Database Systems, spring 2014
Mini Project

Elias Obeid
eobeid11@student.aau.dk

Kent Caspersen
kcaspe11@student.aau.dk

Martin Madsen
mbma11@student.aau.dk

d601f14, 1.1.01
6th February, 2014 to 1st April, 2014

1 Self Study 1: Preliminary Database Modeling

Deadline: Wednesday 12th February, 2014

As stated in the assignment, we have decided to look at different possible attributes and models by looking at the structure of movie pages on IMDB. Initially, we think it would require many join tables, as we've identified a few many-to-many relationships among structures we've discussed. These structures are: *actors*, *directors*, *writers*, *movies*, *awards*, *ratings*, and *users*.

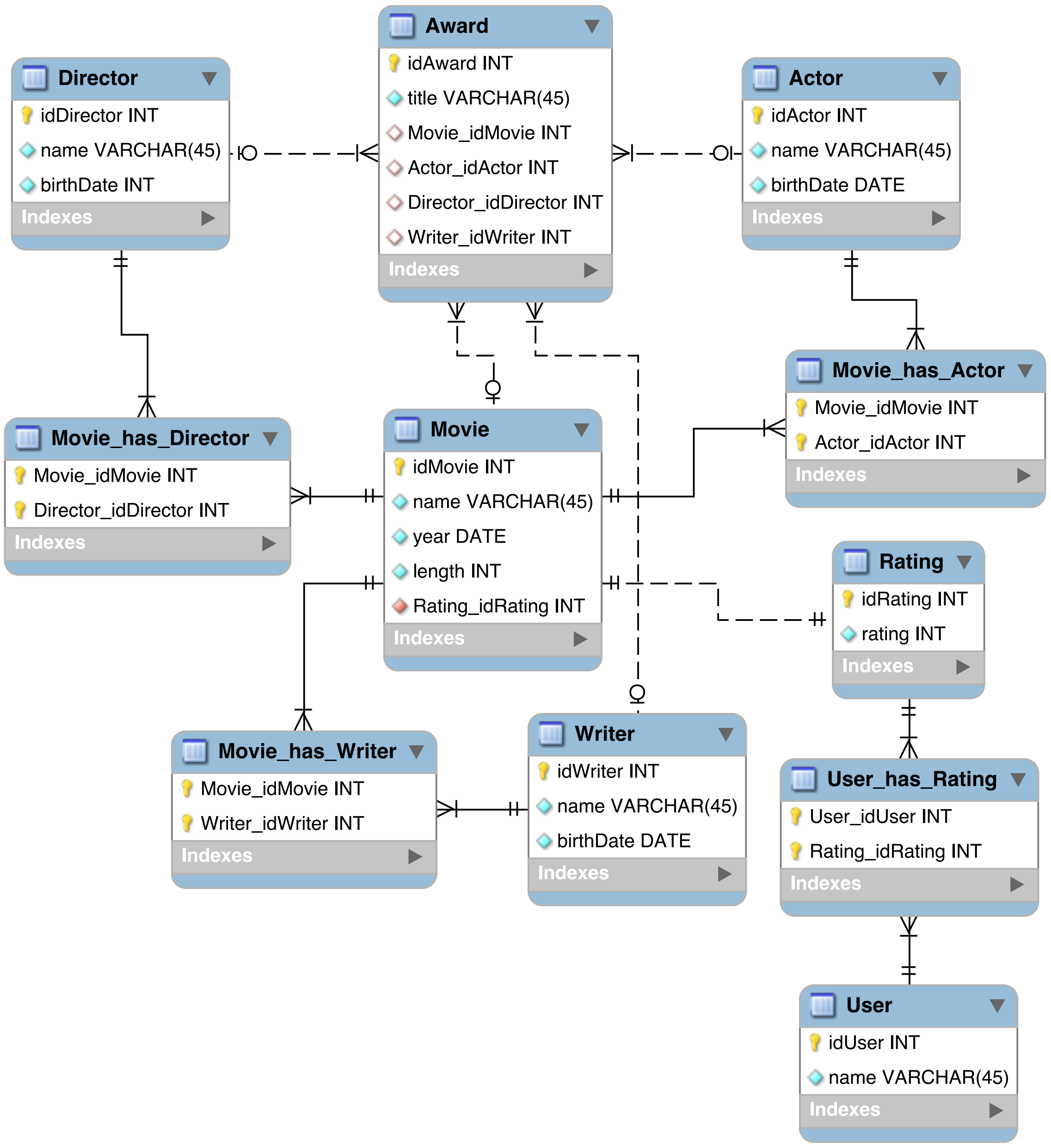

Figure 1: *Enhanced entity-relationship (EER) model diagram of a simplified movie database.*

We spent time on figuring out how to map the relationships between tables instead of focusing on the attributes. In our opinion it is easy to just add a birthdate if that should be necessary.

Figure 1 shows relationships between the chosen models and their corresponding join tables. Dashed lines between tables repressent *non-identifying* relationships and solid lines between tables represent *identifying* relationships.

When lines branch toward a table then there is a “has many” relationship to that table. When the lines have two orthogonal dashes (or a orthogonal dash and a circle) by a table then there is a “has one” relationship to that table. If there is a circle then the relationship is non-identifying. For example one *Director* has many *Awards*. The relationship is also non-identifying because the tables can exist indenpendently of each other.

2 Self Study 2: Database Modeling

Deadline: Wednesday 12^{th} March, 2014

Entity-relationship Diagram

In figure 2, we show an updated ER diagram based on concepts we’ve learned in the course.

Primary keys are underlined. Chen, min max, and arrows on lines represent the different cardinalities between entities and their relations. Circles are attributes and squares represent entities. Diamonds are relationships, just like we have learned in the course.

Schema

The entities and relationships have been mapped to relations in the diagram of figure 3. Attributes acting as foreign keys in relation A are marked by an ASCII arrow `->`, where the arrow points to the primary key(s) in relation B. The symbols to the left of each attribute signal whether the attribute can be null or not. When black, they cannot take on the null value, when hollow, the attribute can be null, like the *dateOfDeath* attribute on the *Person* relation, since we cannot know when living actors/directors will die.

