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

SQL (Relational)

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

NoSQL

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

Non-Relational

NoSQL (Document/Key-Value/Graph)

- NoSQL is non-relational
  - Document Stores
    - » Centers around the concept of a document, and it's related meta-data
    - » 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

Relational

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

Relational Concepts

Relational Concepts

Relational Concepts

- Schema

Relational Concepts

- Schema
  - Tables

Relational Concepts

- Schema
  - Tables
  - Indexes

Relational Concepts

- Schema
  - Tables
  - Indexes
  - Relationships

Relational Concepts

- Schema
  - Tables
  - Indexes
  - Relationships
- Stored Procedures

Relational Concepts

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

Data Types

Name	What

Name	What
int	exact whole numbers

Name	What
int	exact whole numbers
decimal	exact decimal numbers (fixed length)

Name	What
int	exact whole numbers
decimal	exact decimal numbers (fixed length)
text	text

Name	What
int	exact whole numbers
decimal	exact decimal numbers (fixed length)
text	text
blob	binary data

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

Create a Users Table

Users Table

Users Table

- Unique Identifier

Users Table

- Unique Identifier
- Username

Users Table

- Unique Identifier
- Username
- Password

Users Table

- Unique Identifier
- Username
- Password
- Email Address

Users Table

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

Users Table

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

Users Table

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

Users Table

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

Users Table

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

Users Table

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

Consider:

- Column Names
- Column Types

User Table

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

Exercise 1: Create a User Table

User Table

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

User Table

```
CREATE TABLE
```

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

User Table

```
CREATE TABLE
```

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

(

User Table

CREATE TABLE

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

(

,

User Table

CREATE TABLE

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

(

,

User Table

CREATE TABLE

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

(
,
,

User Table

```
CREATE TABLE
```

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

```
(
,
,
,
```

User Table

CREATE TABLE

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

(
,
,
,
,

User Table

```
CREATE TABLE
```

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

```
(
,
,
,
,
,
);
```

Users Table (Schema)

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

SQL

SQL: Four Main Queries

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

CRUD

CRUD

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

Conditions

Conditions

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

Operators

Operators

Operator	

Operators

Operator	
=	Equality
<>, !=	Inequality

Operators

Operator	
=	Equality
<>, !=	Inequality
<	Less Than
<=	Less Than or Equal To
>	Greater Than
>=	Greater Than or Equal To

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

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

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

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

Exercise 2: Insert a User

INSERT

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

UPDATE

UPDATE

```
UPDATE
   table name
SET
   column = "some",
   name = "value"
```

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!

Exercise 3: Update a User

UPDATE

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

SELECT

SELECT

```
SELECT
    list,
    of,
    columns
FROM
    table
```

SELECT

```
SELECT
    list,
    of,
    columns
FROM
    table
```

```
WHERE
    column = "some"
    AND name = "value"
    OR other_column = "other value"
```

SELECT

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

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
  users
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
  users
ORDER BY first_name, last_name
LIMIT 0, 10;
```

Exercise 6: Select the Next 10 Users

SELECT

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

DELETE

DELETE

```
DELETE
FROM
     table
```

DELETE

```
DELETE
FROM
    table
WHERE
    column = "some"
    AND name = "value"
    OR other_column = "other value"
```

DELETE

```
DELETE
FROM
    table
WHERE
    column = "some"
    AND name = "value"
    OR other_column = "other value"

ORDER BY some ASC, columns DESC
```

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 users;
```


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

Exercise 7: Delete a User

DELETE

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

Constraints

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

Constraints: Users Table

Constraints: Users Table

Users

Constraints: Users Table

Users	Constraints

Constraints: Users Table

Users		Constraints
id	int	

Constraints: Users Table

Users		Constraints
id	int	not null, unique

Constraints: Users Table

Users		Constraints
id	int	not null, unique
username	text	

Constraints: Users Table

Users		Constraints
id	int	not null, unique
username	text	not null, unique

Constraints: Users Table

Users		Constraints
id	int	not null, unique
username	text	not null, unique
password	text	

