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 18th March, 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*.

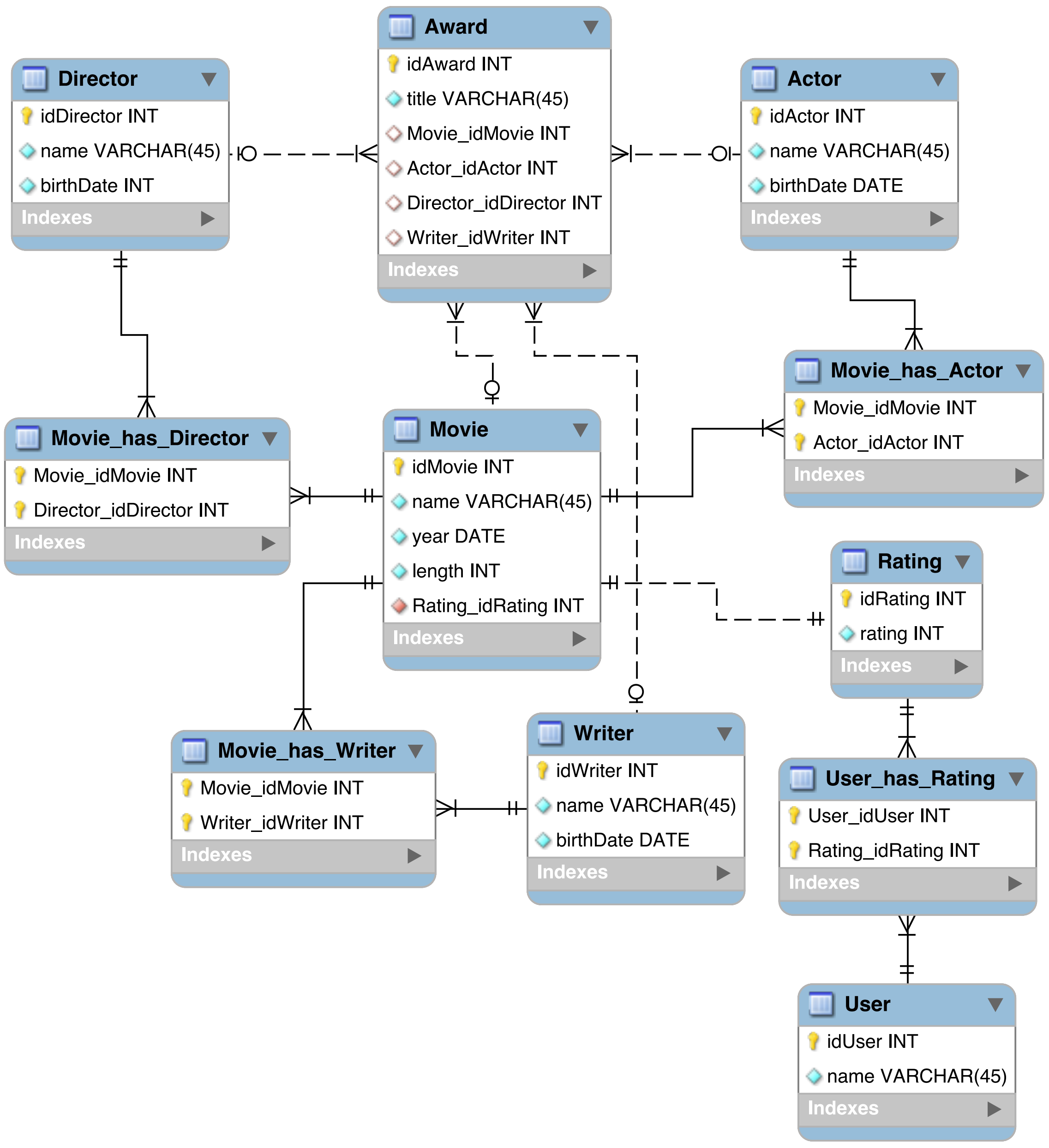

Figure 1: *Enhanced entity-relationship (EER) model diagram of a simplified movie database.*