Non-trivial considerations

The *Participate* relationship is 3-way due to the fact that many *People* (actors) can have many different roles in different movies, or even multiple roles in one movie. This relationship construct allows us to express both in the database.

Comparison of the previous and current solution

In our first attempt to construct a diagram for the movie database, we used the Enhanced Entity-Relationship (EER) model to construct the relevant information for the database. In this version of our database, we use the Entity-Relationship (ER) model as described in the course.

In this version we include Chen notation and min-max notation to emphasize to type of relations. This is also visualised in form of arrows or no arrows on each connection

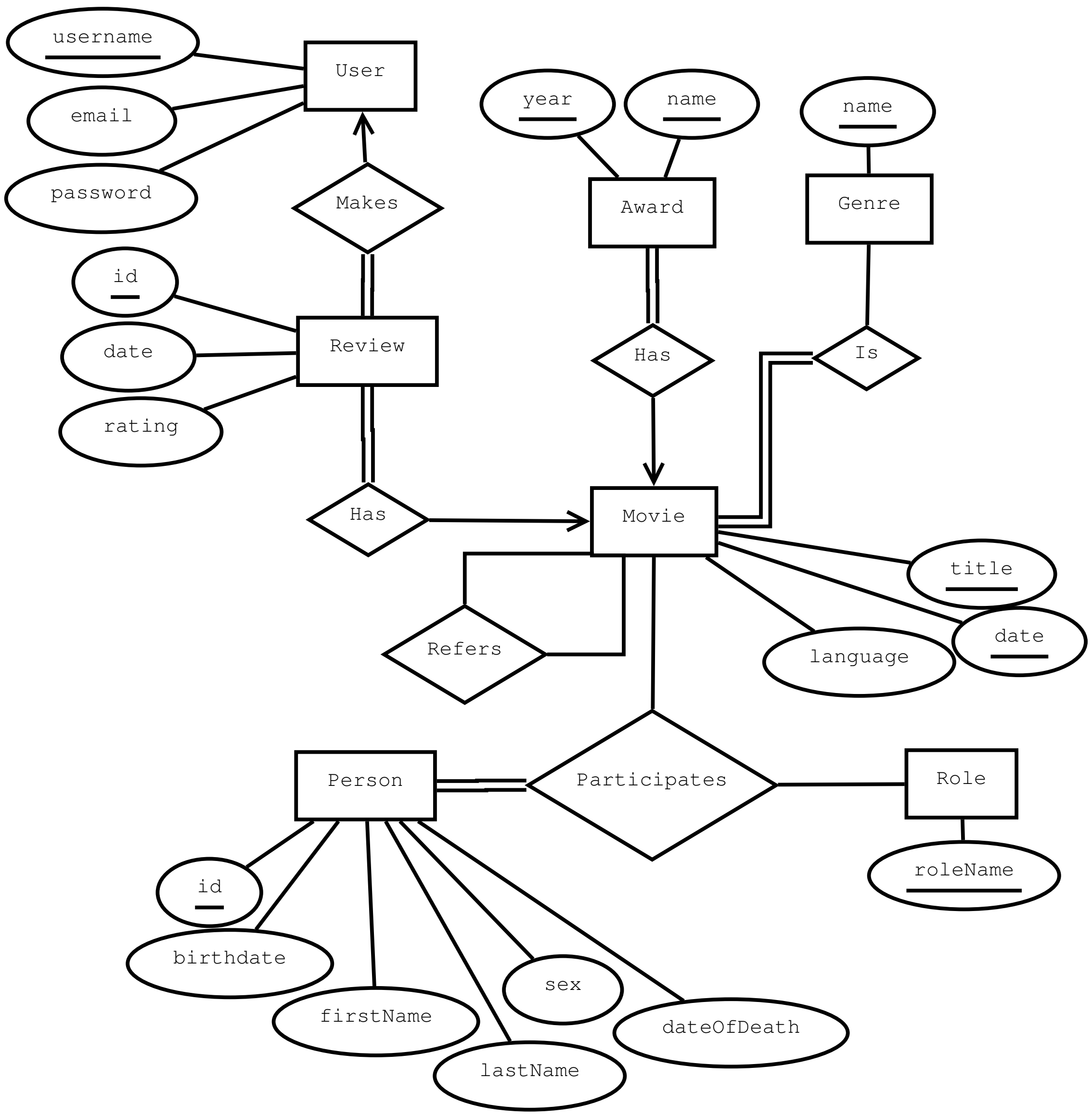

Figure 2: *ER diagram of an IMDB-like website.*

between relations. Another clear difference is that we include total participation for some of the relations. Actually, total and partial participation is expressed as identifying and non-identifying relations in the EER model. This is not covered in the course. We could also have included weak entities, but we did not find any which should be marked as weak.

We have removed redundancy, because an actor can also be a director in movies. We have introduced a relation called Role in which it is clear which role a person has in a given movie. We also considered an ISA relation between the roles in a movie. We chose not to use it, because there is nothing different between an actor and a director.

We have only included the necessary primary keys. If there was no need for a unique id, we have not included one.

