The Movie Database

Ellen Munthe-Kaas
February 2008

The following is a slightly shorter version in English of a document written in conjunction with the mandatory exercises in INF3100.

1 Background

The test database to be used in the mandatory exercises is a version of the Internet Movie Database (imdb) [1], which is a large database containing information on approximately 700000 films and 60000 TV series, 1.7 million persons related to the films, descriptions of the films etc. The database runs on Postgres.

History

A first version containing a subset of the films was realized in 2002 on Sybase ASE 11.9.2. Later the department migrated to Oracle. In 2007 the department decided to migrate to Postgres; in connection with this we have decided to include a more or less full version of imdb. The students have read access to this 2007 version. They are however to make their own copy of the 2002 version. Notice that the schema of the two versions differ somewhat due to changes in the source files available from imdb.
2 Design of the 2002 Version

The database is centered around three entity types – films, persons, and participation which links the first two together. The ORM-UML diagram can be seen in Figure 1. Each concept in an ORM-UML model translates into a table during the realization stage. Not all concepts warrant a table, and in this particular case Country, Language, Rating, Votes, Year, Title, Name, and Gender have been suppressed. The resulting tables are as follows:

AlternativeFilmTitle, Film, FilmCountry, FilmGenre, FilmLanguage, FilmRating, Genre, Participation, ParticipationRange, ParticipationValue, and Person.

The table Rating reflects how the film has been rated in imdb. This is based on people
rating the films (1 to 10) over the Internet. The table Participation represents the participation of persons in films. The participation can be of different types (all types are listed in ParticipationRange). For instance, the following SQL query describes the participation of Luc Besson in the film “The Fifth Element”:

```sql
username=> select * from participation
username=> where personid = 26513 and filmid = 76914;
pid | personid | partname     | filmid
---------+----------+-----------------+-------
2278671 | 26513    | director        | 76914
2749250 | 26513    | writing credits | 76914
(2 rows)
```

`pid` is an artificially created attribute, to facilitate joins between Participation and ParticipationValue. (The attribute is superfluous since `<personId,partName,filmId>` constitutes a candidate key. But it is easier to join on one attribute rather than three, and less information is duplicated in this fashion.)

ParticipationValue contains the values describing participation, e.g. roles played by actors and actresses in a film. The following example shows the role played by Milla Jovovich in the film “The Fifth Element”:

```sql
username=> select * from participationvalue
username=> where pid = 1846849;
pid | value
---------+-------
1846849 | Leeloo
(1 row)
```

The following example shows all the roles played by Milla Jovovich with the corresponding films.

```sql
username=> select pv.value, f.maintitle
username=> from participationvalue pv, participation pa,
username=>   person p, film f
username=> where pv.pid = pa.pid and pa.filmid = f.filmid and
username=>   pa.personid = p.personid and
username=>   p.surName = 'Jovovich' and
username=>   p.firstName = 'Milla';
value       |            maintitle
----------------+-----------------------------------------
Mildred Harris | Chaplin
Lucia          | Claim, The
Leeloo         | Fifth Element, The
Joan of Arc    | Messenger: The Story of Joan of Arc, The
(4 rows)
```

Contribution

The primary force behind the 2002 database was Igor Rafienko. David Ranvig converted imdb files to SQL insert sentences. Rune Aske assisted in testing the imdb mirror.
3 Design of the 2007 Version

Films and TV Series

The 2007 version of the database contains ordinary cinema films as well as TV films and series etc. FilmItem contains all kinds of "films", from ordinary cinema films to episodes in TV series. For each collection of episodes that make up a series, FilmItem in addition contains an item that represents the series as such, and which can be used to find the title of the series.

Figure 2: FilmItem and its subconcepts

FilmItem types:

C ordinary cinema film
V video film
VG video game
TV TV film
TVS TV series
mini mini TV series
E episode in a TV series or a mini TV series. The corresponding occurrence in Series has type TVS or mini.

