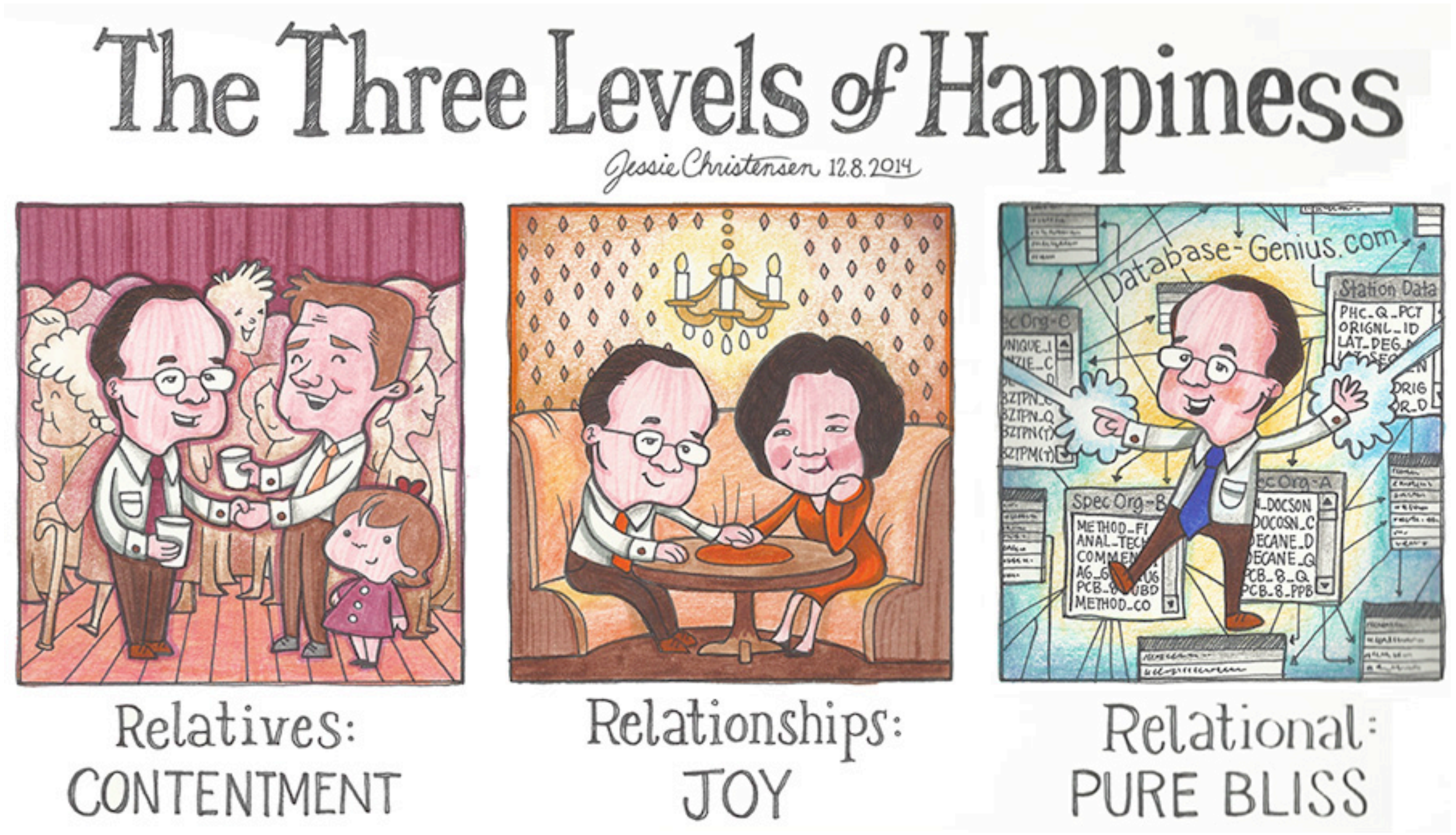
CSC 337



https://www.database-genius.com/

Database Design and More Commands

Rick Mercer

Relational Data
The R is RDBMS

- A relational data base spreads data across tables
- Data is joined using a key common to tables
  - The key is the relation between tables
  - The primary key must be unique
  - Form links between tables in a foreign key relationship
    - Match a customer in TABLE customer with a sales order in TABLE sales_order
    - Match all reviews in TABLE Reviews to a specific Reviewer
- Think of how many records in one table relate to how many records in another table
  - There are three types of relationships