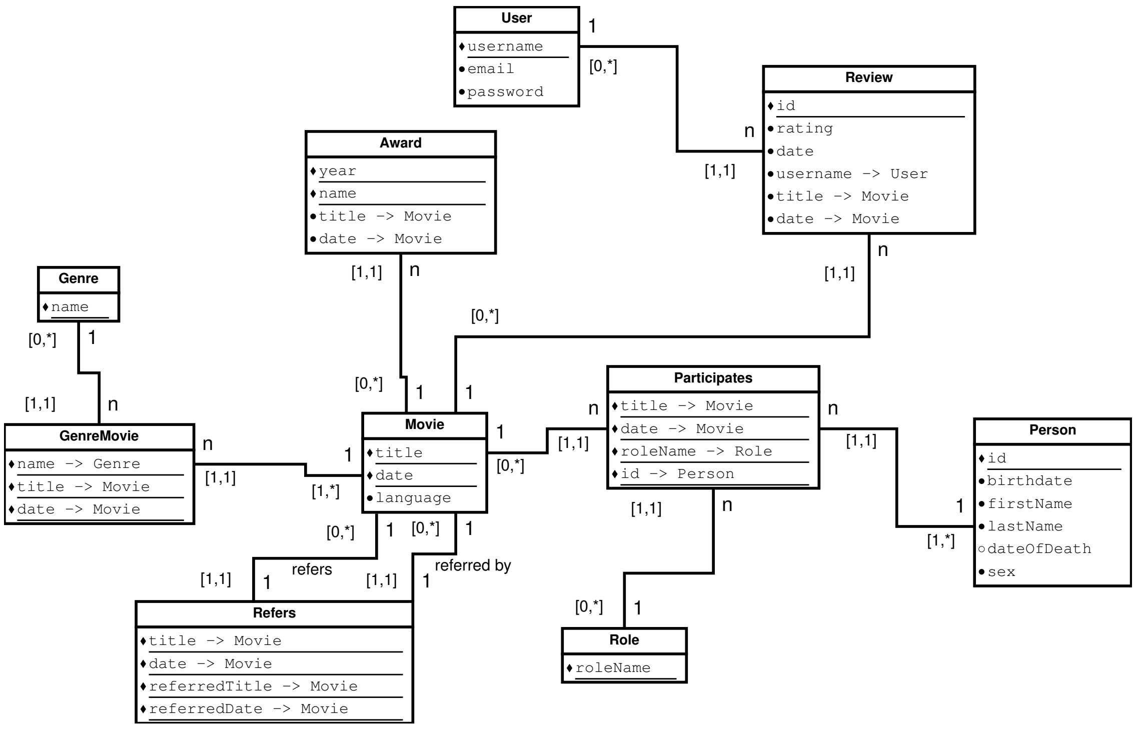

Figure 3: *The schema, i.e. mapped relations of an IMDB-like website.*

3 Self Study 3: Exam Preparation 1

Exercise 1: ER Modeling

The ER diagram for the database about borrowing books from the university's library as shown in figure 4.

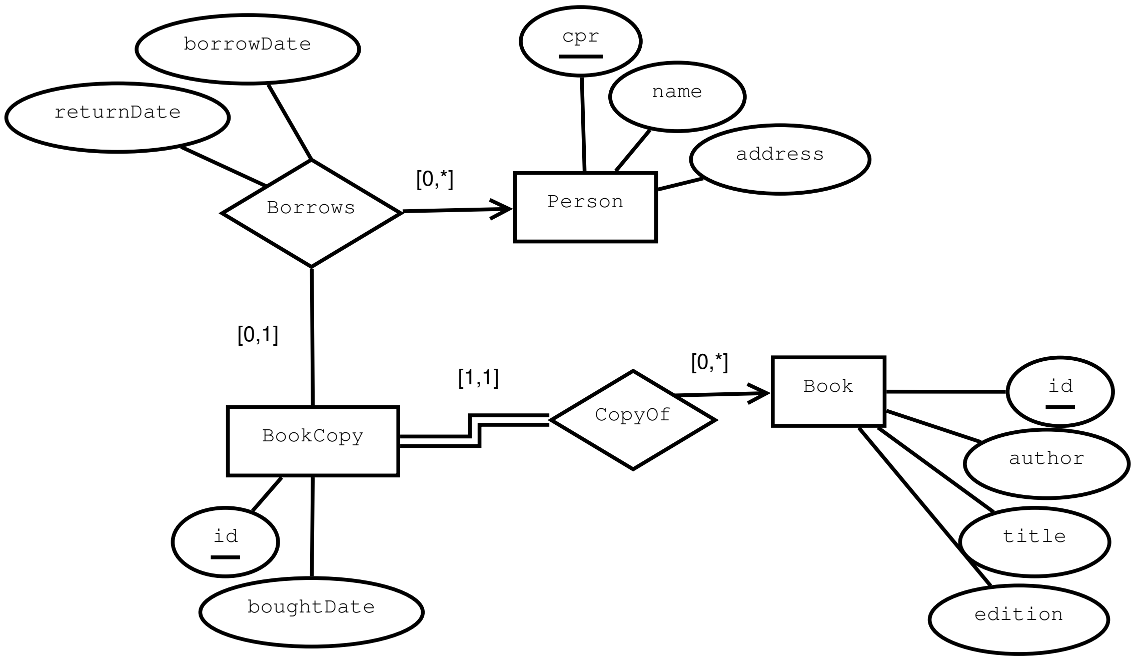

Figure 4: *ER diagram of a library.*

Exercise 2: Banking System

The ER diagram of a banking schema has been transformed to a relational diagram as shown below in figure 5.

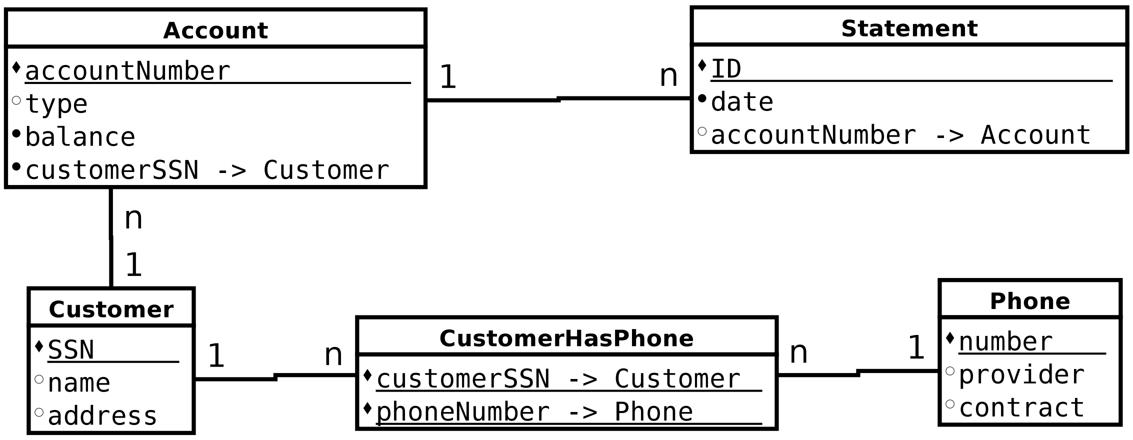

Figure 5: *Banking system schema.*

Exercise 3: Relational Algebra

1

In the first relational algebra expression we begin by selecting all entries in the `zoos` table, where `country` = ′`Germany`′. We then project the relation onto `zooId`. Before we do the division, we project the `animals` relation onto `species` and `zooId`. Finally, we divide the two tables and end up with:

