2CWK30 - Database Systems

Tom Misson (18008043)
Wahab Rehman (17068255)
Jonathan Sifleet (18014017)

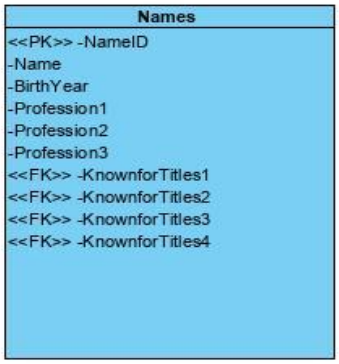
Contents

Preface

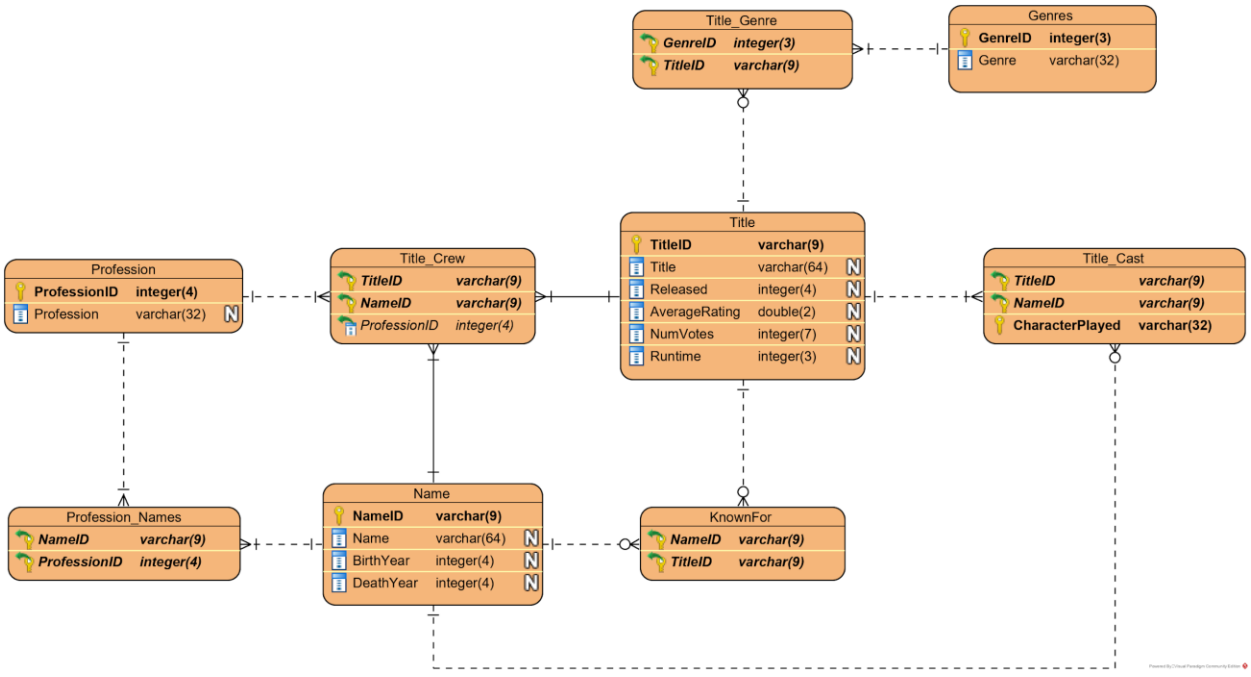
The final submission was written in MySQL and the Database engine we used to migrate the data was MariaDB.
To migrate data, the tables from the original data dumps must exist in your DBMS, and the column names must be identical to those in the original data dumps. The table responsible for names must be called “onames” and the table responsible for titles must be called “otitle” for migrations to complete correctly.

Entity-relationship diagrams

Old system



New design



Assumptions made about the database

We have assumed that Null genres don't matter within the database, aside from that, nothing has been assumed. We have verified situations where people may have played multiple roles etc.

Critique of the current system

Relationships

To begin, the system in its current form can't be converted into a relational database due to the duplication of potential foreign key fields (KNOWNFORTITLES 1-4) and also multiple instances of the same TITLEID in different records. This is the case as in the Titles table, the title id isn't the primary key of the title and therefore would need to be a composite primary key. This causes issues as the composite data can't be paired with the title id (as found in the dump). This ultimately means that while it may not look like it, there is no relationship between the data as there is no one field that uniquely identifies the records.