filmdescription contains some additional information for some of the filmitems.
create table filmitem (  
    filmid int primary key,  
    filmttype varchar(4) not null  
);

cREATE TABLE film (  
    filmid int primary key references filmitem (filmid),  
    title text not null,  
    prodyear int  
);

cREATE INDEX filmtitleindex on film (title);  
cREATE INDEX filmyearindex on film (prodyear);

cREATE TABLE filmdescription (  
    filmid int primary key references filmitem (filmid),  
    year text,  
    filmdescr text,  
    check (year is not null or filmdescr is not null)  
);

cREATE TABLE series (  
    seriesid int primary key references filmitem (filmid),  
    maintitle text not null,  
    firstprodyear int  
);

cREATE INDEX seriesmaintitleindex on series (maintitle);

cREATE TABLE episode (  
    episodeid int primary key references filmitem (filmid),  
    seriesid int not null references filmitem (filmid),  
    subtitle text not null,  
    foreign key (seriesid) references series (seriesid)  
);

Additional Film Information

For some films alternative titles (AlternativeFilmTitle), production country (FilmCountry), and language (FilmLanguage) exist. Some films contain some additional information about the language (FilmLanguageInfo). A film can belong to one or more genres (FilmGenre). RunningTime contains the length of the film when run in different countries, potentially with some additional information (RunningTimeInfo). Imdb allows the audience to vote (points between 1 and 10) for films, this is included in FilmRating. Distribution contains how the votes distribute over the points given. Some films are assigned a rank based on this.

Distribution is a 10-character string.
  1. character: How many voted 1 point (lowest score)
  2.                      .".-  2 points
...
10                      .".-  10 points (best score)

The characters are interpreted as follows:
  "."  no votes
"0"  1-9%
"1"  10-19%
...
"9"  90-99%
"*"  100%
create table alternativefilmtitle (  
  filmid int references filmitem (filmid),  
  akaid int references filmitem (filmid),  
  info text not null  
);  
create index alternativefilmtitlefilmidindex  
on alternativefilmtitle (filmid);  
create index alternativefilmtitleakaidindex  
on alternativefilmtitle (akaid);  
create table filmcountry (  
  filmid int references filmitem (filmid),  
  country text,  
  primary key (filmid, country)  
);  
create index filmcountryfilmidindex on filmcountry (filmid);  
create table country (  
  country text primary key  
);
create table filmlanguage (
    filmid int references filmitem (filmid),
    langid int,
    language text not null,
    primary key (filmid, langid)
);
create index filmlanguagefilmidindex on filmlanguage (filmid);

create table filmlanguageinfo (
    filmid int not null,
    langid int not null,
    info text not null,
    foreign key (filmid, langid) references filmlanguage (filmid, langid)
);
create index filmlanguageinfofilmidlangidindex
    on filmlanguageinfo (filmid, langid);

create table language (
    language text primary key
);

create table filmgenre (
    filmid int references filmitem (filmid),
    genre text,
    primary key (filmid, genre)
);
create index filmgenrefilmidindex on filmgenre (filmid);
create index filmgenregenreindex on filmgenre (genre);

create table genre (
    genre text primary key
);

create table runningtime (
    filmid int references filmitem (filmid),
    timeid int,
    time text,
    country text,
    primary key (filmid, timeid),
    check (time is not null or country is not null)
);
create index runningtimefilmidindex on runningtime (filmid);

create table runningtimeinfo (
    filmid int not null,
    timeid int not null,
    info text not null,
    foreign key (filmid, timeid) references runningtime (filmid, timeid)
);
create index runningtimeinfotimeidindex on runningtimeinfo (filmid,
    timeid);

create table filmrating (
    filmid int primary key references filmitem (filmid),
    votes int not null,
    distribution char(10) not null,
    rank float(3)
);
Persons

All persons have a family name, and mostly also a first name. Gender is only present for some persons. hasFirstName includes first and middle names. In BiographyItem are placed fairly long texts under a number of different codes, e.g.:

RN: Real name
TR: Trade
DB: Date of birth
DD: Date of death
NK: Nickname
BG: Background
BY: Biographer
SP: Spouse
HT: Height
OW: Other work
CV: Curriculum vitae
QU: Quotations

Figure 4: Person

```sql
create table person (
    personid int primary key,
    lastname text not null,
    firstname text,
    gender char(1),
    check (gender = 'M' or gender = 'F');
);
create index person lastname index on person (lastname);
create table biographyitem (personid int references person (personid),
    code char(2),
    description text not null,
    primary key (personid, code));
```
Filmparticipation

Persons can participate in films as cast, composer, costume designer, director, editor, producer, writer. Film participation is contained in FilmParticipation. Some additional information may be found in FilmParticipationInfo.