species
giraffe
ape
owl

2

We start by renaming the `animals` table to two tables called `T1` and `T2`. We then do a theta-join on the two new tables, with `T1.zooId` = `T2.zooId` as a constraint. Now we have a large table with `T1` and `T2` side by side, where the constraint is maintained. We now do a selection from the joined table with `T1.animalId` = `T2.father` ∨ `T1.animalId` = `T2.mother`. This results in a table with rows where the current animal is either the father or mother of another animal, which is in the same zoo. Finally, we do a projection of the nickname of `T1`, and get the following result:

nickname
Wohoo
Huhuu
Eule

Excercise 4: Relational Calculus

The following queries are presented in the following combination:

1. Relation algebra
2. Tuple relational algebra
3. Domain relational algebra

1. Find the names of suppliers who supply some red part

$$\pi_{\texttt{sname}}(\texttt{Suppliers} \bowtie (\texttt{Catalog} \bowtie \sigma_{\texttt{color=red}}(\texttt{Parts})))$$

$$\{s.\texttt{sname} \mid s \in \texttt{Suppliers} \wedge \exists c \in \texttt{Catalog}(s.\texttt{sid} = c.\texttt{sid} \wedge \exists p \in \texttt{Parts}(c.\texttt{pid} = p.\texttt{pid} \wedge p.\texttt{color} = \texttt{red}))\}$$

$$\{\langle b \rangle \mid \exists a, c\ (\langle a, b, c \rangle \in \texttt{Suppliers} \wedge \exists i, j(\langle b, i, j \rangle \in \texttt{Catalog} \wedge \exists y, z(\langle i, y, z \rangle \in \texttt{Parts} \wedge z = \texttt{red})))$$

2. Find the sids of suppliers who supply some red or green part

$$\pi_{\texttt{sid}}(\texttt{Catalog} \bowtie \sigma_{\texttt{color=red} \vee \texttt{color=green}}(\texttt{Parts}))$$

$$\{c.\texttt{sid} \mid c \in \texttt{Catalog} \wedge \exists p \in \texttt{Parts}(c.\texttt{pid} = p.\texttt{pid} \wedge p.\texttt{color} = \texttt{red} \vee p.\texttt{color} = \texttt{green}))\}$$

$$\{\langle b \rangle \mid \exists i, j(\langle b, i, j \rangle \in \texttt{Catalog} \wedge \exists y, z(\langle i, y, z \rangle \in \texttt{Parts} \wedge (z = \texttt{red} \vee z = \texttt{green})))$$

3. Find the sids of suppliers who supply some red part and some green part

$$\pi_{\texttt{sid}}(\texttt{Catalog} \bowtie \sigma_{\texttt{color=red}}(\texttt{Parts})) \bowtie \pi_{\texttt{sid}}(\texttt{Catalog} \bowtie \sigma_{\texttt{color=green}}(\texttt{Parts}))$$

$$\{c_1.\texttt{sid} \mid c_1 \in \texttt{Catalog} \wedge \exists c_2 \in \texttt{Catalog}(c_1.\texttt{sid} = c_2.\texttt{sid} \wedge \exists p_1, p_2 \in \texttt{Parts}(c_1.\texttt{pid} = p_1.\texttt{pid} \wedge c_2.\texttt{pid} = p_2.\texttt{pid} \wedge p_1.\texttt{color} = \texttt{red} \wedge p_2.\texttt{color} = \texttt{green}))\}$$

$$\{\langle b \rangle \mid \exists i, j, k, l(\langle b, i, k \rangle \in \texttt{Catalog} \wedge \langle b, j, l \rangle \in \texttt{Catalog} \wedge \exists u, v, x, y(\langle i, u, v \rangle \in \texttt{Parts} \wedge \langle i, x, y \rangle \in \texttt{Parts} \wedge v = \texttt{red} \wedge y = \texttt{green}))))$$

4. Find pairs of sids such taht the supplier with the first sid charges more for some pat than the supplier with the second sid

$$\pi_{\text{c1.sid, c2.sid}}\ ((\rho_{\text{c1}}\ (\texttt{Catalog}) \times \rho_{\text{c2}}\ (\texttt{Catalog})) \bowtie_{\text{c1.pid = p1.pid}}\ \wedge$$
$$\text{c2.pid}\ =\ \text{p2.pid}\ (\rho_{\text{p1}}\ (\texttt{Parts}) \bowtie_{\text{p1.cost > p2.cost}}\ \rho_{\text{p2}}\ (\texttt{Parts})))$$

$$\{s_1.\texttt{sid}, s_2.\texttt{sid} \mid \exists c_1, c_2 \in \texttt{Catalog}(c_1.\texttt{sid} = s_1.\texttt{sid}\ \wedge\ c_2.\texttt{sid} = s_2.\texttt{sid} \wedge$$
$$\exists p_1, p_2 \in \texttt{Parts}(c_1.\texttt{pid} = p_1.\texttt{pid}\ \wedge\ c_2.\texttt{pid} = p_2.\texttt{pid} \wedge$$
$$p_1.\texttt{cost} > p_2.\texttt{cost}))\}$$

$$\{\langle\ \text{sid, sid'}\ \rangle \mid \exists\ \text{pid, cost}(\langle\ \text{sid, pid, cost}\ \rangle \in \texttt{Catalog} \wedge$$
$$\exists\ \text{pid, cost}\ (\langle\ \text{sid', pid, cost'}\ \rangle \in \texttt{Catalog} \wedge$$
$$(\ \text{cost}\ >\ \text{cost'}\ ) \wedge (\ \text{sid}\ \neq\ \text{sid'}\ )))\}$$

