Introduction to Big Data: Homework 1

Dongxu Han

1.

title.akas.tsv - this file describes details of localized title/tv in different regions.

Content	Description
titleId	unique id of title
ordering	unique id of each title in different regions
title	a localized title
region	display region of this version, represented by acronym of countries
language	language of this version, usually accords with region
types	display type of this version, like dvd, tv, imdbDisplay, etc.
attributes	more description of this version
isOriginalTitle	whether it's original title. 0 means not original; 1 means original

name.basics.tsv - describe basic information of persons, including name, profession, etc.

Content	Description
nconst	unique id of the person
primaryName	name of the person
birthYear	birth year of the person
deathYear	death year of the person, may be null because being alive
primaryProfession	profession of the person
knownForTitles	titles this person is known for

title.basics.tsv - describe basic information of titles, including type, genre, etc.

Content	Description
tconst	unique id of a given title

titleType	type of a given title
primary-Title	the most used title
isAdult	adult restriction. 0-not adult; 1-adult
startYear	year this tile starts
endYear	year this title ends
runtime-Minutes	how much minutes this title runs
genres	genre of this title, like documentary, animation, etc.

title.crew.tsv - describe directing and writing relationship of between titles and persons.

Content	Description
tconst	unique id of a given title
directors	ids indicating the person who direct this title
writers	id indicating the person who write this title

title.episode.tsv - describe episode information of titles and parenting relationship among titles.

Content	Description
tconst	unique id of a given title
parent-Tconst	id indicating the parent title of a given title
season-Number	number of season which this title's episode belongs to
episode-Number	number of episode of this title

title.principals.tsv - describe principal relationship between titles and persons and details of principals.

Content	Description
tconst	unique id of a given title
ordering	a unique number of the principal in a given title
nconst	id indicating the person who is one of principals of a given title
category	the category of job the person is in for a given title
job	specific job the person does, may be "$\backslash n$"

characters	name of the role this person plays, may be "\n"

title.ratings.tsv - describe rating and voting information of titles.

Content	Description
tconst	unique id of a given title
average-Rating	average rating of a given title
numVotes	number of received votes

2.
ER diagram is represented in Figure 1.

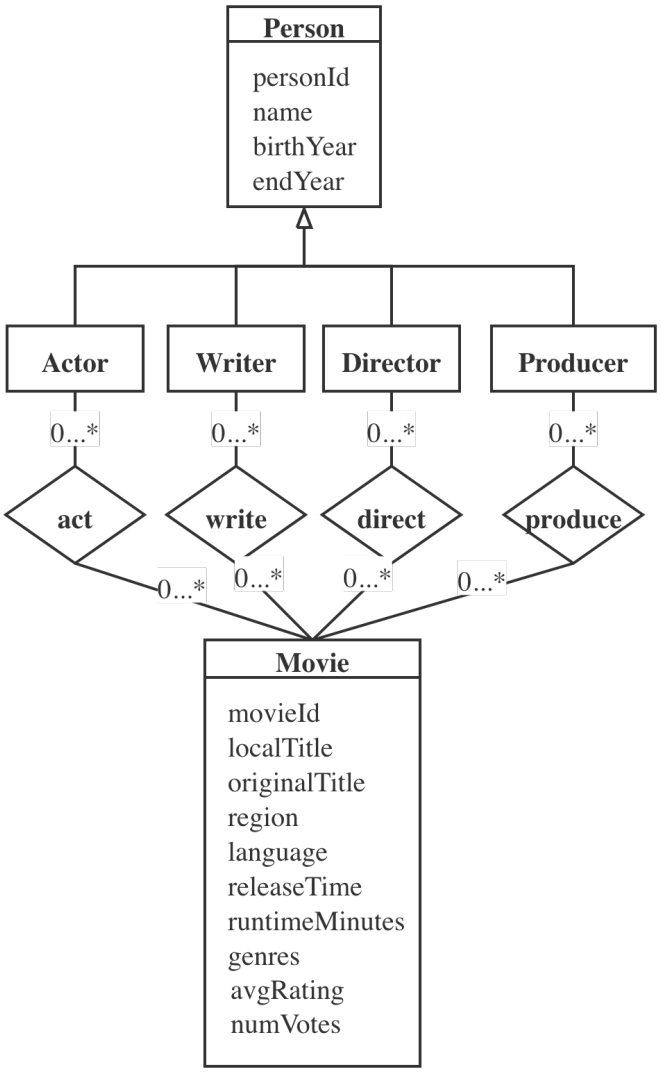

Figure 1: Entity-Relationship diagram.

3.
The **relational model** is represented in **Figure 2**.
To use integer instead of strings for primary keys, as for an alphanumeric id, I

store the numeric part of it in database. When querying, we join the alphabetic head with it according to the circumstance.
Consider "tt000001". Split "tt000001" to "tt" and "000001" and store "000001" only. If we want to export the data in database, we may then join "tt" with the six-digit number.

