

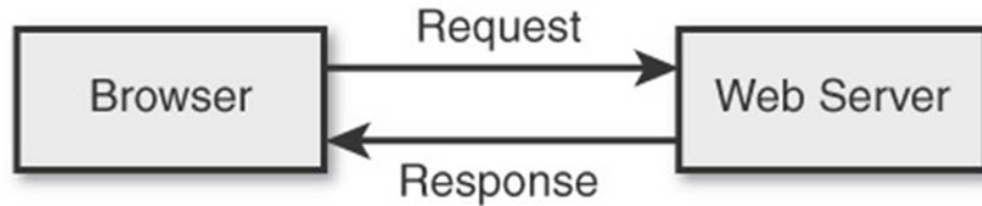
CSIS 3280: Lecture 8

Agenda

- Web Database Architecture
- Relational Database Concepts
- Database users and privileges
- Creating database
- Data types and creating tables in MySQL
- Using script to automate database and table creation
- DML: Select, Insert, Update, Delete
- Joining tables
- PHP mysqli

Web Database Architecture

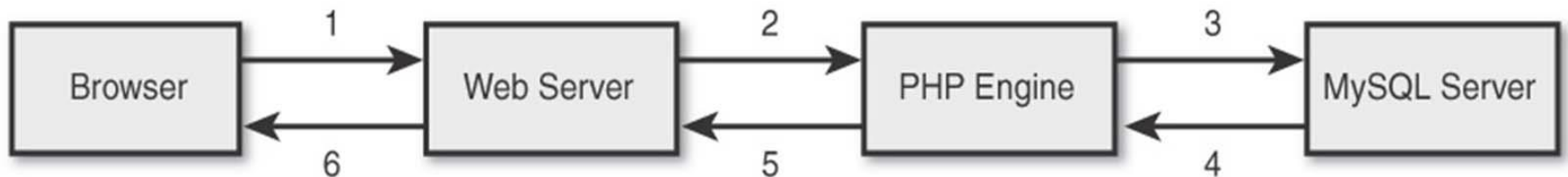
- Previously we have worked on the following:



- We wrote code that ran on our web server (WAMP) and we accessed it using our web browser.

Web Database Architecture

- Adding a database to our repertoire will allow us to write PHP applications that have the ability to easily store and retrieve data
 - Provide faster access to data than flat files
 - Easily queried to extract data
 - Built in multi-user functionality
 - Built-in privileges
 - Enable observation of separation between data storage and application



Relational Databases

- The most commonly used type of database. They depend on the theoretical basis for relational algebra.
- We are not covering relational theory so don't worry too much about it, we need to cover the concepts of relational databases so we can use them with PHP

Relational Database Concepts

- **Tables** – sometimes called tables these are used to store data.
- **Columns** – Each column in a table has a unique name and a data type.
- **Rows** – Rows in a table represent an item that is stored in a table
- **Values** – are the data stored in columns, each value must have a corresponding column and data type.
- **Keys** – Uniquely identify rows in a table, and any corresponding rows in other tables
- **Schemas** – Like the blueprint of a database, this is the complete set of table designs
- **Relationships** – represented as foreign keys these are design implicit and dictate how tables are related to each other.
- **Referential Integrity** – maintain a stable relationship between one table and another

Tables

- If you've used Excel, you already know what these are!
- A single table should represent a single object in the real world, these should match your Objects for example: Car, Person, Team, Player
- Tables have a name and a number of columns

Columns

- Columns represent the fields of a table
- These should match the 'attributes' of your object for example: haircolor, hasGlasses, email, position etc.
- Each column must be assigned a data type.
 - The datatype tells the computer what kind of data to store in the column and how it will be formatted and queried.
- There are many different types of data that the database can store pick one that matches what you are storing
 - for example date of birth would use the date datatype, email might use the text data type and so on.

Rows

- If tables are like the representation of a class (if you have mapped your data after your objects) then
 - a row would be an object, or an instantiated class
 - Each object would be represented in a row. Much like each object could be represented in the line of a CSV file.
- Recall our demo code in the previous lecture!
 - Order object is the table object consisting of row objects

Regular object				
Special object				
Special object				

- Member of Order object: Array of Regular and Special objects.
- Each of this row object has its own attribute or column name

Regular object	type	name	oils	tires
Special object	type	name	oils	tires
Special object	type	name	oils	tires

Values

- Values is the plural name for the data that is stored in the column
- Because the column specifies the data type and constraints, the values in the column must conform to it.