5. Find the pids of parts supplied by at least two different suppliers

$$\pi_{\text{c1.pid}}\ (\sigma_{\text{c1.sid} \neq \text{c2.sid}}\ (\rho_{\text{c2}}\ (\texttt{Catalog}) \bowtie_{\text{c1.pid = c2.pid}}\ \rho_{\text{c1}}\ (\texttt{Catalog})))$$

$$\{\ \text{c1.pid}\ \mid\ \text{c1}\ \in \texttt{Catalog} \wedge \exists\ \text{c2}\ \in \texttt{Catalog}(\ \text{c1.pid}\ =\ \text{c2.pid} \wedge$$
$$\text{c1.sid}\ \neq\ \text{c2.sid}\ )\}$$

$$\{\langle\ \text{pid}\ \rangle \mid \exists\ \text{sid},\ \text{cost}\ (\langle\ \text{sid},\ \text{pid},\ \text{cost}\rangle \in \texttt{Catalog} \wedge$$
$$\exists\ \text{sid'},\ \text{cost'}\ (\langle\ \text{sid'},\ \text{pid},\ \text{cost'}\ \rangle \in \texttt{Catalog} \wedge$$
$$\text{sid}\ \neq\ \text{sid'}))\}$$

Exercise 5: Functional Dependencies

FD	OK or violated?
$A \to C$	violated: tuples 3, 4
$B \to A$	OK
$C \to A$	violated: tuples 1, 3 and 2, 4
$A \to B$	violated: tuples 1, 2
$B \to C$	violated: tuples 3, 4
$BC \to A$	OK
$AC \to B$	OK

4 Self Study 4: Refinement, normalization, and SQL-DDL

Deadline: Monday 24th March, 2014 This document is formatted as closely as possible to the list of requirements for the report in the self study 4 exercises.

Revised ER diagram

We have merely added Chen and [min,max] notation.

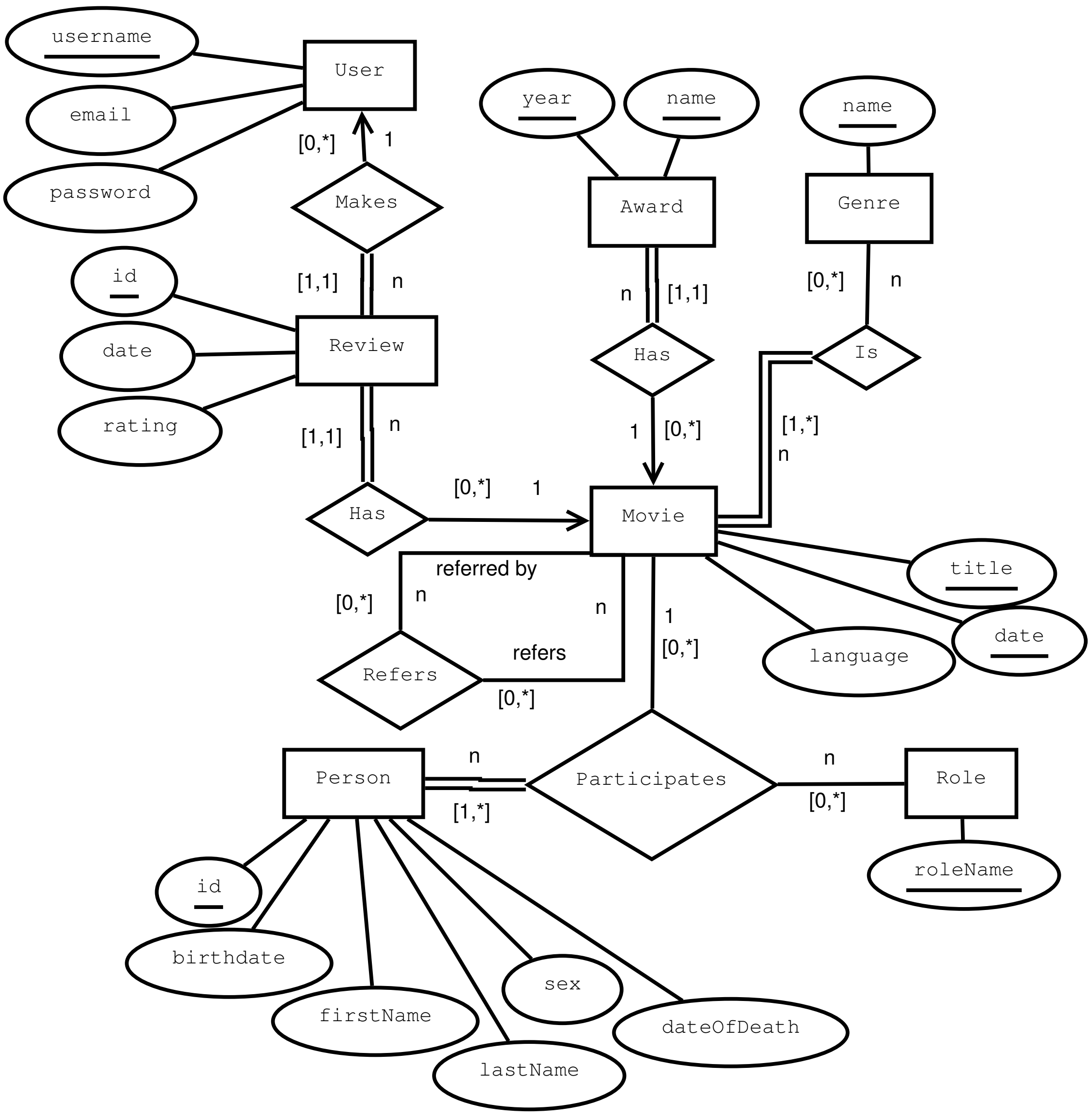

Figure 6: *Revised ER diagram*

Functional dependencies

- User: username → email, password
- Award: year, name → title, date

- Movie: title, date → language
- Person: id → birthdate, firstName, lastName, dateOfDeath, sex
- Review: id → rating, date, username, title, date

List of normalized relations

Definitions

This is also the derived relational schema. We simply derived thee following schema below by taking the mapped table and formalizing it. No algorithms were used.