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 12th 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

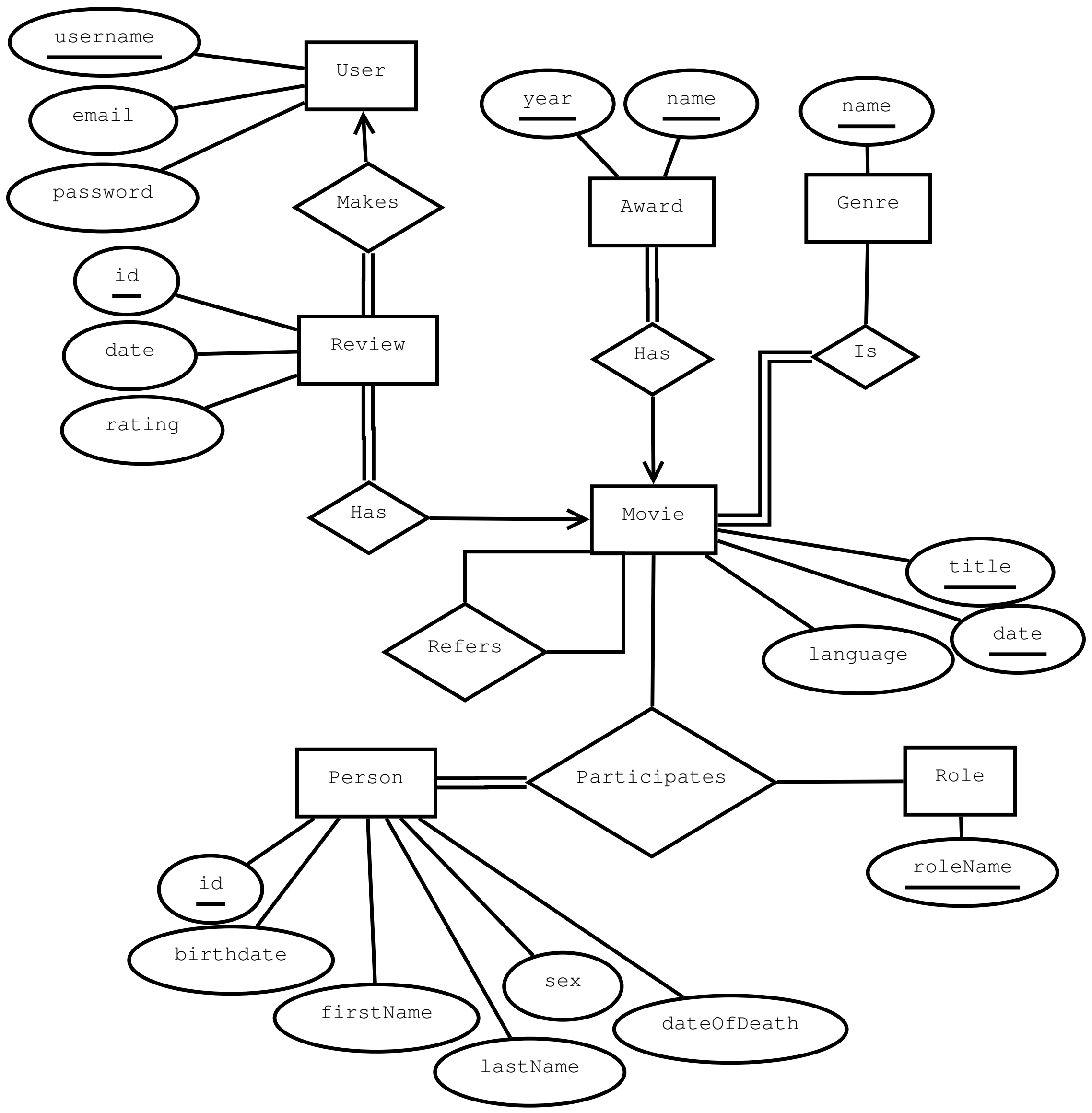

Figure 2: *ER diagram of an IMDB-like website.*

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.