create table filmparticipation (
    partid int primary key,
    personid int not null references person (personid),
    filmid int not null references filmitem (filmid),
    parttype text not null
);

create index filmparticipationpersonidindex
    on filmparticipation (personid);
create index filmparticipationfilmidindex
    on filmparticipation (filmid);

create table filmcharacter (
    partid int primary key references filmparticipation (partid),
    filmcharacter text,
    billingpos int,
    check (filmcharacter is not null or billingpos is not null)
);
create table filmparticipationinfo (  
    partid int not null references filmparticipation (partid),  
    info text not null  
);

cREATE INDEX filmparticipationinfo_partidindex 
    ON filmparticipationinfo (partid);

Examples
Luc Besson’s participation in the film The Fifth Element:

username=> select x.partid, x.personid, x.filmid, x.parttype  
username=> from filmparticipation x, person p, film f  
username=> where  
username=>   p.lastname = 'Besson' and  
username=>   p.firstname = 'Luc' and  
username=>   f.title = 'Fifth Element, The' and  
username=>   x.personid = p.personid and  
username=>   x.filmid = f.filmid;

<table>
<thead>
<tr>
<th>partid</th>
<th>personid</th>
<th>filmid</th>
<th>parttype</th>
</tr>
</thead>
<tbody>
<tr>
<td>781285</td>
<td>89222</td>
<td>237127</td>
<td>director</td>
</tr>
<tr>
<td>1009544</td>
<td>89222</td>
<td>237127</td>
<td>writer</td>
</tr>
<tr>
<td>1009560</td>
<td>89222</td>
<td>237127</td>
<td>writer</td>
</tr>
<tr>
<td>1009576</td>
<td>89222</td>
<td>665467</td>
<td>writer</td>
</tr>
<tr>
<td>1009592</td>
<td>89222</td>
<td>665467</td>
<td>writer</td>
</tr>
</tbody>
</table>
(5 rows)

Milla Jovovich’s roles in The Fifth Element:

username=> select filmcharacter  
username=> from filmcharacter  
username=> where partid = 19580594 or partid = 19580610;

<table>
<thead>
<tr>
<th>partid</th>
<th>personid</th>
<th>filmid</th>
<th>parttype</th>
</tr>
</thead>
<tbody>
<tr>
<td>19580594</td>
<td>89222</td>
<td>237127</td>
<td>writer</td>
</tr>
<tr>
<td>19580610</td>
<td>89222</td>
<td>665467</td>
<td>writer</td>
</tr>
</tbody>
</table>
(2 rows)

Additional information about these roles:

username=> select partid, info  
username=> from filmparticipationinfo  
username=> where partid = 19580594 or partid = 19580610;

<table>
<thead>
<tr>
<th>partid</th>
<th>info</th>
</tr>
</thead>
<tbody>
<tr>
<td>19580610</td>
<td>voice</td>
</tr>
</tbody>
</table>
(1 row)

The reason why there are two filmids for The Fifth Element, is that there is a VG version of the film:
username=> select f.filmid, title, prodyear, filmtype
username=> from film f, filmitem i
username=> where
username=>   f.title = 'Fifth Element, The' and
username=>   f.filmid = i.filmid;

<table>
<thead>
<tr>
<th>filmid</th>
<th>title</th>
<th>prodyear</th>
<th>filmtype</th>
</tr>
</thead>
<tbody>
<tr>
<td>237127</td>
<td>Fifth Element, The</td>
<td>1997</td>
<td>C</td>
</tr>
<tr>
<td>665467</td>
<td>Fifth Element, The</td>
<td>1998</td>
<td>VG</td>
</tr>
</tbody>
</table>

(2 rows)

username=>

All roles played by Milla Jovovich:

username=> select f.title, c.filmcharacter
username=> from filmparticipation x, person p, film f, filmcharacter c
username=> where
username=>   x.parttype = 'cast' and
username=>   p.lastname = 'Jovovich' and
username=>   p.firstname = 'Milla' and
username=>   x.personid = p.personid and
username=>   x.filmid = f.filmid and
username=>   x.partid = c.partid;