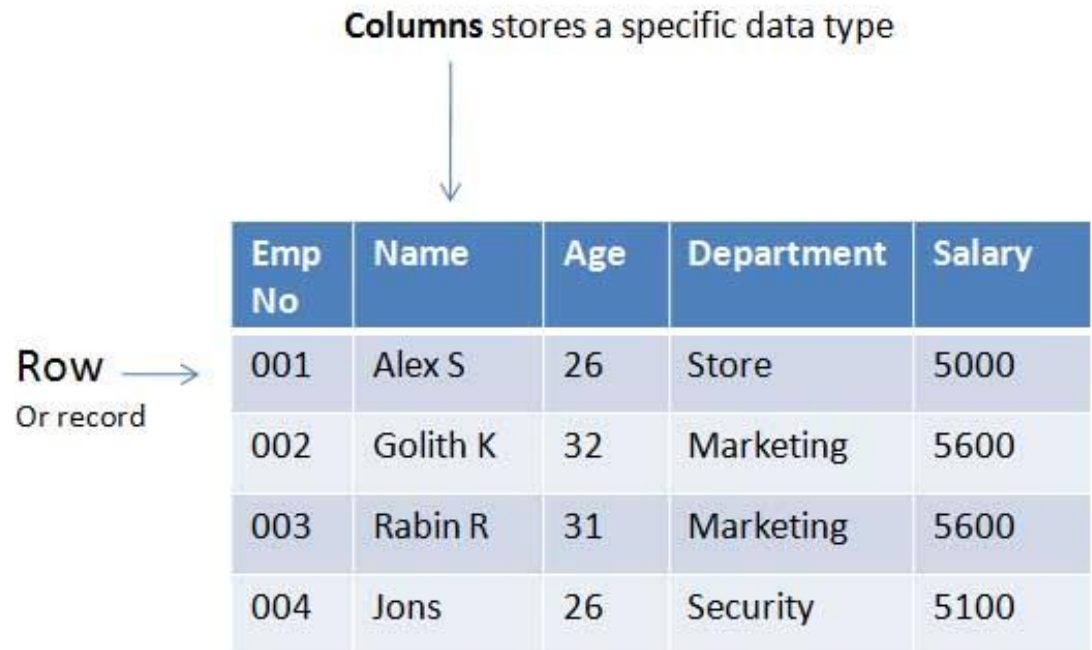
Keys

- Each row of data must be identifiable
- Keys are used to state and enforce these rules
- Two types of keys: Primary and Foreign
- A **primary key** should be set to a **unique value** in the table
 - It can be used to take all the field in a record (row)
- What will be the unique key for a student table?

Primary keys

- Should be able to uniquely identify any row in the table (choose wisely!)
- Can sometimes be a combination of values
- Each item should have a unique value (EmpNo)

Columns stores a specific data type

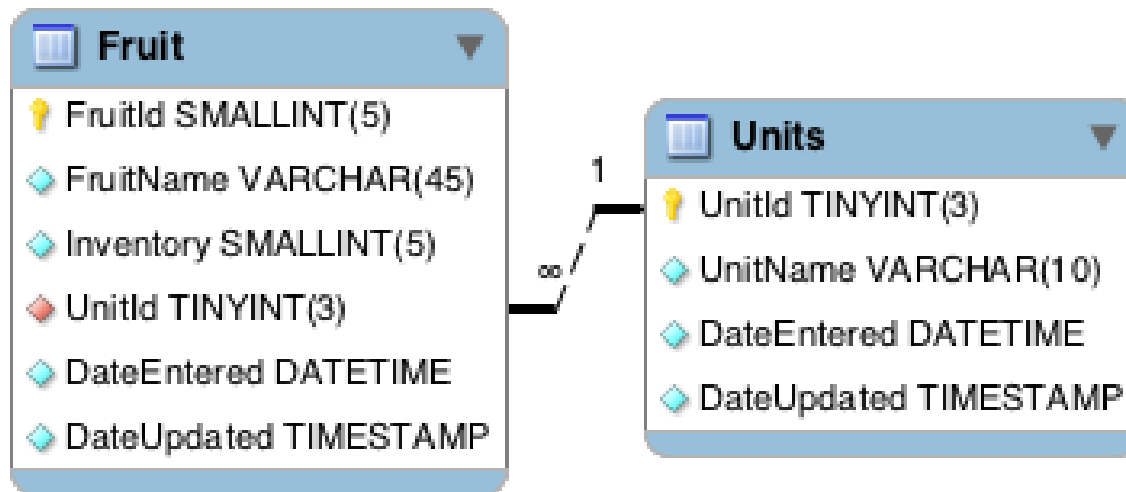


Emp No	Name	Age	Department	Salary
001	Alex S	26	Store	5000
002	Golith K	32	Marketing	5600
003	Rabin R	31	Marketing	5600
004	Jons	26	Security	5100

Row →
Or record

Foreign Keys

- A column or set of columns that need matching pairs in a parent table
- Student Table may have a major identifier that contains all the details about a major
- The Fruit table below has a foreign key which identifies information in the Units table.