Associations

While the existing system has no relationships between the entities, there are associations between the KNOWNFORTITLES, there is an association between that field and the TITLEID

and also an association between the NAMEID and the names in the title table. It's a good idea to try and preserve uniquely identifiable data for individual records as eventually, 1 NAMEID will identify 1 person and 1 TITLEID will identify 1 title.

Duplicate attributes

For the names dump, there are multiple profession fields, e.g. PROFESSION1, PROFESSION2. This should be represented as a separate entity and should be a one to many relationships, with one person being able to have many professions. There is also multiple known for fields, e.g. KNOWNFORTITLES1, KNOWNFORTITLES2. This shouldn't be included in the final system as people and films will have relationships between them.

There are also multiple director fields, e.g. DIRECTOR1, DIRECTOR2. This should be represented as a separate entity, with the TITLE_ID attribute as the foreign key. Should be a one to many relationships, with one title entity being able to have multiple directors. The same issue exists with writers, WRITER1, WRITER2. This should be represented as a separate entity, with the TITLE_ID attribute as the foreign key. This too should be a one-to-many relationship.

There are multiple writer fields, e.g. Multiple genre fields, e.g. GENRE1, GENRE2. This should be represented as a separate entity, with the TITLE_ID attribute as the foreign key. This should be a one to many relationships, one title entity and multiple genres. This is an issue though as many titles can cover many genres, therefore, we will need to include a weak entity between the two.

Lastly, there is a duplication of PROFESSION fields which limits the number of roles one person can have. Ideally, we would have a separate entity for this or replace this with the individual entity and branching off this for entities like WRITER and DIRECTOR.

Redundancy

As well as duplicate attributes, there are also instances where the data itself is repeated in the tuples. For example, the duplication of most titles as they have different principals. This increases the risk of data duplication and the risk of data inconsistency. It also creates issues when you want to Update, Delete or Insert as you're not able to update all records simultaneously, you have to go through each record and update it, you also can't delete records as other data is dependent on the existing data. If you had a relational model when you create data it relies on other data to be present and eliminates the risk of these anomalies so long as the data is in 3rd normal form or higher.

Migration

Create Tables for the new system

```
CREATE TABLE Title (
   TitleID varchar(9) PRIMARY KEY,
   Title varchar(255),
   Released int(4),
   Runtime int(4),
   AverageRating float(3,2),
   NumVotes int(7)
);

CREATE TABLE Name(
  NameID varchar(9) PRIMARY KEY,
  Name varchar(100),
  BirthYear int(4),
  DeathYear int(4)
);

CREATE TABLE Genres(
  GenreID int(3) AUTO_INCREMENT PRIMARY KEY,
  Genre varchar(32)
);

CREATE TABLE Professions(
  ProfessionID int(4) AUTO_INCREMENT PRIMARY KEY,
  Profession varchar(32)
);

CREATE TABLE Title_Cast(
  CharacterPlayed varchar(32),
  TitleID varchar(9) REFERENCES Title(TitleID),
  NameID varchar(9) REFERENCES Name(NameID),
  PRIMARY KEY (CharacterPlayed, TitleID, NameID)
);

CREATE TABLE Title_Genre(
  GenreID int(3) REFERENCES Genres(GenreID),
  TitleID varchar(9) REFERENCES Title(TitleID),
  PRIMARY KEY (GenreID, TitleID)
);
```

```
CREATE TABLE KnownFor(
  NameID varchar(9) REFERENCES Name(NameID),
  TitleID varchar(9) REFERENCES Title(TitleID),
  PRIMARY KEY (NameID, TitleID)
);

CREATE TABLE Profession_Names(
  NameID varchar(9) REFERENCES Name(NameID),
  ProfessionID int(4) REFERENCES Professions(ProfessionID),
  PRIMARY KEY(NameID,ProfessionID)
);

CREATE TABLE Title_Crew(
  JobRole INT(4) REFERENCES professions(ProfessionID),
  NameID varchar(9) REFERENCES Name(NameID),
  TitleID varchar(9) REFERENCES Title(TitleID),
  PRIMARY KEY (NameID, TitleID)
);
```

Migrations

Name table migration

```
INSERT INTO name(NameId, Name, BirthYear, DeathYear)
SELECT DISTINCT NAME_ID AS NameId, Name, birthyear, deathyear
FROM onames;
```

