A6: Relational schema, validation and schema refinement

In this artifact we present the Relational schema, showing its tables and attributes. All of them will be on the database and it's important that they are on BCNF format to reduce the redundancy on the database. The validation is done as well.

1. Tables

WishList(memberID → Member, movieID → Movie)
Member(memberID, email NN UK, firstName NN, lastName NN, password NN, bannedUser DF, billingInformationID → BillingInformation, deliveryAddressID → DeliveryAddress, paymentInformationID → PaymentInformation, cartID → Cart)
Language(languageID, name NN)
SubtitlesFortheHearingImpaired(languageID → Language, movieID → Movie)
Subtitle(languageID → Language, movieID → Movie)
Audio(languageID → Language, movieID → Movie)
Studio(studioID, name NN UK)
Review(reviewID, title, description, rating, movieID → Movie NN, memberID → Member NN)
PurchaseMovie(purchaseID → Purchase, movieID → Movie, quantity NN)
Purchase(purchaseID, price NN, dateOfPurchase NN, memberID → Member NN)
City(cityID, name NN)
Country(countryID, name NN UK)
PostCode(postCodeID, name NN UK)
Actor(personID, name NN)
Director(personID, name NN)
Cart(cartID, price NN)
MovieCart(cartID → Cart, movieID → Movie, quantity NN)
BillingInformation(billingInformationID, cityID → City, countryID → Country, postcodeID → PostCode, address NN, fullName NN)
DeliveryAddress(deliveryAddressID, cityID → City, countryID → Country, postcodeID → PostCode, address NN, fullName NN)
PaymentInformation(paymentInformationID, creditCardNumber NN, cvc NN, expirationDate NN)
Movie(movieID, classification NN, description, imagePath NN, name NN, numberOfDiscs NN, price NN, region NN, releaseDate NN, runtime NN, stock NN, studioId → Studio)
MovieActor(personID → Actor, movieID → Movie)
MovieDirector(personID → Director, movieID → Movie)
Format(formatID, name NN)

Additional notes: UK is UNIQUE, NN means NOT NULL and DF means DEFAULT.

2. Domains

In this section we will describe the domain of each class attribute, that we created for this project.

Movie

Attribute	Type
Classification	ENUM(3, 7, 12, 16, 18)
Region	ENUM(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C)

3. Checks

Movie

numberOfDiscs	>	0
price	>	0
stock	>=	0
runtime	>	0

PurchaseMovie

quantity	>	0

MovieCart

quantity	>	0

Review

rating	>=	1
rating	⇐	5

Cart

totalCost	>=	0

Purchase

price	>	0
dateOfPurchase	>	date in which the movie was added

Payment Information

expirationDate	>	TODAY

3. Relation schema in SQL

Code for the **Member** table:

```
DROP TABLE IF EXISTS Member;
CREATE TABLE Member (
    memberID SERIAL NOT NULL PRIMARY KEY,
    bannedMember INTEGER DEFAULT ,
    email TEXT NOT NULL UNIQUE,
    firstName TEXT NOT NULL,
    lastName TEXT NOT NULL,
    password TEXT NOT NULL,
    billingInformationID INTEGER REFERENCES
BillingInformation(billingInformationID),
    paymentInformationID INTEGER REFERENCES
PaymentInformation(paymentInformationID),
    deliveryAdressID INTEGER REFERENCES DeliveryAdress(deliveryAdressID),
    cartID INTEGER REFERENCES Cart(cartID)
);
```

Code for the **Billing Information**, **Delivery Address** and **Payment Information** tables:

```
DROP TABLE IF EXISTS BillingInformation;
CREATE TABLE BillingInformation (
    billingInformationID SERIAL NOT NULL PRIMARY KEY,
    cityID INTEGER NOT NULL REFERENCES City(cityID),
    countryID INTEGER NOT NULL REFERENCES Country(countryID),
    postcodeID INTEGER NOT NULL REFERENCES PostCode(postcodeID),
    address TEXT NOT NULL,
    fullName TEXT NOT NULL
);

DROP TABLE IF EXISTS DeliveryAdress;
CREATE TABLE DeliveryAdress (
    deliveryAdressID SERIAL NOT NULL PRIMARY KEY,
    cityID INTEGER REFERENCES City(cityID),
    countryID INTEGER REFERENCES Country(countryID),
    postcodeID INTEGER REFERENCES PostCode(postcodeID),
    address TEXT NOT NULL,
    fullName TEXT NOT NULL
);

DROP TABLE IF EXISTS PaymentInformation;
CREATE TABLE PaymentInformation (
    paymentInformationID SERIAL NOT NULL PRIMARY KEY,
    creditCardNumber INTEGER NOT NULL,
    cvc INTEGER NOT NULL,
    expirationDate DATE NOT NULL
);
```