Referential Integrity

- Sound database administration practice dictates that table relationships remain stable throughout the lifetime of a project.
- Maintaining referential integrity means making sure no reference points to a record in another (or the same) table if that record is deleted.
 - If a student record is deleted from the enrollment table, her/his associated record in the other tables: courses, library, etc needs to be deleted as well
- Some database storage engine (textbook pp. 569 -570) can enforce the referential integrity: InnoDB

Designing your database

- Think about classes, your design should be based on the objects that you will represent in your web application.
- Much like your classes represent items in the real world so your database tables should also reflect what would be in your class

Avoiding redundant data

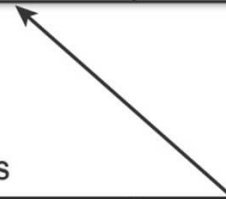
- Notice how the Customer ID is associated with the Order but the Customer details are in a different table.
- This way if we update a customer we only have to update it once in one place!
- This avoids anomalies (UPDATE, INSERT, DELETE)

CUSTOMERS

CustomerID	Name	Address	City
1	Julie Smith	25 Oak Street	Airport West
2	Alan Wong	1/47 Haines Avenue	Box Hill
3	Michelle Arthur	357 North Road	Yarraville

ORDERS

OrderID	CustomerID	Amount	Date
1	3	27.50	02-Apr--2007
2	1	12.99	15-Apr-2007
3	2	74.00	19-Apr-2007
4	3	6.99	01-May-2007