<table>
<thead>
<tr>
<th>title</th>
<th>filmcharacter</th>
</tr>
</thead>
<tbody>
<tr>
<td>.45</td>
<td>Kate</td>
</tr>
<tr>
<td>AFI's 100 Years... 100 Cheers: America's Most Inspiring Movies</td>
<td>Herself</td>
</tr>
<tr>
<td>Cannes: Through the Eyes of the Hunter</td>
<td>Herself</td>
</tr>
<tr>
<td>Chaplin</td>
<td>Mildred Harris</td>
</tr>
<tr>
<td>Claim, The</td>
<td>Lucia</td>
</tr>
<tr>
<td>Corporate Malfeasance</td>
<td>Herself</td>
</tr>
<tr>
<td>Dazed and Confused</td>
<td>Michelle Burroughs</td>
</tr>
<tr>
<td>Dummy</td>
<td>Fangora</td>
</tr>
<tr>
<td>Fifth Element, The</td>
<td>Leeloo</td>
</tr>
<tr>
<td>Fifth Element, The</td>
<td>Leeloo</td>
</tr>
<tr>
<td>Game Babes</td>
<td>Herself</td>
</tr>
<tr>
<td>Game Over: 'Resident Evil' Reanimated</td>
<td>Herself</td>
</tr>
<tr>
<td>He Got Game</td>
<td>Dakota Burns</td>
</tr>
<tr>
<td>House on Turk Street, The</td>
<td>Erin</td>
</tr>
<tr>
<td>Kuffs</td>
<td>Maya Carlton</td>
</tr>
<tr>
<td>Making and Meaning of 'We Are Family', The</td>
<td>Herself</td>
</tr>
<tr>
<td>Messenger: The Story of Joan of Arc, The</td>
<td>Joan of Arc</td>
</tr>
<tr>
<td>Million Dollar Hotel, The</td>
<td>Eloise</td>
</tr>
<tr>
<td>Night Train to Kathmandu, The</td>
<td>Lily McLeod</td>
</tr>
<tr>
<td>Playing Dead: 'Resident Evil' from Game to Screen</td>
<td>Herself</td>
</tr>
<tr>
<td>Resident Evil</td>
<td>Alice</td>
</tr>
<tr>
<td>Resident Evil: Apocalypse</td>
<td>Alice</td>
</tr>
<tr>
<td>Resident Evil: Extinction</td>
<td>Alice</td>
</tr>
<tr>
<td>Return to the Blue Lagoon</td>
<td>Lilli</td>
</tr>
<tr>
<td>Star Element, The</td>
<td>Herself</td>
</tr>
<tr>
<td>Starz on the Set: Ultraviolet</td>
<td>Herself</td>
</tr>
<tr>
<td>Teen Vid II</td>
<td>Herself</td>
</tr>
<tr>
<td>Trailer for a Remake of Gore Vidal's Caligula</td>
<td>Druscilla</td>
</tr>
<tr>
<td>Two Moon Junction</td>
<td>Samantha Delongpre</td>
</tr>
<tr>
<td>Ultraviolet</td>
<td>Violet</td>
</tr>
<tr>
<td>VH1/Vogue Fashion Awards</td>
<td>Herself</td>
</tr>
<tr>
<td>You Stupid Man</td>
<td>Nadine</td>
</tr>
<tr>
<td>Zoolander</td>
<td>Katinka</td>
</tr>
</tbody>
</table>

(33 rows)

username=>

45

API's 100 Years... 100 Cheers: America's Most Inspiring Movies
Cannes: Through the Eyes of the Hunter
Chaplin
Claim, The
Corporate Malfeasance
Dazed and Confused
Dummy
Fifth Element, The
Fifth Element, The
Game Babes
Game Over: 'Resident Evil' Reanimated
He Got Game
House on Turk Street, The
Kuffs
Making and Meaning of 'We Are Family', The
Messenger: The Story of Joan of Arc, The
Million Dollar Hotel, The
Night Train to Kathmandu, The
Playing Dead: 'Resident Evil' from Game to Screen
Resident Evil
Resident Evil: Apocalypse
Resident Evil: Extinction
Return to the Blue Lagoon
Star Element, The
Starz on the Set: Ultraviolet
Teen Vid II
Trailer for a Remake of Gore Vidal's Caligula
Two Moon Junction
Ultraviolet
VH1/Vogue Fashion Awards
You Stupid Man
Zoolander

(33 rows)
These are only her cinema films. Milla Jovovich has played in TV series too:

```sql
username=> select s.maintitle, e.subtitle, c.filmcharacter
username-> from filmparticipation x, person p,
username->   episode e, filmcharacter c, series s
username-> where
username->   x.parttype = 'cast' and
username->   p.lastname = 'Jovovich' and
username->   p.firstname = 'Milla' and
username->   x.personid = p.personid and
username->   x.filmid = e.episodeid and
username->   x.partid = c.partid and
username->   e.seriesid = s.seriesid;
```