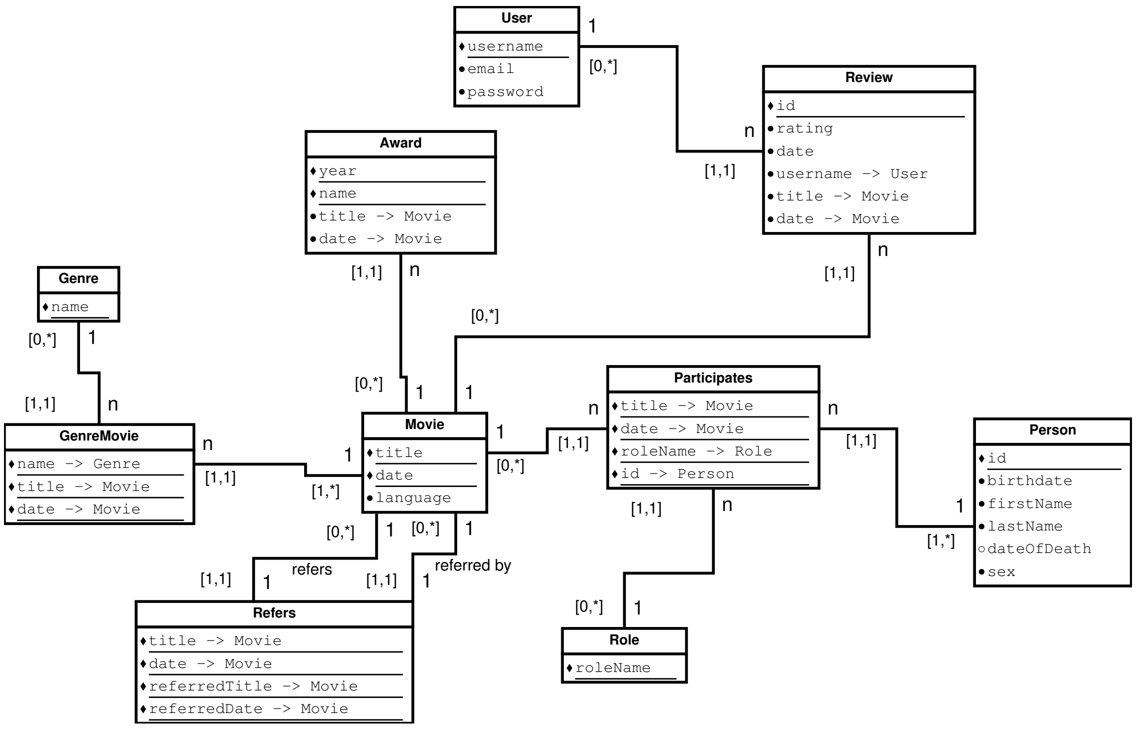

Figure 3: *The schema, i.e. mapped relations of an IMDB-like website.*

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.