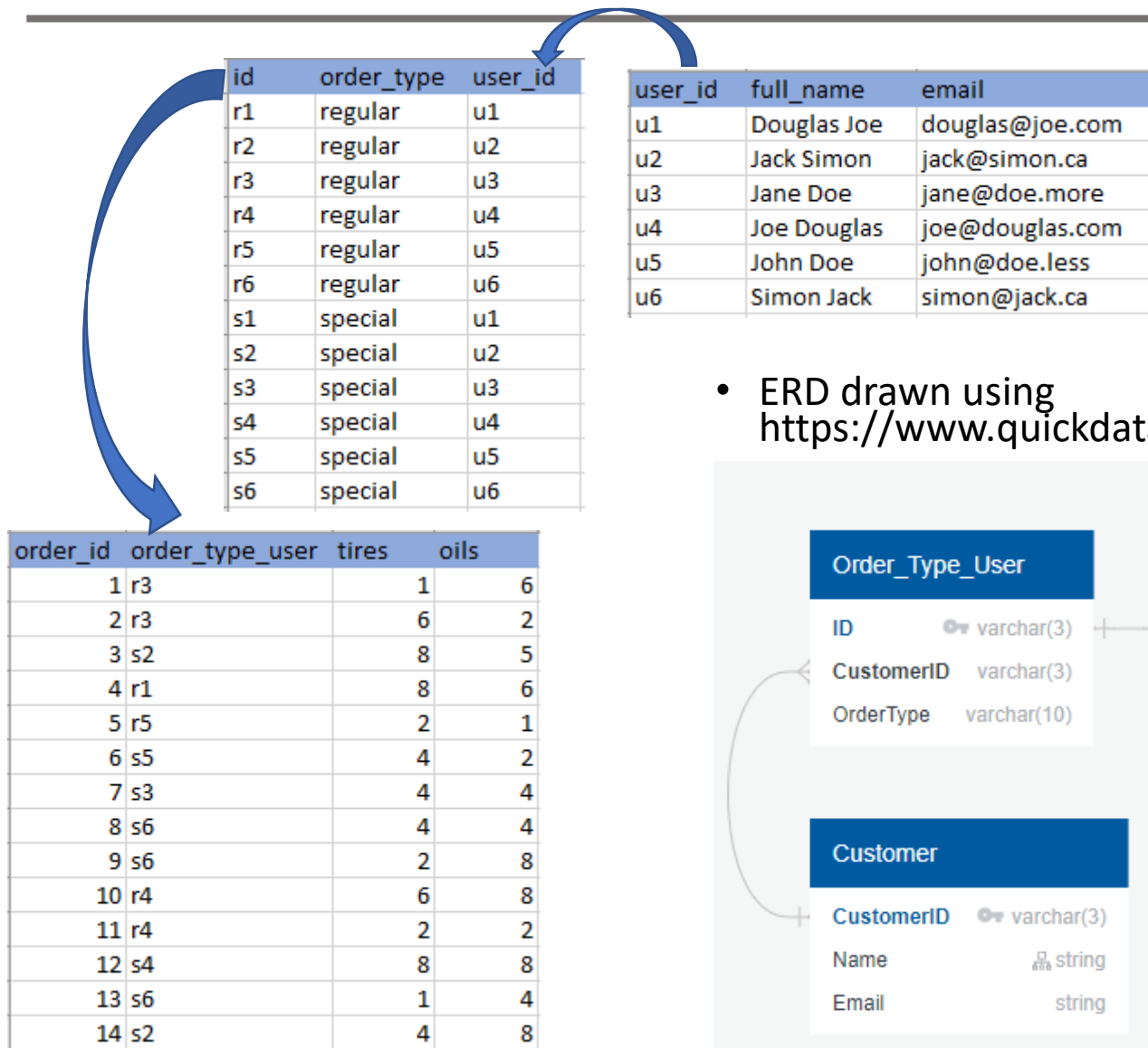


Avoiding redundant data

order_id	order_type	full_name	email	tires	oils
1	regular	Jane Doe	jane@doe.more	1	6
2	regular	Jane Doe	jane@doe.more	6	2
3	special	Jack Simon	jack@simon.ca	8	5
4	regular	Douglas Joe	douglas@joe.com	8	6
5	regular	John Doe	john@doe.less	2	1
6	special	John Doe	john@doe.less	4	2
7	special	Jane Doe	jane@doe.more	4	4
8	special	Simon Jack	simon@jack.ca	4	4
9	special	Simon Jack	simon@jack.ca	2	8
10	regular	Joe Douglas	joe@douglas.com	6	8
11	regular	Joe Douglas	joe@douglas.com	2	2
12	special	Joe Douglas	joe@douglas.com	8	8
13	special	Simon Jack	simon@jack.ca	1	4
14	special	Jack Simon	jack@simon.ca	4	8
15	special	Douglas Joe	douglas@joe.com	5	8

- Notice how the order_type, full_name and email have redundant data?
- We can create another table consisting of user detail, however
 - A user can create multiple orders
 - Different types or order can be made by different users

Example of the DB implementation



- ERD drawn using <https://www.quickdatabasediagrams.com/>

Accessing MySQL Console

- Go to your Command Prompt and type mysql
 - If you are not able to get to the mysql console, it means that you did not setup the environment correctly. You could also go through WAMP's mysql console
- Access MySQL database by providing the username, password and optionally the host

```
C:\Users\home>mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.7.28 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> quit;
Bye
```

Navigating the Databases

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql      |
| performance_schema |
| sys        |
+-----+
4 rows in set (0.00 sec)
```

```
mysql> use mysql;
Database changed
```

```
mysql> show tables;
+-----+
| Tables_in_mysql |
+-----+
| columns_priv     |
| db               |
| engine_cost      |
| event            |
| func             |
| general_log      |
| gtid_executed     |
| help_category    |
| help_keyword     |
| help_relation    |
| help_topic       |
| innodb_index_stats |
| innodb_table_stats |
| ndb_binlog_index |
| plugin           |
| proc             |
+-----+
```

Root password is empty...
Ha! So much for security
Oh well, this is a not a
production environment....

```
mysql> select user, host, authentication_string from user;
+-----+-----+-----+
| user          | host       | authentication_string |
+-----+-----+-----+
| root          | localhost | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
| mysql.session | localhost | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
| mysql.sys     | localhost | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

User management

- Disclaimer, this is not a database course. Proceed the following with your own risk 😊

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'rootPassword!';
Query OK, 0 rows affected (0.01 sec)

mysql> select user, host, authentication_string from user;
+-----+-----+-----+
| user          | host      | authentication_string |
+-----+-----+-----+
| root          | localhost | *3CF5F11041602D823908247C21F865365E2F0019 |
| mysql.session | localhost | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
| mysql.sys     | localhost | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql> flush privileges;
```

You should not use that password for the root though.
Is that the best you can come up with?

User management

- Disclaimer, this is not a database course. Proceed the following with your own risk 😊
- You can create a user login with CREATE USER command, yeah it's so obvious rite....
 - Some people differentiate between mysql command and database/column name. But I don't bother with that
 - TIPS: You should not use the passcode as password
- Assuming that database csis3280_week08 has been created, you can grant some privileges to the newly created user