Database Relationships

- Relationships are quantified as
  - One to one
  - One to many
  - Many to many

1) One-to-One

- Each record is related to exactly one record
  - Each reviewer has exactly one address

Reviewers

Jo	Jo	Miller
Chris	Chris	Baker
Kim	Kim	Cook

Addresses

Jo	Daily Beast
Chris	Wall Street Journal
Kim	New York Times

2) One-to-Many

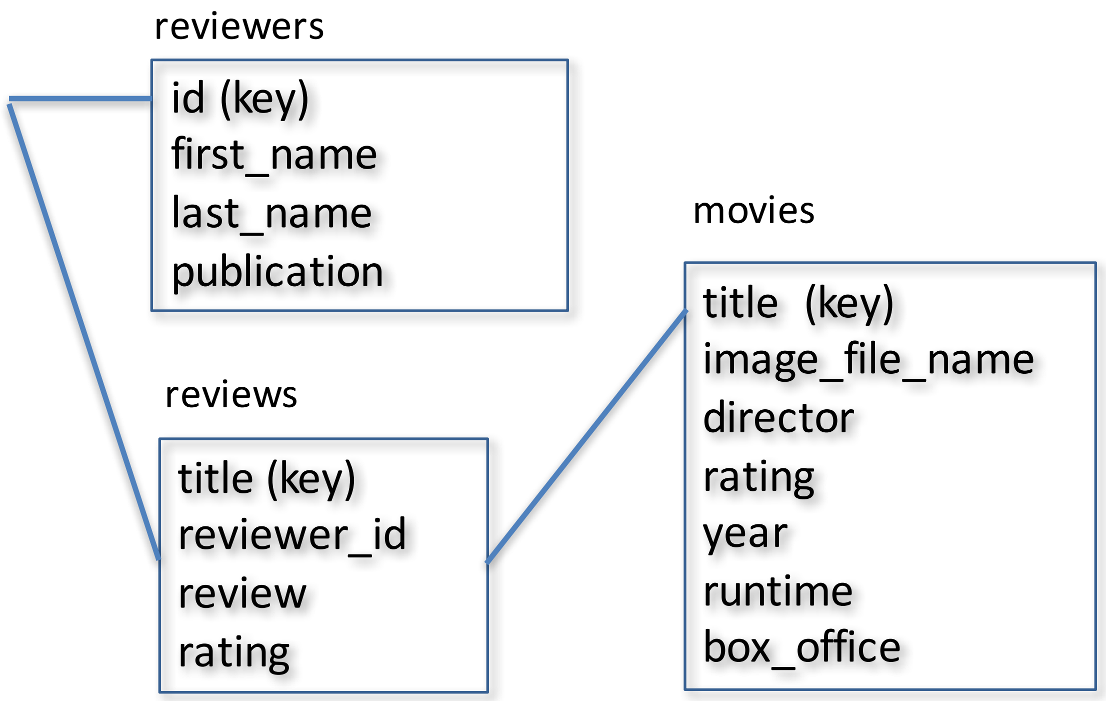
- A key from one table appears many times in another table, the most common relationship
  - Each movie has many reviewers

Reviews

Die Hard 7	Jo	rotten	Time warped
Die Hard 7	Chris	rotten	Flippin' jet skis? Come on.
Die Hard 7	Kim	fresh	He's finally getting a little tired?
Die Hard 7	Fourth	rotten	Predictably predictable
Die Hard 8	Fifth	fresh	Gotta love patterns applied for the 8th time
Die Hard 8	Jo	rotten	Warped time

3) Many-to-Many

- Many reviewers can review the same movie
- Many movies can be reviewed by many reviewers



Keys

- **Primary key** the main reference for the table
- It is used throughout the database to help establish relationships with other tables
- **Foreign key** is a primary key from one table that appears as a field in a different table
- If table A has a primary key X that linked to a table B where X was a field in B, then
  - X would be a foreign key in B
- Later, we will use both keys in SQL joins
  - Get all reviews for one movie for example

