Relational Databases

Learning Objectives

- Understand what a relational database is
- Understand how keys are used in relational databases

Material

Relational Databases

A relational database is a database organised into tables (“relations”). Each row in the table represents an individual entity and each column an attribute of the entity:

```
movies
| id | title                                | release_date |
|----|--------------------------------------|--------------|
| 1  | The Grand Budapest Hotel             | 2014         |
| 2  | Harry Potter and the Goblet of Fire  | 2005         |
| 3  | Frozen                               | 2013         |
```

In the above `movies` table, the `id` column is the primary key. **A primary key uniquely identifies a row in the table:** no two rows can share a primary key. In theory, a primary key can be any column (or combination of columns) that uniquely identifies a row; however, in practice, an extra `id` column is almost always used.

Representing Relationships

Let’s say we want our database to also store the cast of each movie. We should first create an `actors` table to store the information for individual actors.

```
actors
| id | first_name | surname   | date_of_birth |
|----|------------|-----------|---------------|
| 1  | Kristen    | Bell      | 1980          |
| 2  | Daniel     | Radcliffe | 1989          |
| 3  | Ralph      | Fiennes   | 1962          |
```

We then can create a `roles` table to represent roles played by a particular actors in particular movies:

```
roles
| id | character   | actor_id | movie_id |
|----|-------------|----------|----------|
| 1  | Voldemort   | 3        | 2        |
| 2  | M. Gustave  | 3        | 1        |
| 3  | Anna        | 1        | 3        |
```

Notice how `actor_id` corresponds to the primary key in the actors table and `movie_id` the primary key in the movies table. This is how we link tables together in a relational database. **When a primary key appears in another table, it is called a foreign key.** Here, `actor_id` and `movie_id` are foreign keys.

You might wonder why we couldn't just put all this information in the movies table, rather than create two additional tables. We could've created a cast column and put the actor roles data in there, perhaps in JSON format. However, a best-practice rule of relational databases is that **each cell should only contain a single piece of information**, not a list or nested structure.

We could also have just put the actor information directly in the roles table, however this would create duplicate data because actors appear in multiple movies. If we then wanted to update an actor's surname, we would have to remember to update all the rows they appear in. If your database contains duplicate data, this is also a sign that it has been improperly constructed.

The structure of a database - its tables and their columns - is referred to as its "schema".

Core Assignments

Cinema Schema

Imagine you're a developer working on the online booking system for a chain of cinemas. Design a database schema for this application. Your database should include the following tables:

- Cinemas: their location and number of screens.
- Movies: their title and duration.
- Screenings: the movie, cinema, start time and screen number.

For each table, list the columns needed and clearly indicate the primary and foreign keys (you will need to add these in addition to the columns above).

Extension Assignments

Cinema Schema++

Expand your cinema schema so it also contains a screens and bookings table. The screens table should indicate the number of seats each cinema’s screen has. The bookings table should store a customer’s email, seat number and the screening they’re going to view.