```
mysql> CREATE USER csis3280 IDENTIFIED BY 'passcode';  
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON csis3280_week08.*  
-> TO 'csis3280' IDENTIFIED BY 'passcode';  
Query OK, 0 rows affected, 1 warning (0.01 sec)  
  
mysql> flush privileges  
-> ;  
Query OK, 0 rows affected (0.01 sec)
```

User management

- Disclaimer, this is not a database course. Proceed the following with your own risk 😊

```
C:\Windows\System32>mysql -u csis3280 -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.28 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| csis3280_week08 |
+-----+
2 rows in set (0.00 sec)

mysql> use csis3280_week08;
Database changed
mysql> show tables;
Empty set (0.00 sec)
```

HELPFUL commands

- SHOW DATABASES; – Show all the databases
- USE *database_name* – change the database into *database_name*
- SHOW TABLES; – Show all the tables in the database
- DESCRIBE *table_name*; – Show all the columns and datatypes in the database
- Quit – quit the mysql console
- Help (or \h) – to get the list of command

Creating Database

- Use the following command: CREATE DATABASE

CREATE DATABASE Lab07_csis3280;

- It is always a good idea to check whether the database exists or not. If it exists, drop the database

```
mysql> DROP DATABASE IF EXISTS Lab07_csis3280;  
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
mysql> CREATE DATABASE Lab07_csis3280;  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> show databases;
```

```
+-----+  
| Database |  
+-----+  
| information_schema |  
| csis3280_week08 |  
| lab07_csis3280 |  
| mysql |  
| performance_schema |  
| sys |  
+-----+  
6 rows in set (0.00 sec)
```

CREATE TABLE

- Basic CREATE TABLE Statement:

```
CREATE TABLE tablename (  
    column_name datatype constraints  
) ENGINE=STORAGE_ENGINE
```

- Tablename – the canonical name you want to give the table (in this course you must append the naming convention ABcXXXXX to any table you create)
- Columname – the canonical name you want to give the column
- Datatype – declares the (textbook calls these column types)
- Constraints (next page)
- Storage engine: InnoDB to enforce referential integrity (we will use this). Default value in WAMP is MyISAM

Create Table -- Constraint

- NOT NULL – must have a value
- AUTO_INCREMENT – This is similar to AutoNumber in Access or IDENTITY in SQL Server – MySQL automatically will generate a number for you
- PRIMARY KEY – Entries must be unique, this is specified for the table.
 - If you need to specify a concatenated key, you must place the PRIMARY KEY constraint on its own.
- FOREIGN KEY – Specify the column in the current table and the table and column it references, they must be the same data type.

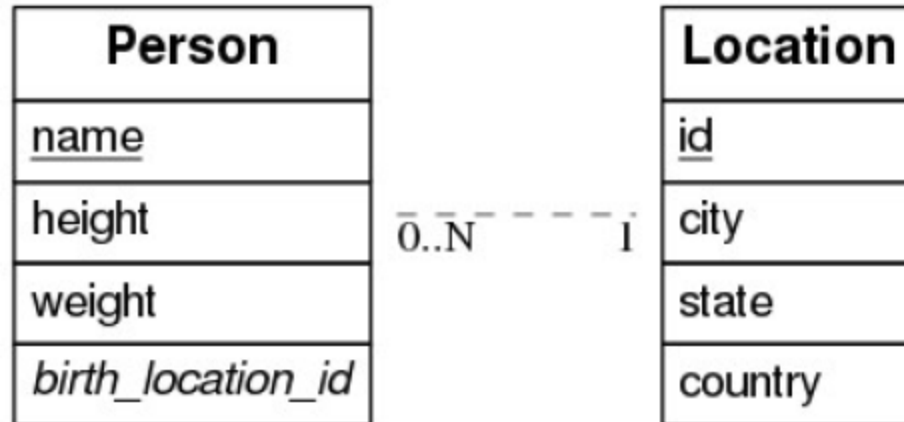
Data Types

- Four basic types:
 - Numeric
 - Date and time
 - String
 - Spatial (for geocoding)
- You should always pick the smallest data type that the data you want to store will fit into.
 - See the table for numeric (integers) data type
 - Please see <https://dev.mysql.com/doc/refman/8.0/en/data-types.html>

Type	Range	Storage (Bytes)	Description
TINYINT [(M)]	-127..128 or 0..255	1	Very small integers
SMALLINT [(M)]	-32768..32767 or 0..65535	2	Small integers
MEDIUMINT [(M)]	-8388608.. 8388607 or 0..16777215	3	Medium-sized integers
INT [(M)]	$-2^{31}..2^{31}-1$ or $0..2^{32}-1$	4	Regular integers
INTEGER [(M)]			Synonym for INT
BIGINT [(M)]	$-2^{63}..2^{63}-1$ or $0..2^{64}-1$	8	Big integers

Create this table!

- Sometimes multiple tables are drawn in an ERD, this table has a foreign key relationship `birth_location_id`, which references the `id` field in the Location table



Create this table!