- User: $\{[\underline{username : string}, email : string, password : string]\}$
- Award: $\{[\underline{year : integer, name : string}, title : string \rightarrow Movie, date : string \rightarrow Movie]\}$
- Genre: $\{[\underline{name : string}]\}$
- GenreMovie: $\{[\underline{name : string \rightarrow Genre, title : string \rightarrow Movie, date : date \rightarrow Movie}]\}$
- Movie: $\{[\underline{title : string, date : date}, language]\}$
- Refers: $\{[\underline{title : string \rightarrow Movie, date : date \rightarrow Movie, referredTitle : string \rightarrow Movie, referredD}$
- Role: $\{[\underline{roleName : string}]\}$
- Participates: $\{[\underline{title : string \rightarrow Movie, date : date \rightarrow Movie, roleName : string \rightarrow Role, id : inte}$
- Person: $\{[\underline{id : integer}, birthdate : date, firstName : string, lastName : string, dateOfDeath :$ $date, sex : string]\}$
- Review: $\{[\underline{id : integer}, rating : integer, date : date, username : string \rightarrow User, title : string \rightarrow Movie, date : date \rightarrow Movie]\}$

Normalization

We have not made any changes since all our relations are already on BCNF. It is obvious, that the relations with no functional dependencies are all on BCNF, that is: **Participates, Role, Refers, GenreMovie, Genre**.

For the remaining relations, User, Award, Movie, Person and Review, we say that they are all clearly on 1NF, since every attribute is atomic. For 2NF, every non-prime attribute must be fully functional dependent on each candidate key. For the User relation, every candidate key contains only a single attribute, and every non-prime attribute is thus fully functionally dependent on every candidate key. For the remaining relations, no subset of any candidate key exists such that a non-prime attribute is functionally dependent thereof. For 3NF, we need one of these 3 conditions to hold for each functional dependency $A \rightarrow B$:

1. $A \rightarrow B$ is trivial
2. A is a superkey
3. B is part of a candidate key

For all functional dependencies $A \rightarrow B$, we have that A is a superkey, thus we conclude all our relations are on 3NF. For BCNF, all functional dependencies must meet one of the first two conditions just mentioned. Since they all meet condition 2, we now conclude that all of our relations are on BCNF.

SQL for creating tables

Below are SQL statements for creating each of the tables in our mapped relation.

```
CREATE TABLE user(
   username varchar(50),
   email varchar(50) UNIQUE NOT NULL,
   password varchar(50) NOT NULL,
  PRIMARY KEY(username)
);

CREATE TABLE award(
   year int,
   name varchar(50),
   title varchar(50) NOT NULL,
   date date NOT NULL,
  PRIMARY KEY(year, name)
);

CREATE TABLE genre(
   name varchar(50),
  PRIMARY KEY(name)
);

CREATE TABLE genreMovie(
   name varchar(50),
   title varchar(50),
   date date,
  PRIMARY KEY(name, title, date),
  FOREIGN KEY(name) REFERENCES genre(name),
  FOREIGN KEY(title, date) REFERENCES movie(title, date)
);

CREATE TABLE movie(
   title varchar(50),
   date date,
   language varchar(50),
  PRIMARY KEY(title, date),
);

CREATE TABLE refers(
   title varchar(50),
   date date,
   referredTitle varchar(50),
   referredDate date,
  PRIMARY KEY(title, date, referredTitle, referredDate),
  FOREIGN KEY(title, date) REFERENCES movie(title, date),
  FOREIGN KEY(referredTtitle, referredDate)
    REFERENCES movie(title, date)
);
```

```
CREATE TABLE role(
   roleName varchar(50),
  PRIMARY KEY(roleName)
);

CREATE TABLE participates(
   title varchar(50),
   date date,
   roleName varchar(50),
   id int,
  PRIMARY KEY(title, date, roleName, id),
  FOREIGN KEY(title, date) REFERENCES movie(title, date),
  FOREIGN KEY(roleName) REFERENCES role(roleName),
  FOREIGN KEY(id) REFERENCES person(id)
);

CREATE TABLE person(
   id int,
   birthdate date NOT NULL,
   firstName varchar(50) NOT NULL,
   lastName varchar(50) NOT NULL,
   dateOfDeath date NOT NULL,
   sex char(1) NOT NULL,
  PRIMARY KEY(id)
);

CREATE TABLE review(
   id int,
   rating int,
   dateOfReview date NOT NULL,
   username varchar(50) NOT NULL,
   title varchar(50) NOT NULL,
   date date NOT NULL,
  PRIMARY KEY(id),
  FOREIGN KEY(username) REFERENCES user(username),
  FOREIGN KEY(title, date) REFERENCES movie(title, date)
);
```

Reflections

After having completed self study 4, we can conclude that overall we believe we have a strong schema and grasp of creating tables that reduce redundancy and have the tools and skills to prove this.

After having learned the correct, standard notation, we can now pass this on to another developer to implement the database we have designed.

Differences

In self study 2, we fixed the relationships to reduce dependencies and express the fact that one actor (participant) can have multiple roles in a single movie.