Title table migration

```
INSERT INTO title(TitleID, Title, Released, Runtime, AverageRating, NumVotes)
SELECT DISTINCT TITLE_ID AS TitleID, TITLE AS Title, RELEASED AS Released, RUNTIME
AS Runtime, AVERAGERATING AS AverageRating, NUMVOTES AS NumVotes
FROM otitles;

INSERT IGNORE INTO title(TitleID)
SELECT DISTINCT TitleID
FROM
(
  SELECT KNOWNFORTITLES1 AS TitleID FROM onames
  UNION
  SELECT KNOWNFORTITLES2 AS TitleID FROM onames
  UNION
  SELECT KNOWNFORTITLES3 AS TitleID FROM onames
  UNION
```

```
  SELECT KNOWNFORTITLES4 AS TitleID FROM onames
) x;
```

Genres table migration

```
INSERT INTO genres (Genre)
SELECT DISTINCT genres FROM
(
  SELECT GENRE1 AS genres FROM otitles
  UNION
  SELECT GENRE2 AS genres FROM otitles
  UNION
  SELECT GENRE3 AS genres FROM otitles
) A;
```

Professions table migration

```
INSERT INTO professions (Profession)
SELECT DISTINCT prof FROM
(
  SELECT Profession1 AS prof FROM onames
  UNION
  SELECT Profession2 AS prof FROM onames
  UNION
  SELECT Profession3 AS prof FROM onames
  UNION
  SELECT DISTINCT JOB_CATEGORY AS prof FROM otitles
) A;
```

Title_genre migration

```
INSERT INTO title_genre (TitleID, GenreID)
SELECT DISTINCT TITLE_ID AS TitleID, GenreID
FROM otitles
INNER JOIN genres ON otitles.GENRE1 = genres.Genre;

INSERT INTO title_genre (TitleID, GenreID)
SELECT DISTINCT TITLE_ID AS TitleID, GenreID
FROM otitles
INNER JOIN genres ON otitles.GENRE2 = genres.Genre;

INSERT INTO title_genre (TitleID, GenreID)
SELECT DISTINCT TITLE_ID AS TitleID, GenreID
FROM otitles
INNER JOIN genres ON otitles.GENRE3 = genres.Genre;
```

KnownFor Migration

```
INSERT INTO knownfor(NameID,TitleID)
SELECT NAME_ID AS NameID, KNOWNFORTITLES1 AS TitleID
FROM onames
WHERE KNOWNFORTITLES1 IS NOT NULL;

INSERT INTO knownfor(NameID,TitleID)
SELECT NAME_ID AS NameID, KNOWNFORTITLES2 AS TitleID
FROM onames
WHERE KNOWNFORTITLES2 IS NOT NULL;

INSERT INTO knownfor(NameID,TitleID)
SELECT NAME_ID AS NameID, KNOWNFORTITLES3 AS TitleID
FROM onames
WHERE KNOWNFORTITLES3 IS NOT NULL;

INSERT INTO knownfor(NameID,TitleID)
SELECT NAME_ID AS NameID, KNOWNFORTITLES4 AS TitleID
FROM onames
WHERE KNOWNFORTITLES4 IS NOT NULL;
```

Title_Crew Migration

```
INSERT INTO title_crew(JobRole, NameID, TitleID)
SELECT ProfessionID, PRINCIPAL AS NameID, TITLE_ID AS TitleID
FROM otitles
INNER JOIN professions ON professions.Profession = otitles.JOB_CATEGORY;
```

Profession_Names

```
INSERT INTO profession_names(NameID, ProfessionID)
SELECT DISTINCT PRINCIPAL AS NameID, ProfessionID
FROM otitles
INNER JOIN professions ON professions.Profession = otitles.JOB_CATEGORY;
```

Title_Cast

```
INSERT IGNORE INTO title_cast(CharacterPlayed, TitleID, NameID)
SELECT SUBSTRING_INDEX(CHARACTERS_PLAYED,";", 1) AS CharacterPlayed, TITLE_ID
AS TitleID, PRINCIPAL AS NameID
FROM otitles
WHERE SUBSTRING_INDEX(CHARACTERS_PLAYED,";", 1) IS NOT NULL;

INSERT IGNORE INTO title_cast(CharacterPlayed, TitleID, NameID)
```

```
SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(CHARACTERS_PLAYED,";", 2), ";", -1)
AS CharacterPlayed, TITLE_ID AS TitleID, PRINCIPAL AS NameID
FROM otitles
WHERE SUBSTRING_INDEX(SUBSTRING_INDEX(CHARACTERS_PLAYED,";", 2), ";", -1) IS
NOT NULL;

INSERT IGNORE INTO title_cast(CharacterPlayed, TitleID, NameID)
SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(CHARACTERS_PLAYED,";", 3), ";", -1)
AS CharacterPlayed, TITLE_ID AS TitleID, PRINCIPAL AS NameID
FROM otitles
WHERE SUBSTRING_INDEX(SUBSTRING_INDEX(CHARACTERS_PLAYED,";", 3), ";", -1) IS
NOT NULL;

UPDATE title_cast
SET CharacterPlayed = SUBSTR(CharacterPlayed, 2)
WHERE CharacterPlayed LIKE '"%';

UPDATE IGNORE title_cast
SET CharacterPlayed = SUBSTR(CharacterPlayed, 2)
WHERE CharacterPlayed LIKE '"%';

UPDATE IGNORE title_cast
SET CharacterPlayed = SUBSTR(CharacterPlayed, 2)
WHERE CharacterPlayed LIKE '"%";

UPDATE IGNORE title_cast
SET CharacterPlayed = SUBSTR(CharacterPlayed, 1, LENGTH(CharacterPlayed)-1)
WHERE CharacterPlayed LIKE '%"';
```