```
CREATE TABLE Location (  
    id INT (10) AUTO_INCREMENT PRIMARY KEY,  
    city VARCHAR(15),  
    state VARCHAR(2),  
    country VARCHAR(15)  
) ENGINE=InnoDB;
```

```
CREATE TABLE Person(  
    id INT(10) AUTO_INCREMENT PRIMARY KEY,  
    height INT(3),  
    weight INT(3),  
    locationid INT(10) REFERENCES Location(id)  
) ENGINE=InnoDB;
```

Create this table!

```
mysql> use csis3280_week08
Database changed
mysql> CREATE TABLE Location (
  -> id INT (10) AUTO_INCREMENT PRIMARY KEY,
  -> city VARCHAR(15),
  -> state VARCHAR(2),
  -> country VARCHAR(15)
  -> )ENGINE=InnoDB;
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> CREATE TABLE Person(
  -> id INT(10) AUTO_INCREMENT PRIMARY KEY,
  -> height INT(3),
  -> weight INT(3),
  -> locationid INT(10) REFERENCES Location(id)
  -> ) ENGINE=InnoDB;
Query OK, 0 rows affected (0.03 sec)

mysql> show tables;
+-----+
| Tables_in_csis3280_week08 |
+-----+
| location                   |
| person                     |
+-----+
2 rows in set (0.00 sec)
```

```
mysql> describe person;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | int(10)   | NO   | PRI | NULL    | auto_increment |
| height     | int(3)    | YES  |     | NULL    |                |
| weight     | int(3)    | YES  |     | NULL    |                |
| locationid | int(10)   | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

SQL script

- We can create a script to manage our database, and tables
- For example, create a script named exercise.sql as shown.
- Then, you can either
 - Use command line and type `mysql -u username -p password < path_to_sql_file`
 - Use mysql console, i.e., login to mysql and then type `source path_to_sql_file`

```
-- DROP the database if it exists
DROP DATABASE IF EXISTS csis3280_aBc12345;

-- CREATE THE DATABASE
CREATE DATABASE csis3280_aBc12345;

-- SHOW DATABASES;

-- Use the database
USE csis3280_aBc12345;

-- Create the Location Table
CREATE TABLE Location (
  id INT (10) AUTO_INCREMENT NOT NULL,
  city VARCHAR(15),
  state VARCHAR(2),
  country VARCHAR(15),
  PRIMARY KEY(id)
)ENGINE=InnoDB;

-- Create the Person Table
CREATE TABLE Person(
  id INT(10) AUTO_INCREMENT PRIMARY KEY,
  height INT(3),
  weight INT(3),
  locationid INT(10) REFERENCES Location(id)
) ENGINE=InnoDB;
```


Using script: method 1

```
C:\Windows\System32>mysql -u root < C:\temp\csis3280\001\demo07\exercise.sql
```

```
C:\Windows\System32>mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.28 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| csis3280_abc12345 |
| csis3280_week08 |
| lab07_csis3280 |
| mysql |
| performance_schema |
| sys |
+-----+
7 rows in set (0.00 sec)

mysql> use csis3280_abc12345
Database changed
mysql> show tables;
+-----+
| Tables_in_csis3280_abc12345 |
+-----+
| location |
| person |
+-----+
2 rows in set (0.00 sec)
```

Using script: method 2

- Login to mysql
- Use the source command
 - Be careful! **No semicolon**

```
mysql> drop database csis3280_abc12345;
Query OK, 2 rows affected (0.03 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| csis3280_week08 |
| lab07_csis3280 |
| mysql |
| performance_schema |
| sys |
+-----+
6 rows in set (0.00 sec)


mysql> source C:\temp\csis3280\001\demo07\exercise.sql
Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Database changed
Query OK, 0 rows affected (0.03 sec)

Query OK, 0 rows affected (0.03 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| csis3280_abc12345 |
| csis3280_week08 |
| lab07_csis3280 |
| mysql |
| performance_schema |
| sys |
+-----+
```