Code for the **Cart** and **Movie Cart** table:

```
DROP TABLE IF EXISTS Cart;
CREATE TABLE Cart (
    cartID SERIAL NOT NULL PRIMARY KEY,
    totalCost REAL NOT NULL,
    CHECK(totalCost > )
);

DROP TABLE IF EXISTS MovieCart;
CREATE TABLE MovieCart (
    cartID INTEGER REFERENCES Cart(cartID),
    movieID INTEGER REFERENCES Movie(movieID),
    quantity INTEGER NOT NULL,
    CHECK(quantity > ),
    PRIMARY KEY(cartID, movieID)
);
```

Code for the **City**, **Country** and **PostCode** tables:

```
DROP TABLE IF EXISTS City;
CREATE TABLE City (
    cityID SERIAL NOT NULL PRIMARY KEY,
    name TEXT NOT NULL
);

DROP TABLE IF EXISTS Country;
CREATE TABLE Country (
    countryID SERIAL NOT NULL PRIMARY KEY,
    name TEXT UNIQUE
);

DROP TABLE IF EXISTS PostCode;
CREATE TABLE PostCode (
    postCodeID SERIAL NOT NULL PRIMARY KEY,
    name INTEGER NOT NULL UNIQUE
);
```

Code for the **Format** table:

```
DROP TABLE IF EXISTS Format;
CREATE TABLE Format (
    formatID SERIAL NOT NULL PRIMARY KEY,
    name TEXT NOT NULL
);
```

Code for the **Movie** table:

```
DROP TABLE IF EXISTS Movie;
CREATE TABLE Movie (
    classification INTEGER NOT NULL,
    description TEXT,
    imagePath TEXT NOT NULL,
    name TEXT NOT NULL,
    numberOfDiscs INTEGER NOT NULL,
    price REAL NOT NULL,
    region TEXT NOT NULL,
    releaseDate DATE,
    runtime INTEGER NOT NULL,
    stock INTEGER NOT NULL,
    averageScore REAL,
    movieID SERIAL NOT NULL PRIMARY KEY,
    genreID SERIAL NOT NULL REFERENCES Genre(genreID),
    formatID INTEGER NOT NULL REFERENCES Format(formatID),
    studioID INTEGER NOT NULL REFERENCES Studio(studioID),
    CHECK(numberOfDiscs > ),
    CHECK(price > ),
    CHECK(stock >= ),
    CHECK(runtime > ),
    CHECK(averageScore >  AND averageScore <= 5),
    CHECK((classification = 3) OR (classification = 7) OR (classification =
12) OR (classification = 16) OR (classification = 18))
);
```

Code for the **Director**, **Movie Director**, **Actor** and **Movie Actor** tables:

```
DROP TABLE IF EXISTS Director;
CREATE TABLE Director (
    personID SERIAL NOT NULL PRIMARY KEY,
    name TEXT NOT NULL
);

DROP TABLE IF EXISTS Actor;
CREATE TABLE Actor (
    personID SERIAL NOT NULL PRIMARY KEY,
    name TEXT NOT NULL
);

DROP TABLE IF EXISTS MovieActor;
CREATE TABLE MovieActor (
    movieID INTEGER REFERENCES Movie(movieID),
    personID INTEGER REFERENCES Actor(personID),
    PRIMARY KEY(movieID, personID)
);


DROP TABLE IF EXISTS MovieDirector;
CREATE TABLE MovieDirector (
    movieID INTEGER REFERENCES Movie(movieID),
    personID INTEGER REFERENCES Director(personID),
    PRIMARY KEY(movieID, personID)
);
```

