Introduction to Databases

What is a Database?

“A database is **an organized collection of data**. The data is typically organized to *model* relevant aspects of reality, in a way that **supports processes requiring this information**.”

Source: Wikipedia
(Emphasis Mine)

Types of Database

SQL (Relational)

- Highly Structured Data
- Using Tables, Columns and Rows
- One or more relationships exist within the data
- Constraints
  - Primary Keys (unique row identifier)
  - Unique Keys (one or more columns must have unique values, either individually or as a group)
  - Foreign Keys (column value must be derived from a column value in another table)
- Indexes
  - A lookup for rows based on values of one or multiple columns

NoSQL (Document/Key-Value/Graph)

- Sometimes called "**Not Only SQL**" because some NoSQL DBs have a SQL-like query language
- Not always non-relational
- Always unstructured
- Intended to provide higher scalability and higher availability
- Looser consistency models

NoSQL (Document/Key-Value/Graph)

- NoSQL is non-relational
  - Document Stores
    - Centers around the concept of a document, and its related metadata
    - Collections of documents
    - Hierarchies of documents
    - Examples: Couchbase Server, CouchDB, MongoDB, Amazon SimpleDB, Oracle NoSQL DB
  - Key-Value Stores
    - Data stored and accessible directly by a unique key
    - Examples: Memcache, MongoDB, Couchbase Server, Cassandra, Riak, Amazon DynamoDB, Redis, Oracle NoSQL DB

NoSQL (Document/Key-Value/Graph)

- NoSQL is relational (say what?!)
  - Graph Databases
    - » All data is related to N other data
    - » Relationships are in the data, not indexes
    - » Examples: Neo4J, OQGraph for MySQL
    - » Example Implementation: Facebook's Graph API

A Note on MySQL

- MySQL supports multiple drivers (called engines) for its tables.
- These engine provide different features.
- The two most common are InnoDB (default since MySQL 5.5) and MyISAM (previously the default).
- InnoDB has far more features, and is recommended for almost all situations
- We will assume InnoDB for all MySQL examples

Relational Concepts

- Schema
  - Tables
  - Indexes
  - Relationships
- Stored Procedures
- Triggers

Name	What
integer	exact whole numbers
decimal	exact decimal numbers (fixed length)
text	text
blob	binary data
NULL	Null values

Users Table

- Unique Identifier
- Username
- Password
- Email Address
- Name **or** First Name/Last Name

Consider:

- Column Names
- Column Types

User Table

User	
id	integer
username	text
password	text
email	text
first_name	text
last_name	text

Exercise 1:

Create a user table

User Table

CREATE TABLE

User		
id	integer	,
username	text	,
password	text	,
email	text	,
first_name	text	,
last_name	text	

(

);

Users Table (Schema)

```
CREATE TABLE user (
  id INTEGER,
  username TEXT,
  password TEXT,
  email TEXT,
  first_name TEXT,
  last_name TEXT
);
```

SQL: Four Main Queries

- INSERT — Create Data
- UPDATE — Update Existing Data
- SELECT — Fetch Data
- DELETE — Delete Data

CRUD

C reate	INSERT
R etrieve	SELECT
U pdate	UPDATE
D elete	DELETE

Conditions

- Used with:
  - SELECT
  - UPDATE
  - DELETE
  - JOINs
- Preceded by the **WHERE**, **ON**, **USING**, or **HAVING** keyword

Operators

Operator	
=	Equality
<>, !=	Inequality
<	Less Than
<=	Less Than or Equal To
>	Greater Than
>=	Greater Than or Equal To
IS NULL	NULL Equality
IS NOT NULL	NULL Inequality
AND	Boolean AND
OR	Boolean OR
BETWEEN	Range Equality

INSERT

```
INSERT INTO table name (
    list,
    of,
    columns
) VALUES (
    "list",
    "of",
    "values"
);
```

Exercise 2:

Insert a user

INSERT

```
INSERT INTO user (
    id,
    username,
    password,
    email,
    first_name,
    last_name
) VALUES (
    1,
    "dshafik",
    "$2y$10$OI/KS4/Bhs5ENUh7OpIDL.Gs1SIWDG.rPaBkPAjjQ2UTITI60YDmG",
    "davey@engineyard.com",
    "Davey",
    "Shafik"
);
```

UPDATE

```
UPDATE
  table name
SET
  column = "some",
  name = "value"
WHERE
  some condition;
```

WARNING:

Don’t forget your conditions!

Otherwise you update every row in the table!

