS
  - In order to satisfy ANSI standard, all versions support certain core commands

Why Learn SQL?

- All companies have data and that is stored in databases

- Most companies use relational databases

- Applies to many fields such as:
  - Web Development
  - Cloud Computing
  - Big Data Analytics
  - Machine Learning

- Learning SQL makes you more **EMPLOYABLE!**

MySQL

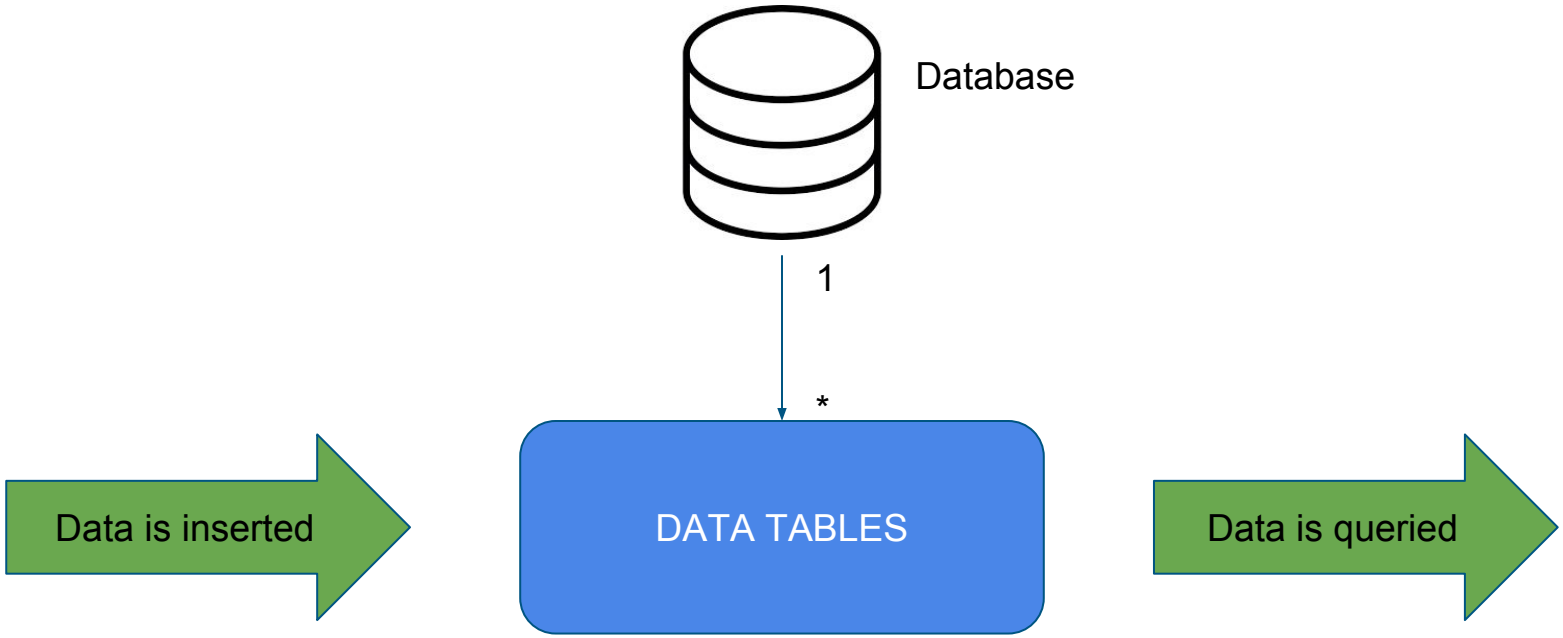
- One of the most popular DBMS
- Open Source
- Multi-platform
- Used in many websites including Google, Facebook, Youtube

SQL Tutorial

MySQL General Syntax

- Case insensitive
  - However, it is good practise to write reserved words in all UPPERCASE
    - i.e.) SELECT * FROM table;
- Whitespace insensitive
- Semicolon terminated