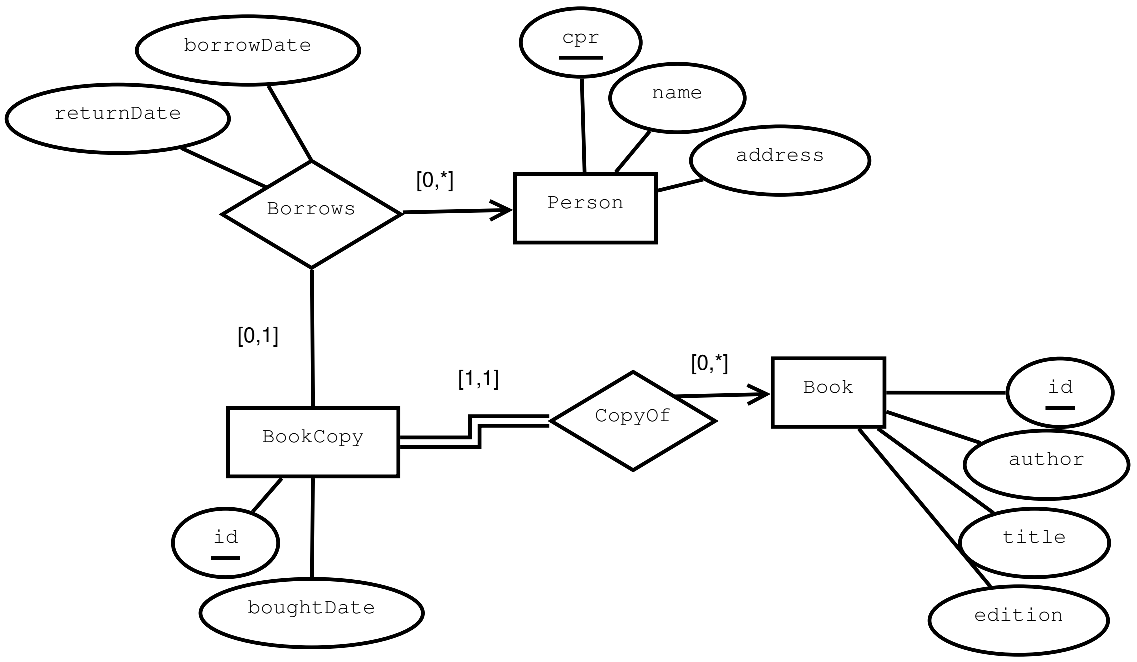

Figure 4: *ER diagram of a library.*

Exercise 2: Banking System

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

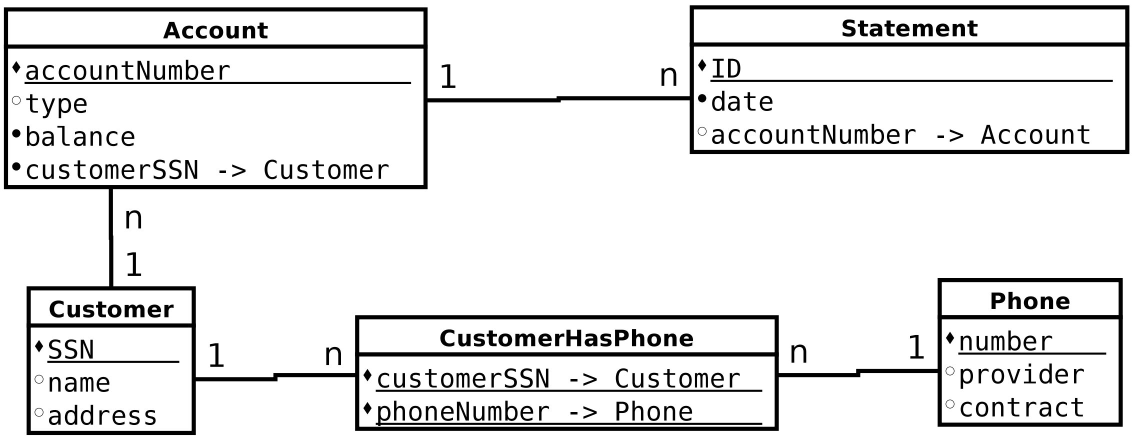

Figure 5: *Banking system schema.*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