Constraints: Users Table

Users		**Constraints**
id	int	not null, unique
username	text	not null, unique
password	text	not null

Constraints: Users Table

Users		Constraints
id	int	not null, unique
username	text	not null, unique
password	text	not null
email	text	

Constraints: Users Table

Users		Constraints
id	int	not null, unique
username	text	not null, unique
password	text	not null
email	text	not null, unique

Constraints: Users Table

Users		**Constraints**
id	int	not null, unique
username	text	not null, unique
password	text	not null
email	text	not null, unique
first_name	text	

Constraints: Users Table

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

Constraints: Users Table

Users		Constraints
id	int	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	

Constraints: Users Table

Users		Constraints
id	int	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 INT 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
);
```

Auto Increment

Users Table: Auto Increment

- ID should be **auto increment**
- ID should be the **Primary Key**

Features: Users Table Schema

```
CREATE TABLE users (
    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
);
```

Entries

Entry Table

Entry Table

- Unique Identifier

Entry Table

- Unique Identifier
- Title

Entry Table

- Unique Identifier
- Title
- Article

Entry Table

- Unique Identifier
- Title
- Article

Entry Table

- Unique Identifier
- Title
- Article

- Must link to the Users table

Entry Table

- Unique Identifier
- Title
- Article

Consider:

- Must link to the Users table

Entry Table

Entry Table

Entry Table

Entry

Entry Table

Entry
id

Entry Table

Entry	
id	int PRIMARY KEY AUTOINCREMENT
users_id	

Entry Table

Entry	
id	int PRIMARY KEY AUTOINCREMENT
users_id	int

Entry Table

Entry	
id	int PRIMARY KEY AUTOINCREMENT
users_id	int
title	TEXT

Entry Table

Entry	
id	int PRIMARY KEY AUTOINCREMENT
users_id	int
title	TEXT
article	

Entry Table

Entry	
id	int PRIMARY KEY AUTOINCREMENT
users_id	int
title	TEXT
article	TEXT

Entry Table Schema