Additionally, consider using the safe-updates option

Exercise 3:

Update a user

UPDATE

```
UPDATE
  user
SET
  username = "davey",
  email = "davey@engineyard.com"
WHERE
  id = 1;
```

SELECT

```
SELECT
    list, of, columns
FROM
    table
WHERE
    column = "some"
    AND name = "value"
    OR other_column = "other value"
ORDER BY
  some ASC, columns DESC
LIMIT
  start, offset;
```

Exercise 4:

Select one user
with a given username and password

SELECT

```
SELECT
  *
FROM
  user
WHERE
  username = "davey"
  AND password = "$2y$10$Ol..."
LIMIT
  1;
```

Exercise 5:

Select the first 10 users

SELECT

```
SELECT
  first_name, last_name, email
FROM
  user
ORDER BY
  first_name, last_name
LIMIT
  0, 10;
```

Exercise 6:

Select the second 10 users

SELECT

```
SELECT
  first_name, last_name, email
FROM
  user
ORDER BY
  first_name, last_name
LIMIT
  10, 10;
```

DELETE

```
DELETE FROM
    table
WHERE
    column = "some"
    AND name = "value"
    OR other_column = "other value"
ORDER BY
  some ASC, columns DESC
LIMIT
  number;
```

DELETE

```
DELETE FROM
  user;
```

JUST WANT TO DELETE A ROW
AAAAAND THEY'RE ALL GONE
made on imgur

Exercise 7:

Delete one user

DELETE

```
DELETE FROM
  user
WHERE
  id = 1;
```

Constraints: Users Table

- IDs should be **unique**
- Usernames should be **unique**
- Passwords **should not be** unique
- Email Address should be **unique**
- First Name **should not be** unique
- Last Name **should not be** unique

- All column **should not be null**

Constraints: Users Table

Users		Constraints
id	integer	not null, unique
username	text	not null, unique
password	text	not null
email	text	not null, unique
first_name	text	not null
last_name	text	not null

Constraints: Users Table Schema

```
DROP TABLE
  user;

CREATE TABLE user (
  id INTEGER NOT NULL UNIQUE,
  username TEXT NOT NULL UNIQUE,
  password TEXT NOT NULL,
  email TEXT NOT NULL UNIQUE,
  first_name TEXT NOT NULL,
  last_name TEXT NOT NULL
);
```

Users Table: AutoIncrement

- ID should be **autoincrement**
- ID should be the **primary key**

Features: Users Table Schema

```
CREATE TABLE user (
  id INTEGER PRIMARY KEY AUTOINCREMENT
    NOT NULL,
  username TEXT NOT NULL UNIQUE,
  password TEXT NOT NULL,
  email TEXT NOT NULL UNIQUE,
  first_name TEXT NOT NULL,
  last_name TEXT NOT NULL
);
```

Entry Table

- Unique identifier
- Title
- Article

Consider:

- Must link to the user table

Entry Table

Entry		
id	integer	primary key, autoincrement
user_id	integer	not null
title	text	not null
article	text	not null

Entry Table Schema

```
CREATE TABLE entry (
  id INTEGER NOT NULL PRIMARY KEY
    AUTOINCREMENT,
  user_id INTEGER NOT NULL,
  title TEXT NOT NULL,
  article TEXT NULL
 );
```