Basic Database Structure



CREATE DATABASE

- The database is where TABLES are created and organized

- The CREATE DATABASE command is used to created a new SQL database

```
CREATE DATABASE databasename;
```

- NOTE: For the workshop, SQLFiddle will act as our database
  - Thus, the CREATE DATABASE statement is not required in this case

CREATE TABLE

- The CREATE TABLE statement creates a table within the database

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
    ....
);
```

Datatypes (Text)

Data type	Description
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. **Note:** The values are sorted in the order you enter them. You enter the possible values in this format: ENUM('X','Y','Z')
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

Datatypes (Numeric)

Data type	Description
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

Datatypes (Date)

Data type	Description
DATE()	A date. Format: YYYY-MM-DD **Note:** The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MI:SS **Note:** The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. **Note:** Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

CREATE Constraints

- The CREATE statement may also be used with creates to implement special rules for the data in the table

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

Common Constraints

- Common constraints include
  - **NOT NULL**: Ensures that a column cannot have a NULL value
  - **UNIQUE**: Ensures that all values in a column are different
  - **PRIMARY KEY**: A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
  - **FOREIGN KEY**: Uniquely identifies a row/record in another table
  - **CHECK:** Ensures that all values in a column satisfies a specific condition
  - **DEFAULT**: Sets a default value for a column when no value is specified
  - **INDEX:** Used to create and retrieve data from the database very quickly

CREATE TABLE Example

```
CREATE TABLE students
(
 student_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
 first_name VARCHAR(30) NOT NULL,
 last_name VARCHAR(30) NOT NULL,
 email VARCHAR(60) NULL,
 province CHAR(2) NOT NULL DEFAULT "ON",
 birth_date DATE NOT NULL,
 sex ENUM('M', 'F', 'O') NOT NULL
);
```

SQL INSERT Statement

- The INSERT statement is used to insert new data into a TABLE

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

- The order of the column names are stated first, followed by the data value that correspond to each column

- i.e.)

```
INSERT INTO students (student_id, first_name, last_name, email,
                      province, birth_date, sex)
VALUES (1, "John", "Smith", "jsmith@example.com",
        "ON", "1990-01-21", "M");
```

SELECT statement

- The SELECT statement is used to select and display data from a database

```
SELECT column1, column2, ...
FROM table_name;
```

- Select all columns, use the * operator:
  - * = 'ALL'

```
SELECT * FROM table_name;
```

WHERE statement

- Used to filter records

- Extracts records that fulfill specific condition

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- Note: conditions are specified on columns

WHERE conditions

Operator	Description
=	Equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

WHERE - AND, OR, NOT

- A combination of the logical operators (AND, OR, NOT) can be used in the WHERE statement

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

UPDATE statement

- Used to modify existing records in the table

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

DELETE statement

- Used to delete existing records from the database

```
DELETE FROM table_name
WHERE condition;
```

ALTER TABLE

- The ALTER statement allows users to ADD, DELETE (DROP), MODIFY columns

- ADD column Statement
  - ```
    ALTER TABLE table_name
    ADD column_name datatype;
    ```
  - i.e.) `ALTER TABLE students ADD middle_name VARCHAR(255);`
    ```

ALTER TABLE (con't)

- The DROP statement allows you to remove a column from the TABLE
- DROP column statement
  - ```
    ALTER TABLE table_name
    DROP COLUMN column_name;
    ```
  - i.e.) `ALTER TABLE students DROP province;`
- MODIFY column statement
  - ```
    ALTER TABLE table_name
    MODIFY COLUMN column_name datatype;
    ```
  - i.e.) `ALTER TABLE students MODIFY COLUMN birth_date INT UNSIGNED;`

DROP TABLE

- The DROP TABLE statement removes the TABLE from the DATABASE you are currently in
- The DROP statement
  - ```
    DROP TABLE table_name;
    ```
  - i.e.)
    ```
    DROP TABLE students;
    ```
    ```