Indices

```
ALTER TABLE genres ADD INDEX(GenreID);
ALTER TABLE knownfor ADD INDEX(NameID);
ALTER TABLE name ADD INDEX(NameID);
ALTER TABLE professions ADD INDEX(ProfessionID);
ALTER TABLE profession_names ADD INDEX(ProfessionID);
ALTER TABLE title ADD INDEX(TitleID);
ALTER TABLE title_cast ADD INDEX(TitleID);
ALTER TABLE title_crew ADD INDEX(TitleID);
ALTER TABLE title_genre ADD INDEX(TitleID);
```

Create database

See .zip

SQL queries

A:

List all actors and actresses and the names of the titles for which they are known. When the title's name is not stored in the database, show the title_id instead. Order the results by the name_id of the actor/actress:

```
SELECT Name,
CASE
        WHEN Title IS NULL THEN TitleID
ELSE Title
END AS "Known for title"
FROM name
LEFT JOIN knownfor USING(NameID)
LEFT JOIN title USING(TitleID)
ORDER BY NameID ASC;
```

B:

List all writer and director pairs who have worked together at least once in a drama:

```
SELECT t1.TitleID, t1.Title,
(
  SELECT name.Name
  FROM title t3
  INNER JOIN title_crew ON t3.TitleID = title_crew.TitleID
  INNER JOIN profession_names ON title_crew.NameID = profession_names.NameID
  INNER JOIN professions ON profession_names.ProfessionID = professions.ProfessionID
  INNER JOIN name ON title_crew.NameID = name.NameID
  WHERE professions.Profession = "writer" AND t3.TitleID = t1.TitleID
        LIMIT 1
) AS Writer,
(
  SELECT name.Name
  FROM title t2
  INNER JOIN title_crew ON t2.TitleID = title_crew.TitleID
  INNER JOIN profession_names ON title_crew.NameID = profession_names.NameID
  INNER JOIN professions ON profession_names.ProfessionID = professions.ProfessionID
  INNER JOIN name ON title_crew.NameID = name.NameID
  WHERE professions.Profession = "director" AND t2.TitleID = t1.TitleID
        LIMIT 1
) AS Director
FROM title t1
```

```
INNER JOIN title_genre ON title_genre.TitleID = t1.TitleID
INNER JOIN genres ON title_genre.GenreID = genres.GenreID
WHERE genres.Genre = "Drama" AND (
SELECT name.Name
  FROM title t3
  INNER JOIN title_crew ON t3.TitleID = title_crew.TitleID
  INNER JOIN profession_names ON title_crew.NameID = profession_names.NameID
  INNER JOIN professions ON profession_names.ProfessionID = professions.ProfessionID
  INNER JOIN name ON title_crew.NameID = name.NameID
  WHERE professions.Profession = "writer" AND t3.TitleID = t1.TitleID
        LIMIT 1
) IS NOT NULL AND (
 SELECT name.Name
  FROM title t2
  INNER JOIN title_crew ON t2.TitleID = title_crew.TitleID
  INNER JOIN profession_names ON title_crew.NameID = profession_names.NameID
  INNER JOIN professions ON profession_names.ProfessionID = professions.ProfessionID
  INNER JOIN name ON title_crew.NameID = name.NameID
  WHERE professions.Profession = "director" AND t2.TitleID = t1.TitleID
        LIMIT 1
) IS NOT NULL;
```

C:

List every group of “actor/actress, writer, director” who worked on the same title, ordered by title_ID. That is, each row should contain an actor/actress, a writer and a director who all worked on the same title. Make sure that you exclude groups where the same person appears in more than one column, e.g. where one person was both writer and director:

```
SELECT t1.TitleID, t1.Title, name.Name AS Actor,
(
  SELECT name.Name
  FROM title t2
  INNER JOIN title_crew ON t2.TitleID = title_crew.TitleID
  INNER JOIN profession_names ON title_crew.NameID = profession_names.NameID
  INNER JOIN professions ON profession_names.ProfessionID = professions.ProfessionID
  INNER JOIN name ON title_crew.NameID = name.NameID
  WHERE professions.Profession = "writer" AND t2.TitleID = t1.TitleID
        LIMIT 1
) AS Writer,
(
  SELECT name.Name
  FROM title t3
  INNER JOIN title_crew ON t3.TitleID = title_crew.TitleID
```

```
  INNER JOIN profession_names ON title_crew.NameID = profession_names.NameID
  INNER JOIN professions ON profession_names.ProfessionID = professions.ProfessionID
  INNER JOIN name ON title_crew.NameID = name.NameID
  WHERE professions.Profession = "director" AND t3.TitleID = t1.TitleID
        LIMIT 1
) AS Director
FROM title t1
INNER JOIN title_crew ON t1.TitleID = title_crew.TitleID
INNER JOIN name ON title_crew.NameID = name.NameID
INNER JOIN profession_names ON name.NameID = profession_names.NameID
INNER JOIN professions ON profession_names.ProfessionID = professions.ProfessionID
WHERE (professions.Profession = 'actor' OR professions.Profession = 'actress') AND (
  SELECT name.Name
  FROM title t2
  INNER JOIN title_crew ON t2.TitleID = title_crew.TitleID
  INNER JOIN profession_names ON title_crew.NameID = profession_names.NameID
  INNER JOIN professions ON profession_names.ProfessionID = professions.ProfessionID
  INNER JOIN name ON title_crew.NameID = name.NameID
  WHERE professions.Profession = "writer" AND t2.TitleID = t1.TitleID
        LIMIT 1
) IS NOT NULL AND (
  SELECT name.Name
  FROM title t3
  INNER JOIN title_crew ON t3.TitleID = title_crew.TitleID
  INNER JOIN profession_names ON title_crew.NameID = profession_names.NameID
  INNER JOIN professions ON profession_names.ProfessionID = professions.ProfessionID
  INNER JOIN name ON title_crew.NameID = name.NameID
  WHERE professions.Profession = "director" AND t3.TitleID = t1.TitleID
        LIMIT 1
) IS NOT NULL AND
(
  SELECT name.Name
  FROM title t2
  INNER JOIN title_crew ON t2.TitleID = title_crew.TitleID
  INNER JOIN profession_names ON title_crew.NameID = profession_names.NameID
  INNER JOIN professions ON profession_names.ProfessionID = professions.ProfessionID
  INNER JOIN name ON title_crew.NameID = name.NameID
  WHERE professions.Profession = "writer" AND t2.TitleID = t1.TitleID
        LIMIT 1
) <> (
 SELECT name.Name
  FROM title t3
  INNER JOIN title_crew ON t3.TitleID = title_crew.TitleID
  INNER JOIN profession_names ON title_crew.NameID = profession_names.NameID
```

```
  INNER JOIN professions ON profession_names.ProfessionID = professions.ProfessionID
  INNER JOIN name ON title_crew.NameID = name.NameID
  WHERE professions.Profession = "director" AND t3.TitleID = t1.TitleID
        LIMIT 1
);
```

D:

Count the number of people involved (in any capacity) with each genre as well as the total number of people regardless of genre:

```
SELECT Genre, COUNT(DISTINCT title_crew.NameID) AS 'Number'
FROM genres
LEFT JOIN title_genre USING(GenreID)
LEFT JOIN title USING(TitleID)
LEFT JOIN title_crew USING(TitleID)
GROUP BY Genre
ORDER BY Genre ASC;
```

E:

List all the living cinematographers who are known for titles with average ratings of 4.5 or less and the name of the title with their lowest rating. Order the results by the lowest average rating from highest to lowest:

```
SELECT Name, AverageRating, Title
FROM Name
INNER JOIN profession_names USING(NameID)
INNER JOIN professions USING(ProfessionID)
INNER JOIN title_crew USING(NameID)
INNER JOIN title USING(TitleID)
WHERE Profession = "cinematographer" AND AverageRating <= 4.5 AND DeathYear IS NULL
ORDER BY AverageRating DESC;
```