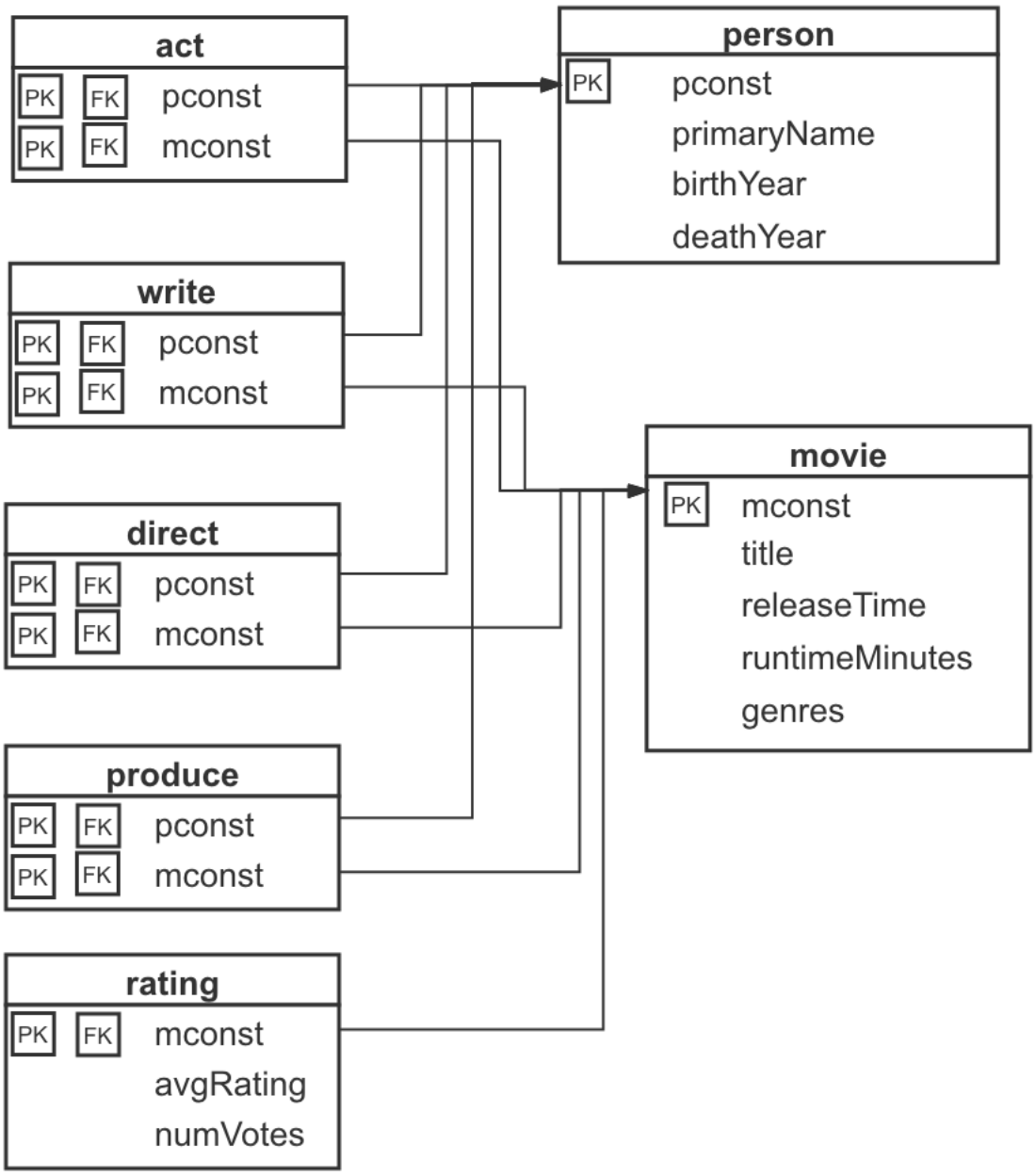

Figure 2: Relational model.

Sql script to create a database for imdb is in file imdb_script.sql.

4.
As for my program, I try to create multiple threads to read different file and insert data into database simultaneously. As shown in relational model, I use a

lot of "conceptual" foreign keys. The word "conceptual" means I don't physically use it while creating database and loading data. I admit that foreign key could help keep integrity of data. However, since database would make a check on foreign key when doing operations, this would reduce the performance of loading data into database. This doesn't mean I don't take care of integrity of data. It's just a strategy to improve performance by sacrificing other aspects. Besides, it is actually developer-friendly with no foreign key. We are able to check data integrity in our program as well, with no dependence on database itself.
Time reporting:
Thread loads ratings within 5.46 minutes.
Thread loads write and direct within 28.38 minutes.
Thread loads movie within 30.52 minutes.
Thread loads person within 38.37 minutes.
Thread loads act and produce within 103.35 minutes.
Finish loading all data within 103.36 minutes.

Program source code: Imdb.java.

5.
Part of the program:
Code below would generate a duplicate key error

```
String sql1 ="insert into person values(9993720, 'dongxu', 1997, null);";
String sql2 ="insert into person values(9993720, 'error', 1909, 1992);";
```

Code below would catch the error in sql2. And take a rollback on sql1.

```
try {
    stmt.execute(sql1);
    System.out.println("sql1 executed.");
    // sql2 would generate an error and this transaction would rollback
    stmt.execute(sql2);
    stmt.execute(sql3);
    con.commit();
} catch (Exception e) {
    System.out.println("Exception occur: " + e.getMessage());
    if (con != null) {
        con.rollback();
```

```
12              System.out.println("Rollback occurs.");
13          }
14      }
```

Program source code: RollBackTest.java.