INSERT

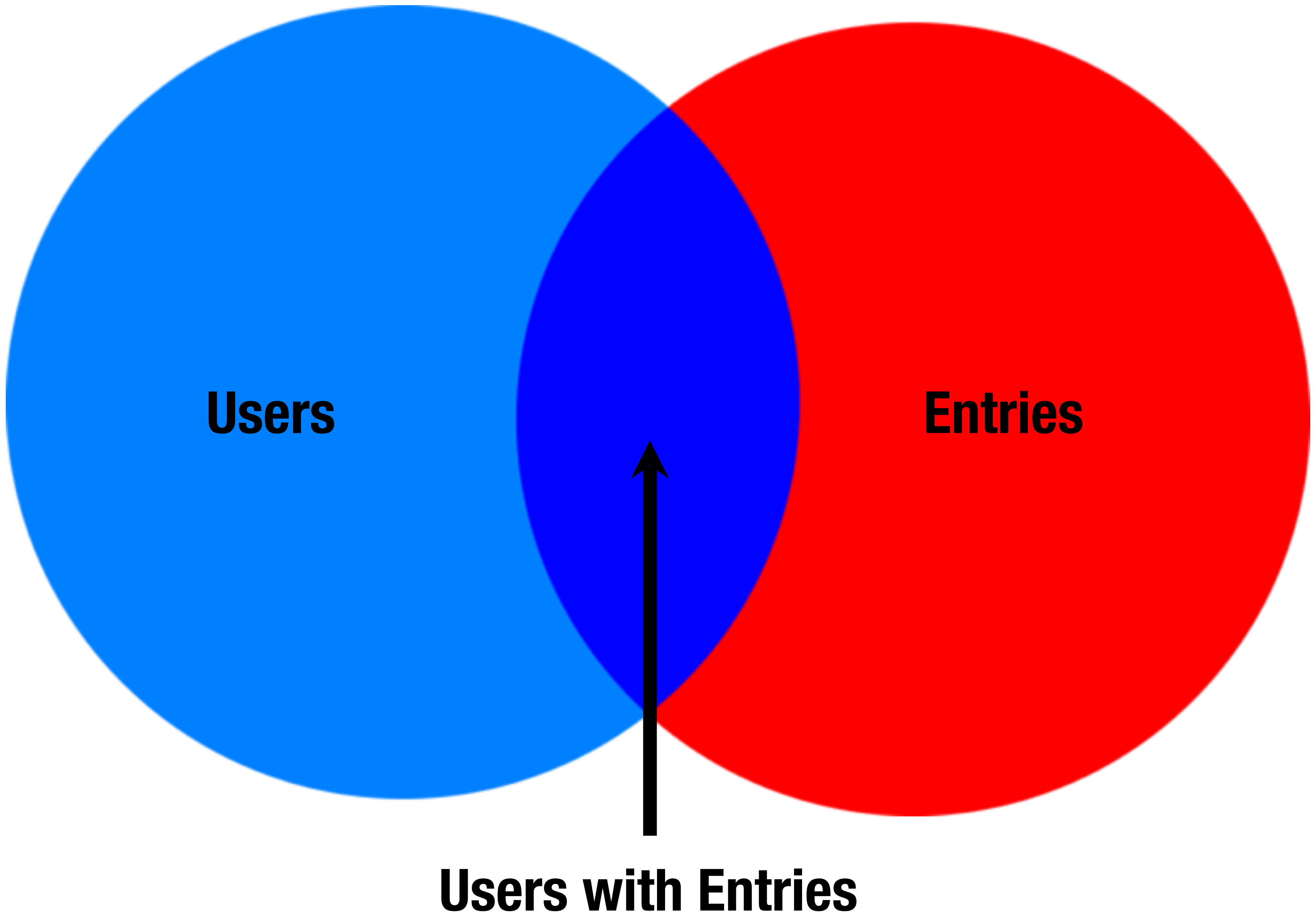
```sql
INSERT INTO entry (
    user_id,
    title,
    entry
) VALUES (
    1,
    "How to Write SQL",
    "Writing SQL in PHP is fun and easy!"
);
```

