Exercises

1. Introduction

For these exercises, you can work in groups of two.

You will be using the MovieLens 100K Dataset, a dataset containing 100,000 movie ratings from the MovieLens website. You will create an entity-relationship model for the data, load the dataset in a SQL-server (PostgreSQL) and a NoSQL document store (MongoDB) and perform some queries on the database.

First, download the 100K dataset (don't forget to validate the download with the checksum). You will only need the following files from the archive:

- u.data
- u.item
- u.genre
- u.user
- u.occupation

2. Entity-relationship model

Create a data model for this data set. First you will need to devise a simple ER model for the data. What are the entities, attributes and relations?

- Identify the strong and weak entities in the model.
- Identify the relations between the entities and determine their cardinality.
- Identify the attributes of the entities and relations.
- Determine the candidate keys and choose the primary keys for the entities in your model.
- Specialise and/or generalise entities where possible.

Hand in a diagram of your model (there are several good online tools for this. For example: https://www.draw.io).

Based on this model. Convert the ER model into relations (tables) according to the 8 steps:

1. For each strong entity create a table with columns for each simple attribute. The key attribute becomes the primary key.
2. For each weak entity create a table with columns for each simple attribute and include columns for the primary keys of those entities on which the entity depends (foreign keys).
3. When two entities are in a one-to-many relationship, the entity with the many cardinality must have a foreign key representing this relationship.
4. When two entities are in a one-to-one relationship, a foreign key must be included in one of the two.
5. When two entities are in a many-to-many relationship a table must be created consisting of foreign keys for the two entities.
6. When an entity has a multi-valued attribute, create a table with a column as foreign key to the entity and a column for the multi-valued attribute
7. When more than two entities participate in a relationship then a table must be created consisting of foreign keys to those entities.
8. When a subtyping is defined by attributes create separate tables for each subtype consisting of those attributes which are peculiar to the subtype.

Finally check your database model and try to see if any further normalization is required.

As a reference for ER modelling please see tutorialspoint section on the ER model. This site has an excellent section on normalization as well.

3. Implementation in PostgreSQL

3.1 Installation

Install the PostgreSQL database on your machine. Download the software directly from the official website or use the package from your Linux distribution.

After installation, you will need to configure PostgreSQL and create a database for use in this exercise.

3.2 Loading the data

Load the data in the database according to the relations (tables) you derived from the E/R model. You will first need to use the create table commands to create the necessary tables. Do not forget to correctly create the foreign keys and other constraints.

Next, some preprocessing of the raw data might be needed before you can insert the data. Use the text manipulation tools you learned about to convert the data in to appropriate input (e.g. csv). You can then use the copy command to import data.

Here you can find for a full reference of all PostgreSQL SQL commands and syntax.

Create a project in your version control system for these exercises and commit all conversion scripts and SQL-scripts to your version control system.

3.3 Simple queries

Once the data is imported you can use the SQL data manipulation commands to browse your database. Try some simple queries first and then write and perform SQL queries that answer the following questions:

- Which movie has received the most ratings?
- What is the average rating for the movie 'Star Wars'?
- What is the occupation and age of the user that has given the most ratings?

Extra: Based on the queries above, and the general schema of the database, can you identify any columns that will need indexes?

4. MongoDB

4.1 Installation

Install MongoDB on your machine. Download the software directly from the official website or use the package from your Linux distribution.

4.2 Loading the data

Store the data in a single collection where every document contains a rating. The user and movie information is stored in subdocuments. For example, here is the json document of a single rating:

```
{
    "movie": {
        "title": "Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963)",
        "release_date": "01-Jan-1963",
        "genres": [
            "Sci-Fi",
            "War"
        ],
    },
    "rating": 5,
    "time stamp": 888625200,
    "user": {
        "age": 24,
        "gender": "M",
        "occupation": "student",
        "zip": "41850"
    }
}
```

First convert the data to a single json file with this structure using your favorite text processing tool (Python, Awk, etc.). If you don't know how to approach this, use the following hint:

Ernq va gur h.hfre naq h.vgrz qngn naq fgber gurfr va gjb ybbxhc gnoyrf (unfuznc, qvpgvbanel). Gura ernq gur h.qngn svyr naq hfr gur ybbxhc gnoyrf gb bhgchg gur qrabeznyvmrq qngn.

If you are still stuck after this, use our Python `movielens2mongo.py` script in this repository.

4.3 Simple queries

Repeat the queries from 3.3, but on the MongoDB database using its own query language.

5. Optional: more complex queries

If you want to practice some more advanced queries, try answering the following questions:

- How many different genres does a movie have on average?
- What is the male/female distribution over both the userbase and the ratings?
- What is the most controversial movie (has the highest variance in its ratings)?

6. Related material

If you have time, or are interested, we have provided some extra references for extra reading and/or watching. In no particular order:

- Schema on read vs. schema on write: A story about George
- Composing and Scaling Data Platforms
- The Essence of Databases
- DBMS2 blog
- Guide to ACID and transactions
- Locking basics
- Visual explanation of SQL Joins
- DBMS tutorial
- MongoDB is Web Scale (video)
- Why you should never use MongoDB
- MongoDB is growing up
- Introduction to NoSQL (video)
- Don't Settle for Eventual Consistency