DML

- Data Manipulation Language
 - SELECT
 - UPDATE
 - INSERT
 - DELETE
- This is the language we use to work with data in our database, this is different from the Data Definition Language we used for creating database objects.
- With the data definition language, the above is analogous to the basic **CRUD: Create, Read, Update, Delete**

SELECT statement

```
SELECT CompanyName, Phone FROM Shippers;
```

```
SELECT * FROM shippers;
```

```
SELECT DISTINCT ShipName FROM Orders;
```

```
SELECT ShipName FROM orders LIMIT 10;
```

```
SELECT CompanyName, Phone FROM customers WHERE  
Country = "Italy";
```

You can also perform subquery from the same table

```
SELECT customerNumber, checkNumber, amount  
FROM payments  
WHERE amount = (  
    SELECT MAX(amount) FROM payments  
);
```

SELECT statement

```
SELECT CompanyName, Country FROM customers WHERE  
Country like 'Sw%';
```

```
SELECT CompanyName, ShipperID FROM shippers WHERE  
ShipperID BETWEEN 2 AND 4;
```

```
SELECT CompanyName, ContactNAME FROM suppliers ORDER  
BY CompanyName DESC, ContactName ASC LIMIT 10;
```

```
SELECT COUNT(*) FROM customers;
```

```
SELECT Avg(UnitPrice) FROM `order details`;
```

```
SELECT min(UnitPrice) FROM `order details`;
```

- **Note:**

- The backtick ` should be used if the name of the field or table are not confirming to style guideline, has white space, or include sql reserved keywords

INSERT statements

- INSERT INTO is used to put new rows in the table (adding a new row of data).

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

```
INSERT INTO table_name VALUES (value1, value2...)
```

```
INSERT INTO shippers (CompanyName, Phone) VALUES
('Canada Post', '123-456-7890');
```

UPDATE statements

- UPDATE command can update existing records in a table

```
UPDATE table_name  
    SET column_name = newvalue  
    WHERE column_name = some_value
```

```
UPDATE Orders  
    SET ShipCountry = "United States"  
    WHERE ShipCountry = "USA";
```

DELETE statements

- DELETE deletes existing records **be careful!**

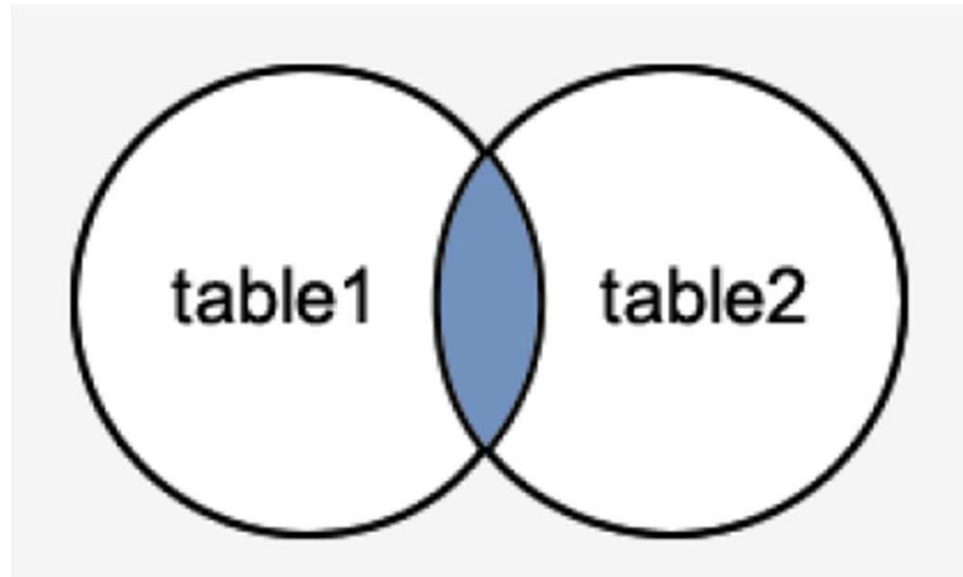
```
DELETE FROM table_name  
WHERE column_name = some_value
```

```
DELETE from shippers  
WHERE CompanyName = "Canada Post";
```

- **If you don't specify a WHERE clause all the rows in the table will be deleted!**

Joins

- Most of the joins you will do in SQL are inner joins, this occurs when you ask for data from the database and specify the relationships between tables (primary keys and foreign keys)



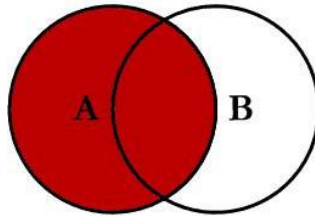
Joins

```
SELECT column_name, column_name  
      FROM table1, table2  
      WHERE table1.key = table2.key
```