JOINs

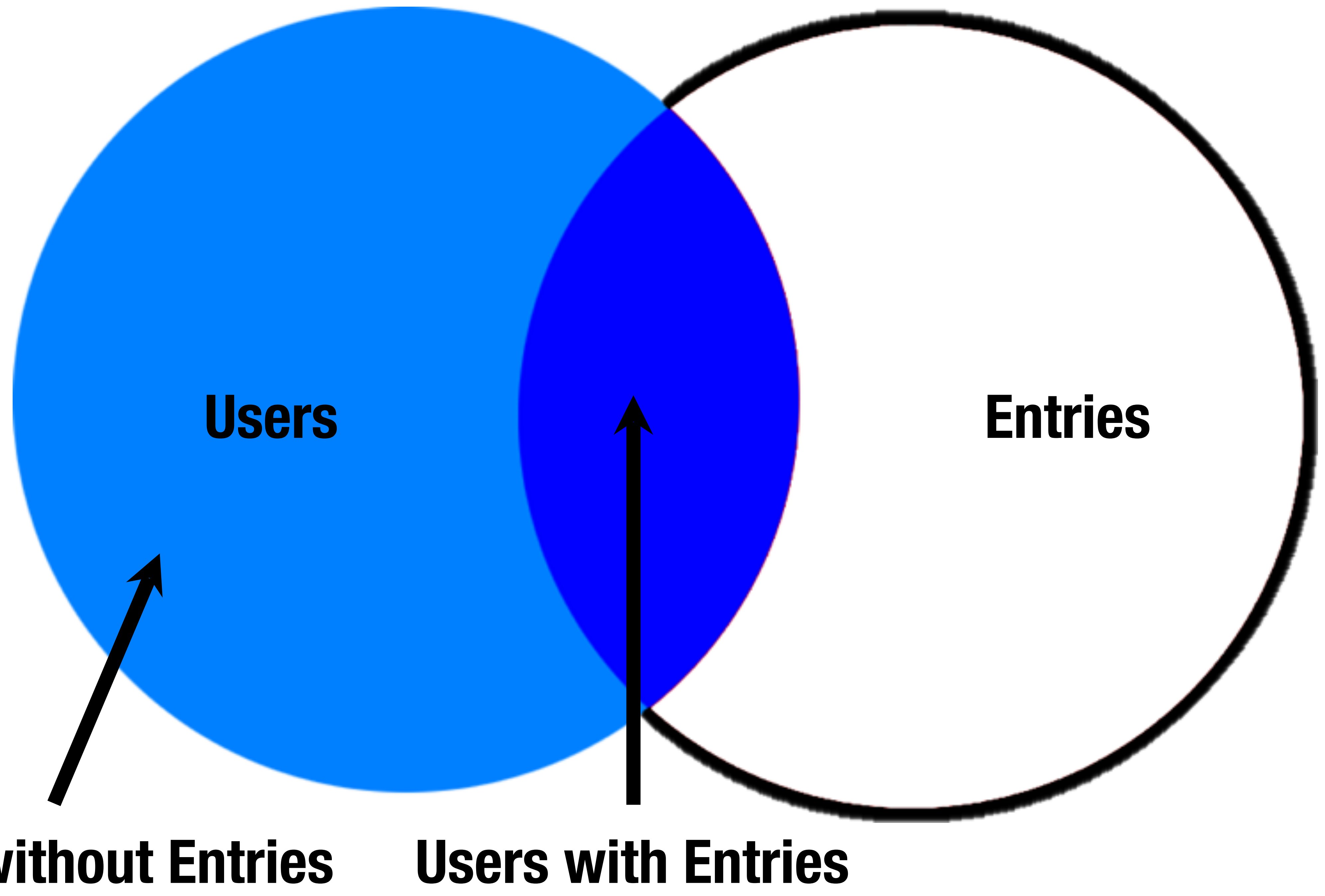
- Used to JOIN multiple tables
  - INNER JOIN
  - LEFT or RIGHT OUTER JOIN
- See: A Visual Explanation of SQL Joins

```
SELECT
  *
FROM
  table_1
INNER JOIN
  table_2 ON (condition);
```