Code for the **Language**, **Subtitle**, **Subtitle for the hearing impaired** and **Audio** tables:

```
DROP TABLE IF EXISTS LANGUAGE;
CREATE TABLE LANGUAGE (
    languageID SERIAL NOT NULL PRIMARY KEY,
    name TEXT NOT NULL
);

DROP TABLE IF EXISTS Subtitle;
CREATE TABLE Subtitle (
    languageID INTEGER REFERENCES LANGUAGE(languageID),
    movieID INTEGER REFERENCES Movie(movieID),
    PRIMARY KEY(languageID, movieID)
);

DROP TABLE IF EXISTS SubtitlesFortheHearingImpaired;
CREATE TABLE SubtitlesFortheHearingImpaired (
    languageID INTEGER REFERENCES LANGUAGE(languageID),
    movieID INTEGER REFERENCES Movie(movieID),
    PRIMARY KEY(languageID, movieID)
);

DROP TABLE IF EXISTS Audio;
CREATE TABLE Audio (
    languageID INTEGER REFERENCES LANGUAGE(languageID),
    movieID INTEGER REFERENCES Movie(movieID),
    PRIMARY KEY(languageID, movieID)
);
```

Code for the **Purchase** and **Purchase Movie** table:

```
DROP TABLE IF EXISTS Purchase;
CREATE TABLE Purchase (
    purchaseID SERIAL NOT NULL PRIMARY KEY,
    dateOfPurchase DATE NOT NULL,
    price REAL NOT NULL,
    memberID INTEGER NOT NULL REFERENCES Member(memberID)
);

DROP TABLE IF EXISTS PurchaseMovie;
CREATE TABLE PurchaseMovie (
    purchaseID INTEGER REFERENCES Purchase(purchaseID),
    movieID INTEGER REFERENCES Movie(movieID),
    quantity INTEGER NOT NULL,
    CHECK(quantity > ),
    PRIMARY KEY(purchaseID, movieID)
);
```

Code for the **Review** table:

```
DROP TABLE IF EXISTS Review;
CREATE TABLE Review (
    reviewID SERIAL NOT NULL PRIMARY KEY,
    title TEXT NOT NULL,
    description TEXT,
    rating INTEGER NOT NULL,
    movieID INTEGER NOT NULL REFERENCES Movie(movieID),
    memberID INTEGER NOT NULL REFERENCES Member(memberID),
    CHECK (rating >= 1 AND rating <= 5)
);
```

Code for the **Studio** table:

```
DROP TABLE IF EXISTS Studio;
CREATE TABLE Studio (
    studioID SERIAL NOT NULL PRIMARY KEY,
    name TEXT NOT NULL UNIQUE
);
```

Code for the **Genre** table:

```
DROP TABLE IF EXISTS Genre;
CREATE TABLE Genre (
    genreID SERIAL NOT NULL PRIMARY KEY,
    name TEXT NOT NULL UNIQUE
);
```

Code for the **Wishlist** table:

```
DROP TABLE IF EXISTS WishList;
CREATE TABLE WishList (
    memberID INTEGER REFERENCES Member(memberID),
    movieID INTEGER REFERENCES Movie(movieID),
    PRIMARY KEY(memberID, movieID)
);
```

4. Validation

To validate our tables it's important to see if they do not violate the BCNF. Below we will present all the Functional Dependencies.

WishList	No functional dependencies
Member	memberID → email, firstName, lastName, password, bannedUser, billingInformationID, deliveryAddressID, paymentInformationID, cartID
	email → memberID, firstName, lastName, password, bannedUser, billingInformationID, deliveryAddressID, paymentInformationID, cartID
Language	languageID → name
SubtitlesFortheHearingImpaired	No functional dependencies
Subtitle	No functional dependencies
Audio	No functional dependencies
Studio	studioID → name
Review	reviewID → description, rating, movieID, memberID
PurchaseMovie	purchaseID, movieID → quantity

Purchase	purchaseID → price, dateOfPurchase, memberID
City	cityID → name
Country	countryID → name
PostCode	postCodeID→ name
Actor	personID → name
Director	personID → name
Cart	cartID → price
MovieCart	cartID → movieID, quantity
BillingInformation	billingInformationID → countryID, postcodeID, address, fullName
DeliveryAddress	deliveryAddressID → countryID, postcodeID, address, fullName
PaymentInformation	paymentInformationID → cityID, postcodeID, creditCardNumver, cvc, expirationDate
Movie	movieID → classification, description, imagePath, name, numberOfDiscs, price, region, releaseDate, runtime, stock
MovieActor	No functional dependencies
MovieDirector	No functional dependencies
Format	formatID → name

All the tables are in the BCNF, since each relation depends only the key.