Could have one table (bad)
Normalization

- Organizing data to minimize duplication of data
- First normal form (1NF) *do_**not**_use this to store all reviews* ↓
  - No redundant data in a row ← *use 3 tables*
  - Exactly one value per column
- Second normal form (2NF)
  - No columns repeat their values in rows
- Third normal form
  - Data dependent on the primary key
  - Would mean an address be split into
    - zip_codes, cities, states
- We will settle for second normal form

reviewer (key)
last_name
first_name
address
review
rating
title
director
rating
release_year
runtime
box_office

A few column data types

Type name	example
INT	123
FLOAT	4.56
VARCHAR(256)	"A string up to a length of 256"
DATE	2015-11-30
TIME	09:37:59

- Use auto_increment to guarantee unique keys
- Hard coding? Ensure unique keys
  - Unique reviewer ids, unique movie titles
    - You have to hand check
- Better to set a column as the PRIMARY KEY

More SQL Commands

SHOW DATABASES

```
MariaDB [imdb]> show databases
+--------------------+
| Database           |
+--------------------+
| first              |
| imdb_small         |
| information_schema |
| movie_titles       |
| mysql              |
| performance_schema |
| phpmyadmin         |
| quotes             |
| rancid_tomatoes    |
| second             |
| simpsons           |
| test               |
+--------------------+
12 rows in set (0.00 sec)

MariaDB [imdb]>
```

CREATE DATABASE, USE, NULL, DESCRIBE, AUTO_INCREMENT, PRIMARY KEY

```
CREATE DATABASE imdb;
USE imdb;
CREATE TABLE actors (
  id int(11) NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(16) NOT NULL,
  last_name VARCHAR(16) NOT NULL,
  gender char(1) NOT NULL,
  film_count int(11) NOT NULL,
  PRIMARY KEY(id)
);
DESCRIBE actors;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | int(11)      | NO   | PRI | 0       | auto_increment |
| first_name | varchar(100) | YES  |     | NULL    |                |
| last_name  | varchar(100) | YES  |     | NULL    |                |
| gender     | char(1)      | YES  |     | NULL    |                |
| film_count | int(11)      | YES  |     | 0       |                |
+------------+--------------+------+-----+---------+----------------+
```

INSERT records

- Two ways: 1) You specify the key

```
insert into actors values (1, 'Chris', 'Wall', 'f', 8 );
```

- or 2) with an auto_increment primary key, let the RDBMS assign the unique key
  - Specify columns first, skip the primary key column, and then specify the values

```
insert into actors(last_name, first_name, gender,
film_count) values ('Bette', 'Davis', 'f', 2 );
```

DELETE

```
MariaDB [imdb]> DELETE from actors where id >= 0;
```

```
MariaDB [imdb]> insert into actors(first_name, last_name,
gender, film_count)  values ('Bette', 'Davis', 'f', 6 );
```

- Notice the DBMS knows ids 1, 2, and 3 were previously assigned, so start with 4

```
MariaDB [imdb]> select * from actors;
+----+------------+-----------+--------+------------+
| id | first_name | last_name | gender | film_count |
+----+------------+-----------+--------+------------+
|  4 | Bette      | Davis     | f      |          6 |
+----+------------+-----------+--------+------------+
```

DROP TABLE

```
DROP TABLE actors;
-- Create the TABLE again to see the first key will be 1 again
CREATE TABLE actors (
  id int(11) NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(16) NOT NULL,
  last_name VARCHAR(16) NOT NULL,
  gender char(1) NOT NULL,
  film_count int(11) NOT NULL,
  PRIMARY KEY(id)
);
insert into actors(last_name, first_name, gender, film_count)
values ('Bette', 'Davis', 'f', 6 );
select * from actors;  -- Auto increment starts at 1 after CREATE
+----+------------+-----------+--------+------------+
| id | first_name | last_name | gender | film_count |
+----+------------+-----------+--------+------------+
|  1 | Bette      | Davis     | f      |          6 |
+----+------------+-----------+--------+------------+
```

RDBs have more than 1 table

```
CREATE TABLE roles (
  actor_id int(11),   movie_id int(11),   role VARCHAR(100)
);
DESCRIBE roles;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| actor_id | int(11)      | YES  |     | NULL    |       |
| movie_id | int(11)      | YES  |     | NULL    |       |
| role     | varchar(100) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
MariaDB [imdb]> SHOW TABLES;
+----------------+
| Tables_in_imdb |
+----------------+
| actors         |
| roles          |
+----------------+
```