5 Self Study 5: Mini-project part 4

This self study was done on Monday 31st March, 2014. We first converted the IMDB MySQL dump to PosgresSQL and then mapped all the tables to our schema, before finally designing the queries described in the exercise document. The PostgreSQL dump of our converted schema can be found at this link: `https://bitbucket.org/Obeyed/p6-project/src/72274ff3487ef59dfd2782cb3a277775b179654c/DBS/5-31.03.14/ourimdb.sql?at=master` with filename `ourimdb.sql`.

Filling the database

We use a Ruby gem (`https://github.com/maxlapshin/mysql2postgres`) to convert the MySQL dump of the IMDB database found on Moodle to a PostgreSQL database with the same tables and data. To ease transfer, we added the tables defined in fig. 3 of self study 2 in the same database and executed the `INSERT INTO` queries below to map the data from IMDB to our tables.

```
INSERT INTO our_movie
SELECT title, language, year
FROM movie

INSERT INTO our_genre
SELECT DISTINCT genre
FROM genre

INSERT INTO "our_genreMovie"
SELECT genre.genre, movie.title, movie.year
FROM genre, movie
WHERE genre."movieId" = movie.id

INSERT INTO our_refers
SELECT m1.title, m2.title, m1.year, m2.year
FROM movieref, movie m1, movie m2
WHERE movieref."fromId" = m1.id AND movieref."toId" = m2.id

INSERT INTO our_person
SELECT id, birthdate, name, '', deathdate, gender
FROM person

INSERT INTO our_role
SELECT DISTINCT role
FROM involved

INSERT INTO our_participates
SELECT movie.title, involved.role, involved."personId", movie.year
```

Count
419

Table 1: *Results of query 1*

```
FROM involved , movie
WHERE involved."movieId" = movie.id

INSERT INTO our_participates
SELECT movie.title , involved.role , involved."personId", movie.year
FROM involved , movie
WHERE involved."movieId" = movie.id

INSERT INTO our_user
SELECT DISTINCT "user", null , null
FROM ratings

INSERT INTO our_review
SELECT rating , "user", movie.title , null , movie.year
FROM ratings , movie
WHERE movie.id = ratings."movieId"
```

No data was present in the dump to identify awards given to movies.

Sometimes we have decided to drop the “date” attribute, rename it to “year” and change its type to int, causing the order of attributes to be rearranged.

There was no information about gender in the database, making it impossible to do any queries based on gender.

Upon mapping of the ‘ratings’ table in the IMDB database, we noticed out that our structure (as shown in figure 3) can uniquely identifiy ratings (reviews) merely by the user’s primary key: username, and the movie’s primary keys: title and date. This restricts the user in only being able to have one review per movie, which is what we wish and allows us to remove the “id” column.

Querying

1. How many Danish language movies are in the database?

Query can be seen below and result is illustrated in table 1.

```
SELECT COUNT(*) FROM movie WHERE language = 'Danish'
```

2. For each year, what is the total number of reviews to movies from that year?

Query can be seen below and result is illustrated in table 2.

```
SELECT movie.year , COUNT(movie.year) FROM review , movie
WHERE review.title = movie.title AND review.year = movie.year
GROUP BY movie.year
```

movie.year	COUNT(movie.year)
2010	23
2000	24
2006	27
1962	1
2003	27
2012	17
2002	19
1993	4
2001	28
1999	48

Table 2: *Results of query 2*

m.title	m.year
Entertainment Tonight	1981
Late Show with David Letterman	1993
The Tonight Show with Jay Leno	1992
E! True Hollywood Story	1996
Live with Regis and Kathie Lee	1988
The Charlie Rose Show	1991
The Rosie O'Donnell Show	1996
Ellen: The Ellen DeGeneres Show	2003
Be Cool	2005
Pulp Fiction	1994

Table 3: *Results of query 3*

3. Which movies have John Travolta and Uma Thurman starred in together?

Query can be seen below and result is illustrated in table 3.

```
SELECT m.title , m.year
FROM movie m, person pe1 , person pe2 , participates pa1 ,
     participates pa2
WHERE pa1.title = m.title AND pa1.year = m.year AND
      pa2.title = m.title AND pa2.year = m.year AND
      pe1.id = pa1.id AND pe2.id = pa2.id AND pe1.id != pe2.id AND
      pe1.name = 'John␣Travolta' AND
      pe2.name = 'Uma␣Thurman' AND pa1.rolename = 'actor'
     AND pa2.rolename = 'actor'
```

4. How many actors and directors have a first name starting with "Q"?

Query can be seen below and result is illustrated in table 4.

```
SELECT pa.rolename , COUNT(*) FROM person pe , participates pa
```

pa.rolename	COUNT(*)
"actor	2100
"director	45

Table 4: *Results of query 4*

COUNT(*)
34

Table 5: *Results of query 5*

```
WHERE pe.id = pa.id AND pa.rolename IN ('actor', 'director') AND
        pe.name LIKE 'Q%'
GROUP BY pa.rolename
```

5. How many users rated at least 3 movies?

Query can be seen below and result is illustrated in table 5.

```
SELECT COUNT(*) FROM
 (SELECT r1.username FROM review r1, review r2, review r3
WHERE r1.username = r2.username AND r1.username = r3.username
     AND r1.title != r2.title AND r2.title != r3.title
     AND r1.title != r3.title
GROUP BY r1.username) AS "total"
```

6. What is the name and birth year of all actors in "Pulp Fiction"? List the actors in increasing order of birth year.

Query can be seen below and result is illustrated in table 6.

```
SELECT p.name, p.birthdate FROM participates pp,
          person p WHERE pp.title = 'Pulp␣Fiction' AND pp.id = p.id
          AND pp.rolename = 'actor' ORDER BY p.birthdate ASC
```

7. What are the titles and years of all movies from the 1980s that John Travolta starred in?

Query can be seen below and result is illustrated in table 7.

```
SELECT pa.title, pa.year FROM participates pa, person p
WHERE pa.year >= 1980 AND pa.year < 1990
        AND p.name = 'John␣Travolta' AND pa.id = p.id
        AND pa.rolename = 'actor'
```