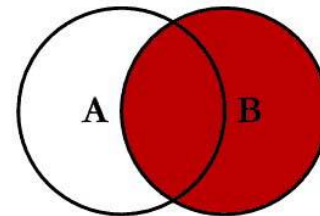
```
SELECT CategoryName, Productname  
      FROM categories, products  
      WHERE products.categoryid = categories.categoryid;
```

Other kinds of Joins

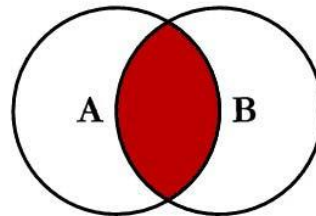
SQL JOINS



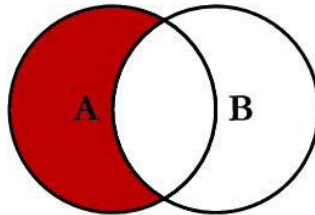
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



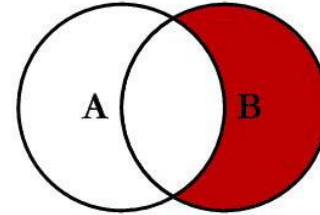
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



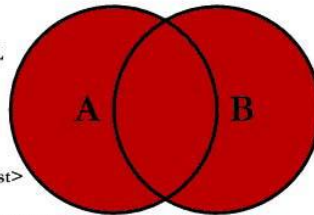
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



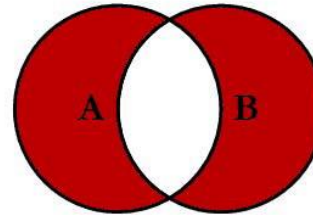
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL  
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

Aliases

- SQL aliases allow you to make addressing columns and tables easier and also generate derived data for the SQL processor to use

```
SELECT column1, column2
      FROM table_name1 AS t1, table_name2 AS t2
      WHERE t1.column1 = t2.column2
```

```
SELECT CategoryName, ProductName
      FROM products AS prod
      INNER JOIN categories AS cat
      ON prod.CategoryID = cat.CategoryID;
```

PHP and MySQL

- Interaction with the MySQL database proceed by connection setup and teardown
 - consisting of connecting to the server and selecting a database,
 - Perform the CRUD, and
 - closing the connection, respectively.
- There are several ways to connect to MySQL
 - mysqli
 - PDO ✓ (will be covered next week)

Connecting to a database

```
mysqli([string host [, string username [, string pswd[,  
string dbname [, int port, [string socket]]]])  
$mysqli = new mysqli('localhost', 'catalog_user',  
'secret', 'corporate');
```

- It is a good practice to store the connection to the database in a separate include file
- You can instantiate a new mysqli object and connect to a database

```
// Instantiate the mysqli class  
$mysqli = new mysqli();  
  
// Connect to the database server and select a database  
$mysqli->connect('localhost', 'catalog_user', 'secret', 'corporate');
```

- Or you can connect and choose a database

```
// Connect to the database server  
$mysqli = new mysqli('localhost', 'catalog_user', 'secret');  
  
// Select the database  
$mysqli->select_db('corporate');
```

Performing query

- In order to perform query, you need to specify the query command (SELECT, INSERT, UPDATE or DELETE)
- Use query() method to get the query result
- Iterate the query result using fetch_row(), fetch_array() or fetch_object()
- Close the connection using close()

```
<?php
```

```
$mysqli = new mysqli('localhost', 'catalog_user', 'secret',  
    'corporate');  
  
// Create the query  
$query = 'SELECT sku, name, price FROM products ORDER by name';  
  
// Send the query to MySQL  
$result = $mysqli->query($query, MYSQLI_STORE_RESULT);  
  
// Iterate through the result set  
while(list($sku, $name, $price) = $result->fetch_row())  
    printf("(%)s %s: \\\$%s <br />", $sku, $name, $price);
```

```
?>
```

Prepared statements

- In order to increase efficiency, we can create a prepared statements for query that are often made.
 - Certain values can be left unspecified using '?'
 - Bind the parameter with the following letters: s (string), i (integer), d (double) or b (blob)

```
// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();
```


References

- Textbook chapter 22, 25, 27
- PHP and MySQL Web Development 5e
- PHP.net
- W3schools.com

Lab and next week

- Please download the Lab. You must submit the lab next week at 9:00 AM
- Next week: PDO