Insert a few actors

```
insert into actors(first_name, last_name, gender,
  film_count) values ('Chris', 'Miller', 'f', 3 );
insert into actors(first_name, last_name, gender,
  film_count) values ('Ali', 'Sauong', 'f', 24 );
insert into actors(first_name, last_name, gender,
  film_count) values ('Jerrie', 'Mander', 'f', 1 );
```

```
select * from actors;
+----+------------+-----------+--------+------------+
| id | first_name | last_name | gender | film_count |
+----+------------+-----------+--------+------------+
|  1 | Chris      | Miller    | f      |          3 |
|  2 | Ali        | Sauong    | f      |         24 |
|  3 | Jerrie     | Mander    | f      |          1 |
+----+------------+-----------+--------+------------+
```

Insert a few roles

```
insert into roles(actor_id, movie_id, role)
          values (1, 5743, 'Herself' );
insert into roles(actor_id, movie_id, role)
          values (1, 6811, 'Courier' );
insert into roles(actor_id, movie_id, role)
          values (1, 4013, 'Chalk Artist' );
```

```
MariaDB [imdb]> select * from roles;
+----------+----------+--------------+
| actor_id | movie_id | role         |
+----------+----------+--------------+
|        1 |     5743 | Herself      |
|        1 |     6811 | Courier      |
|        1 |     4013 | Chalk Artist |
+----------+----------+--------------+
```

Let's use imdb_small

LIKE

- Use the symbol '%' as a wild card character

```
-- Find any value beginning with 'T' or 't'
SELECT * FROM actors WHERE first_name role LIKE 'T%';

-- Find any value ending with 'T' or 't'
SELECT * FROM actors WHERE first_name role LIKE '%T';

-- Find any value containing 'T' or 't'  anywhere
SELECT * FROM actors WHERE first_name role LIKE '%T%';
```

WHERE

- WHERE is followed by a Boolean expression
- Can use Boolean operators AND NOT OR
- Can use relational operators <= != > =

```
select * from roles WHERE role > 'D';
+----------+----------+---------+
| actor_id | movie_id | role    |
+----------+----------+---------+
|        1 |     5743 | Herself |
+----------+----------+---------+
```

WHERE, LIKE

```
SELECT * FROM actors WHERE first_name LIKE '%lint%'
          OR last_name LIKE '%lint%';
+--------+------------+-----------+--------+------------+
| id     | first_name | last_name | gender | film_count |
+--------+------------+-----------+--------+------------+
| 208323 | Clint      | Hill      | M      |          1 |
| 215405 | Clint      | Howard    | M      |          1 |
| 296955 | Clint      | Mansell   | M      |          1 |
| 621159 | Jill       | Flint     | F      |          1 |
+--------+------------+-----------+--------+------------+
```

IN

- Use IN to replace longer Boolean expressions

```
SELECT * FROM actors WHERE first_name IN ('Jill', 'Clint')
           AND last_name IN ('Hill', 'Flint');
+--------+------------+-----------+--------+------------+
| id     | first_name | last_name | gender | film_count |
+--------+------------+-----------+--------+------------+
| 208323 | Clint      | Hill      | M      |          1 |
| 621159 | Jill       | Flint     | F      |          1 |
+--------+------------+-----------+--------+------------+
```

BETWEEN

- Use BETWEEN to select values in range
  - Begin and end values are included (the name is bad)

```
SELECT * FROM actors WHERE film_count BETWEEN 3 and 9;
+--------+-------------+-----------+--------+------------+
| id     | first_name  | last_name | gender | film_count |
+--------+-------------+-----------+--------+------------+
|  22591 | Kevin       | Bacon     | M      |          9 |
|  65536 | Steve       | Buscemi   | M      |          3 |
| 292028 | Michael (I) | Madsen    | M      |          3 |
| 366173 | Bill        | Paxton    | M      |          3 |
| 376249 | Brad        | Pitt      | M      |          3 |
| 378578 | Stevo       | Polyi     | M      |          3 |
| 812916 | Uma         | Thurman   | F      |          3 |
+--------+-------------+-----------+--------+------------+
```

Wanna Practice?

- Use Marty's CSE 154 Query Tester

http://www.martystepp.com/query/?username=cse154&password=cse154

It is slow and/or a bit flaky