8. What are the top-2 highest rated movies (average) from the 1990s according to the users?

Query can be seen below and result is illustrated in table 8.

p.name	p.birthdate
Emil Sitka	1914-12-22
Harvey Keitel	1939-05-13
Rene Beard	1941-06-03
Christopher Walken	1943-03-31
Joseph Pilato	1949-03-16
Brenda Hillhouse	1953-12-11
John Travolta	1954-02-18
Bruce Willis	1955-03-19
Amanda Plummer	1957-03-23
Lawrence Bender	1957-10-17

Table 6: *Results of query 6*

pa.title	pa.year
Entertainment Tonight	1981
Live with Regis and Kathie Lee	1988
Biography	1987
Look Who's Talking	1989
Wetten, dass…?	1981
Larry King Live	1985
Two of a Kind	1983
Staying Alive	1983
That's Dancing!	1985
Urban Cowboy	1980

Table 7: *Results of query 7*

title	year	avgrate
The Usual Suspects	1995	9.33
The Shawshank Redemption	1994	9.13

Table 8: *Results of query 8*

```
SELECT title, year, AVG(rating) AS avgrate
FROM review WHERE year >= 1990 AND year < 2000 GROUP BY title, year
ORDER BY avgrate DESC LIMIT 2
```

9. What are the top-2 highest rated movies (average) from the 1990s according to at least 2 users?

Query can be seen below and result is illustrated in table 9.

```
SELECT r1.title, r1.year, AVG(r1.rating) AS avgrate
FROM review r1, review r2 WHERE r1.year >= 1990 AND
      r1.year < 2000 AND r1.username != r2.username
GROUP BY r1.title, r1.year
```

title	year	avgrate
The Usual Suspects	1995	9.35
The Shawshank Redemption	1994	9.12

Table 9: *Results of query 9*

AVG(r.rating)	m.language
7.00	""
9.00	French
8.30	English

Table 10: *Results of query 10*

r.title	r.year
Pulp Fiction	1994

Table 11: *Results of query 12*

```
ORDER BY avgrate DESC LIMIT 2
```

10. In 1994, what was the average rating of a movie for each language?

Query can be seen below and result is illustrated in table 10.

```
SELECT AVG(r.rating), m.language
FROM movie m, review r
WHERE m.title = r.title AND m.year = r.year AND m.year = 1994
GROUP BY m.language
```

11. Which actors in Pulp Fiction have never, before or after, starred in the same movie as one of the other actors in "Pulp Fiction"?

We did not complete this query, because it required a massive, advanced query.

12. Which movie starring John Travolta has the highest user ratings?

Query can be seen below and result is illustrated in table 11.

```
SELECT r.title, r.year FROM review r, person p, participates pa
WHERE p.name = 'John␣Travolta' AND p.id = pa.id AND
      r.title = pa.title AND r.year = pa.year
GROUP BY r.title, r.year
ORDER BY AVG (r.rating) DESC
LIMIT 1
```

COUNT(*)
11 707

Table 12: *Results of query 13*

gm.name	AVG(r.rating)
Drama	7.84
Comedy	7.24
Fantasy	7.27
Biography	8.13
Thriller	7.75
Crime	8.39
War	8.27
Musical	7.00
History	8.13
Adventure	7.45

Table 13: *Results of query 14*

13. How many actresses have not been alive at the same time as Charles Chaplin?

Query can be seen below and result is illustrated in table 12.

```
SELECT COUNT(*)
FROM person p1 JOIN person p2
ON (p1.birthdate > p2.dateofdeath OR
    p1.dateofdeath < p2.birthdate) AND
    p2.name = 'Charles Chaplin'
```

14. What is the average rating of movies from each genre?

Query can be seen below and result is illustrated in table 13.

```
SELECT gm.name, AVG(r.rating)
FROM review r, genremovie gm
WHERE gm.title = r.title AND gm.year = r.year
GROUP BY gm.name
```

15. What is the average rating of movies from each genre? List only genres with at least 2 ratings.

Query can be seen below and result is illustrated in table 14.

```
SELECT name, AVG(review.rating)
FROM genremovie, review
WHERE genremovie.title = review.title AND genremovie.year = review.year
GROUP BY name HAVING COUNT(*) >= 2
```

name	AVG(review.rating)
Drama	7.84
Comedy	7.24
Fantasy	7.27
Biography	8.13
Thriller	7.75
Crime	8.39
War	8.27
History	8.13
Adventure	7.45
Sci-Fi	7.47

Table 14: *Results of query 15*

r1.title	ref
Saturday Night Live	38 834

Table 15: *Results of query 16*

COUNT(*)
5930

Table 16: *Results of query 17*

16. Which movie has the largest number of 2-link references?

If A refers to B, and B refers to C, then we say that A has a 2-link reference, through B, to C. If there are several paths leasing from A to C, we count all of them. Query can be seen below and result is illustrated in table 15.

```
SELECT r1.title, COUNT(*) AS ref
FROM refers r1, refers r2
WHERE r1.referredtitle = r2.title AND r1.referredyear = r2.year
GROUP BY r1.title ORDER BY ref DESC LIMIT 1
```

17. How many actors have also been active as director of at least one movie?

Query can be seen below and result is illustrated in table 16.

```
SELECT COUNT(*)
FROM (SELECT DISTINCT p1.id
      FROM participates p1, participates p2
      WHERE p1.id = p2.id AND p1.rolename[U+FFFD]actor[U+FFFD]AND
            p2.rolename [U+FFFD]director[U+FFFD]) AS director_actors
```

g1.name	g2.name	COUNT(*)
Romance	Drama	5837

Table 17: *Results of query 18.*

18. Which two genres are most often linked to the same movie?

Query can be seen below and result is illustrated in table 17.

```
SELECT g1.name, g2.name, COUNT(*) as c
FROM genremovie g1, genremovie g2
WHERE g1.name != g2.name AND g1.title = g2.title AND
        g1.year = g2.year GROUP BY g1.name, g2.name
ORDER BY c DESC
LIMIT 1
```