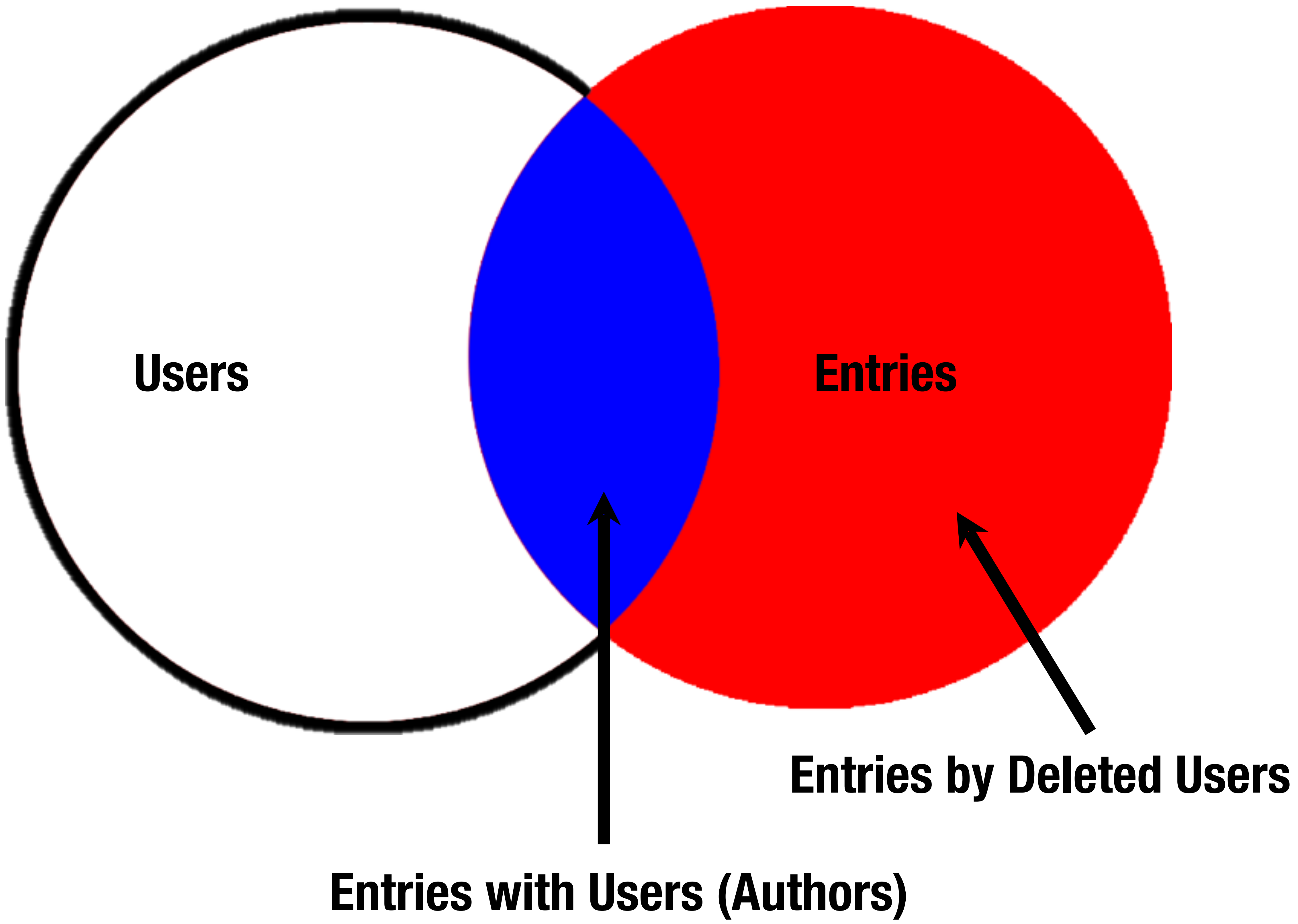
INNER JOIN



LEFT OUTER JOIN



RIGHT OUTER JOIN



Exercise 8:

Select all entries that have users

SELECT... INNER JOIN

```
SELECT
  *
FROM
  entry
INNER JOIN
  user ON (entry.user_id = user.id);
```

Exercise 9:

Select all entries, with users where available

SELECT... LEFT OUTER JOIN

```
SELECT
  *
FROM
  entry
LEFT OUTER JOIN
  user ON (entry.user_id = users.id);
```

Connecting to Databases

- PDO
  - SQLite
  - MySQL
  - PostgreSQL
  - MSSQL
  - Oracle
  - ODBC and DB2
  - Firebird
- DSN — Data Source Name
  - Driver Name
  - Hostname & Port or Unix Socket
  - Username
  - Password
  - Database Name
  - Charset
- Connecting
  - new PDO()

Connecting to SQLite

```php
<?php
try {
    $pdo = new \PDO("sqlite:/path/to/db.sqlite");
} catch (\PDOException $ex) {
    error_log($ex->getMessage());
}
?>
```

Executing Queries

```
try {
    $pdo = new \PDO(…);
    $query = $pdo -> prepare(
        "SELECT * FROM user WHERE id = :id"
    );

    $conditions = array(
        ':id' => 1
    );

    $result = $query->execute($conditions);
} catch (\PDOException $ex) {
    error_log($ex->getMessage());
}
```

Handling Results

```php
<?php
$result = $query->execute($conditions);
if ($result) {
  echo "Results Found: " . $query->rowCount();
  while ($row = $query->fetch()) {
    echo "<a href='/edit/" . $row[‘id'] . "'>"
          . $row[‘first_name'] . ' '
          . $row['last_name'] . '</a>';
  }
}
?>
```

Handling Results as Objects

```php
<?php
$result = $query->execute($conditions);
if ($result) {
  echo "Results Found: " . $query->rowCount();
  while ($row = $query->fetchObject()) {
    echo "<a href='/edit/" . $row->id . "'>"
        . $row->first_name . ' '
        . $row->last_name . '</a>';
  }
}
?>
```

Handling Results as Custom Objects

```
class User {
  function getName() {
    return $this->first_name
         . ' ' . $this->last_name;
  }
}

if ($result) {
  echo "Results Found: " . $query->rowCount();
  while ($row = $query->fetchObject("User")) {
    echo "<a href='/edit/" . $row->id . "'>"
         . $row->getName() . '</a>';
  }
}
```

Feedback & Questions:

Twitter: @elazar

Email: me@matthewturland.com

Slides: http://github.com/phpembark/phpembark