```
CREATE TABLE entry (
   id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
   users_id INT NOT NULL,
   title TEXT NOT NULL,
   article TEXT NULL,
 );
```

INSERT

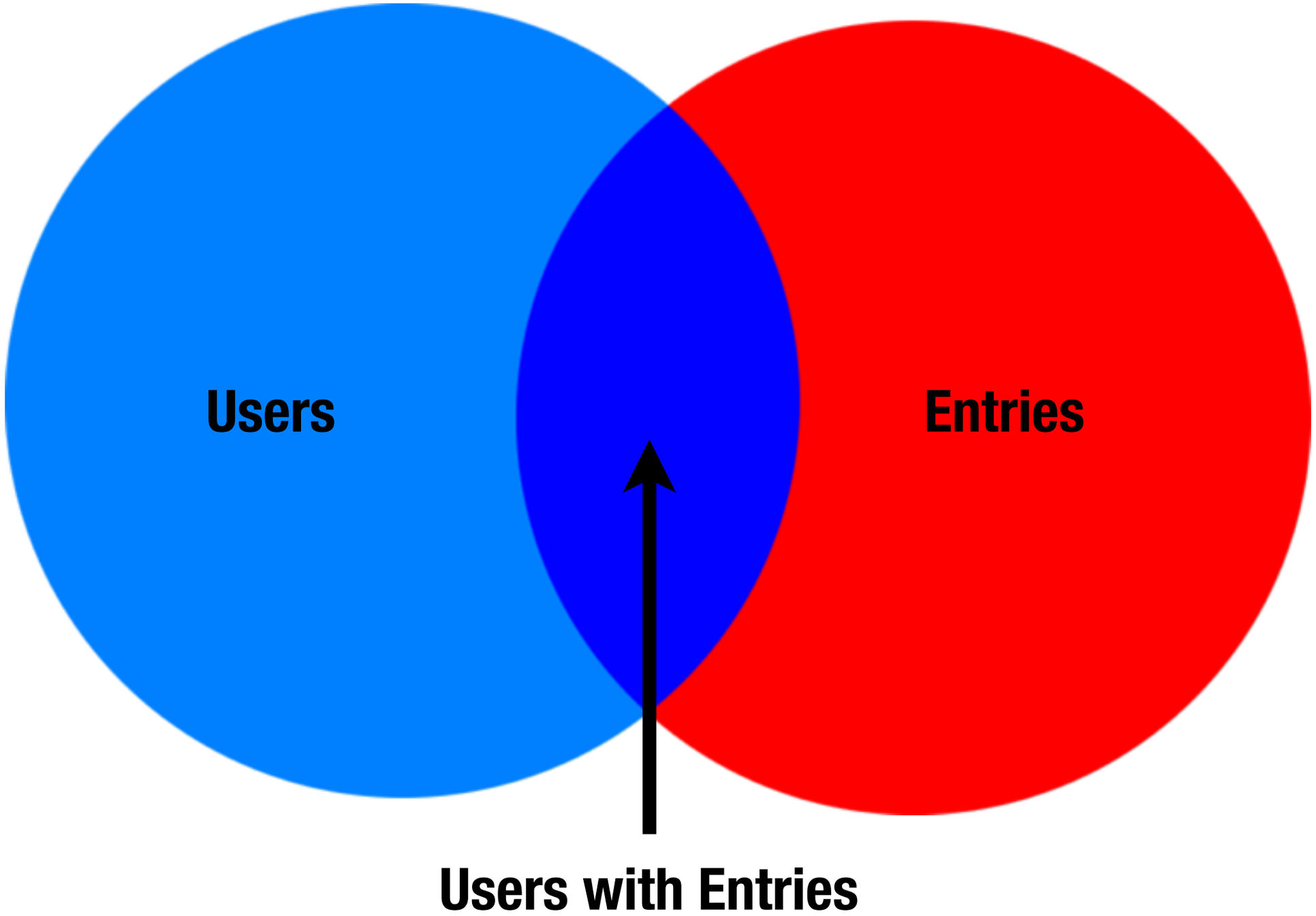
```
INSERT INTO entry (
    users_id,
    title,
    entry
) VALUES (
    1,
    "How to Write SQL",
    "Writing SQL in PHP is fun and easy!"
);
```

JOINs

JOINs

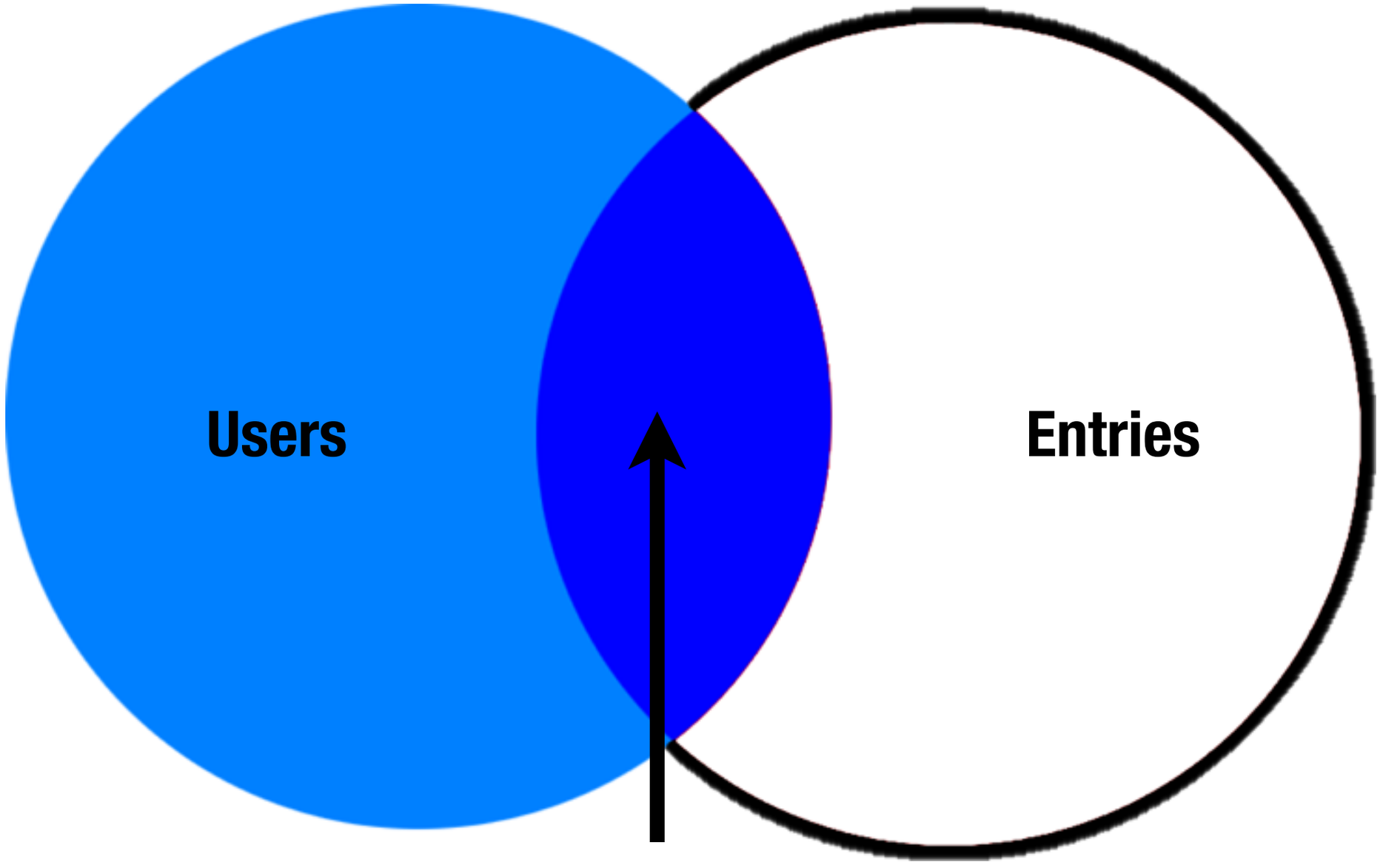
- Used to JOIN multiple tables
  - INNER JOIN
  - LEFT or RIGHT OUTER JOIN

INNER JOIN



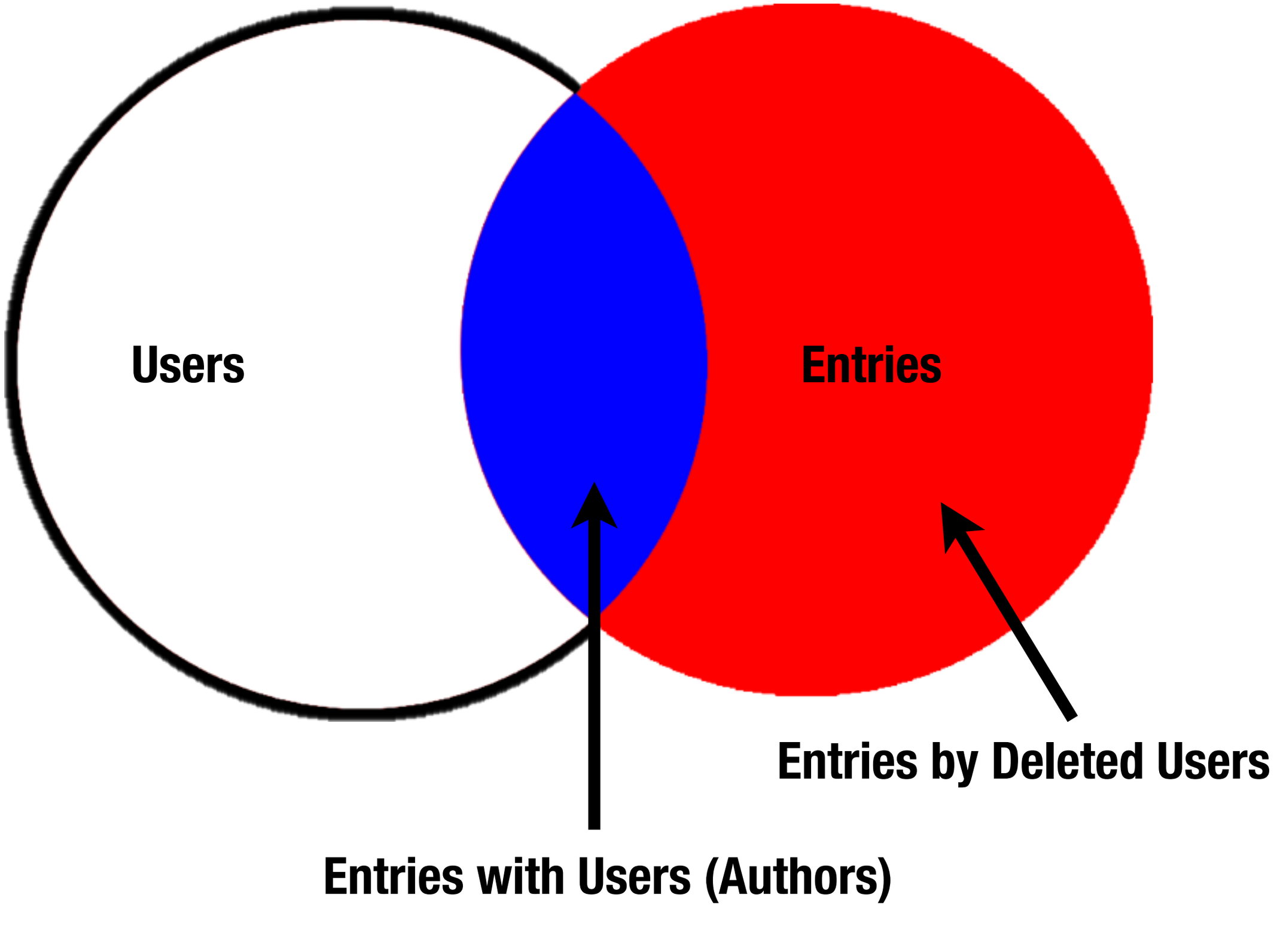
OUTER JOIN

LEFT OUTER JOIN



Users with Entries