<table>
<thead>
<tr>
<th>maintitle</th>
<th>subtitle</th>
<th>filmcharacter</th>
</tr>
</thead>
<tbody>
<tr>
<td>4Pop</td>
<td>Nuo surkeat Hollywood-pelit (#3.6)</td>
<td>Herself</td>
</tr>
<tr>
<td>Harald Schmidt Show, Die</td>
<td>(2002-03-19)</td>
<td>Herself</td>
</tr>
<tr>
<td>HBO First Look</td>
<td>The Messenger: The Story of Joan of Arc</td>
<td>Herself</td>
</tr>
<tr>
<td>HypaSpace</td>
<td>(#5.39)</td>
<td>Herself</td>
</tr>
<tr>
<td>HypaSpace</td>
<td>(#5.40)</td>
<td>Herself</td>
</tr>
<tr>
<td>HypaSpace</td>
<td>(#5.42)</td>
<td>Herself</td>
</tr>
<tr>
<td>HypaSpace</td>
<td>(#5.44)</td>
<td>Herself</td>
</tr>
<tr>
<td>HypaSpace</td>
<td>(#5.45)</td>
<td>Herself</td>
</tr>
<tr>
<td>King of the Hill</td>
<td>Get Your Freak Off (#7.1)</td>
<td>Serena</td>
</tr>
<tr>
<td>Late Late Show with Craig Ferguson, The</td>
<td>(#2.104)</td>
<td>Herself</td>
</tr>
<tr>
<td>Married with Children</td>
<td>Fair Exchange (#4.6)</td>
<td>Yvette</td>
</tr>
<tr>
<td>Paradise</td>
<td>Childhood's End (#1.8)</td>
<td>Katie</td>
</tr>
<tr>
<td>Parker Lewis Can't Lose</td>
<td>Pilot (#1.1)</td>
<td>Robin Pecknowitz</td>
</tr>
<tr>
<td>Tout le monde en parle</td>
<td>(2002-03-23)</td>
<td>Herself</td>
</tr>
<tr>
<td>V Graham Norton</td>
<td>(#1.47)</td>
<td>Herself</td>
</tr>
</tbody>
</table>

(16 rows)

```sql
username=>
```
4 Practicalities about Postgres

Students in INF1300, INF3100, INF5100 get passwords for use on the postgres server kurspg. Login from unix-systems to postgres via the psql interface when on the Ifi network:

```bash
> psql -h kurspg -U username -d dbnavn
```

where `username` is the ordinary user name. Each user has a database (working area) with the same name as the user name. The command

```bash
> psql -h kurspg -U username -d username
```

gives access to the working space. Here each can create his/her own database schema.

**For first time users:** When you get the prompt “=>” the first time, i.e. when you have logged onto postgres, change your password immediately:

```sql
=> alter role username with encrypted password 'newpassword';
```

Do not forget the semicolon (all sql commands must be terminated with semicolon). If forgotten, you get the prompt “->”, i.e. psql believes that the command continues on the next line.

Alternatively, prepare your command in a text file, e.g. `filename.sql`, and run it with the command

```bash
=> \i filename.sql
```

Notice that psql-specific commands are *not* terminated with semicolon.

Exit psql with the command

```bash
=> \q
```

**Postgres documentation:** [http://www.postgresql.org/docs/8.2/interactive/](http://www.postgresql.org/docs/8.2/interactive/)

For more about psql, see [http://www.postgresql.org/docs/8.2/interactive/app-psql.html](http://www.postgresql.org/docs/8.2/interactive/app-psql.html)

**Information about kurspg from windows:** use an X-server, e.g. X-win32, and start psql from this.

**Information about kurspg from outside Ifi’s network:** The simplest way is to use ssh to Ifi’s login-cluster (login.ifi.uio.no) and then use psql. Use for instance putty, available from the Ifi dvd.

**The 2002 Database**

You shall install the 2002 database yourselves. Log in on

```bash
> psql -h kurspg -U username -d username
```
and run

```sql
=> \i ~inf1300/www_docs/imdb/imdb2002/install.sql
```

You can even install postgres on your laptop and install the 2002 database there.

**The 2007 Database**
This database is so large that we do not allow you to make your own version of it. You can access a read-only version by

```bash
> psql -h kurspg -U username -d fdb
```

**5 References**