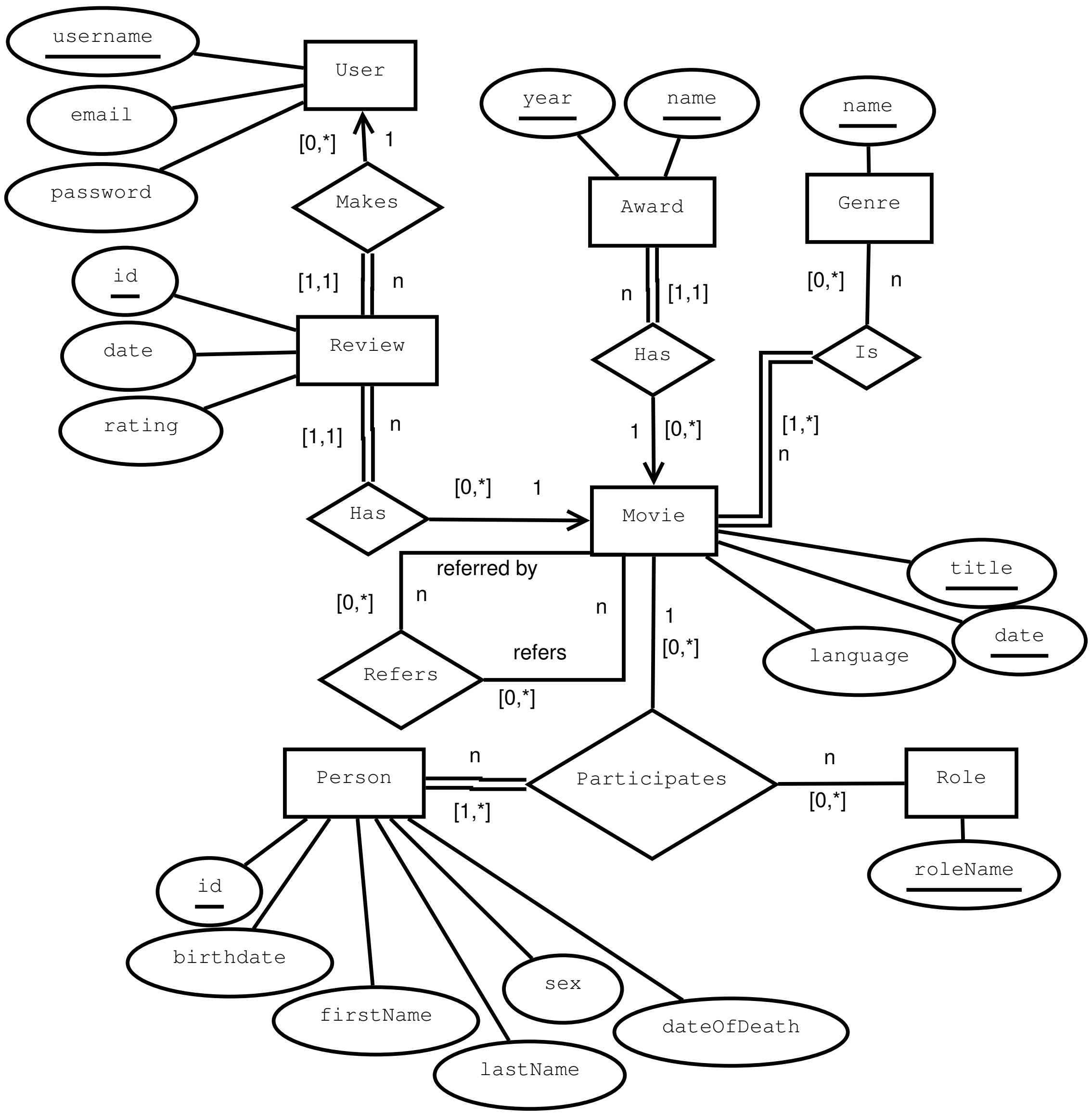

Figure 6: *Revised ER diagram*

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 datetime 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 datetime,
  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 datetime,
   language varchar(50),
  PRIMARY KEY(title, date),
);

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

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

CREATE TABLE participates(
   title varchar(50),
   date datetime,
   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 datetime NOT NULL,
   firstName varchar(50) NOT NULL,
   lastName varchar(50) NOT NULL,
   dateOfDeath datetime NOT NULL,
   sex char(1) NOT NULL,
  PRIMARY KEY(id)
);

CREATE TABLE review(
   id int,
   rating int,
   date datetime NOT NULL,
   username varchar(50) NOT NULL,
   title varchar(50) NOT NULL,
   date datetime NOT NULL,
  PRIMARY KEY(id),
  FOREIGN KEY(username) REFERENCES user(username),
  FOREIGN KEY(title, date) REFERENCES role(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.