RIGHT OUTER JOIN



Exercise 8: Get a User and all their Entries

SELECT... INNER JOIN

```
SELECT * FROM
   users
   INNER JOIN entry
   ON (
      entry.user_id = users.id
   )
WHERE
   entry.title LIKE '%PHPEmbark%'
ORDER BY
   entry.title ASC;
```

Exercise9: Get Users who may or may not have Entries

SELECT... LEFT OUTER JOIN

```
SELECT * FROM
   users
   LEFT OUTER JOIN entry
   ON (
      entry.user_id = users.id
   )
WHERE
   users.id = 1;
```

Exercise 10: Get a User, who might have posts, or have stuff tagged with ‘PHP’

SELECT... RIGHT OUTER JOIN

```
SELECT * FROM
  user
  LEFT OUTER JOIN entry
  ON (
    entry.user_id = users.id
  )
  RIGHT OUTER JOIN tag
  ON (
    entry.tag_id = tag.id
  )
WHERE
  tag.name = 'PHP';
```

Databases and PHP

Connecting to Databases

Connecting to Databases

- PDO
  - MySQL
  - PostgreSQL
  - MSSQL
  - Oracle
  - SQLite
  - ODBC and DB2
  - Firebird
- DSN — Data Source Name
  - Driver Name
  - Hostname & Port

Connecting to SQLite

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

Querying Data

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

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>';
   }
}
```