Database Design of A Film Subscription Website

[Md Shahadat Hossen Nayem]

The topic of the project is designing database of a Film rental website. The Film Subscription database represents some of the data storage and retrieval requirements someone might encounter when dealing with subscription in a film rental website. The film subscription database contains nineteen main type of records: film, region, language, film category, actor category, actor, category, directory, subscription, payment, staff, customer, store, address, city, country, bought items, customer category and categoryc. Each record is identified by a unique number. Each records has been visualized by table in Entity Relationship Diagram below. Each Table Translate into an entity and column category are listed as attributes of their respective entity. In this report, i will explain the film subscription database using 3 different types of entity relation diagrams: Conceptual,Logical and Physical. And presend the data definition language of the database.

Description

The film subscription database contains nineteen main type of records: film, region, language, film category, actor category, actor, category, directory, subscription, payment, staff, customer, store, address, city, country, bought items, customer category and categoryc.

Each film has an identification unique number, title, rate, concept, rental duration, replacement cost, rating, length, special features, picture url and release year. Each film must need to have a title. So the title can not be null. All film related information is stored in the Film table. Each language and region table is assigned to a film.

The Conceptual Entity Relationship diagram below visualize the attributes of their respective tables

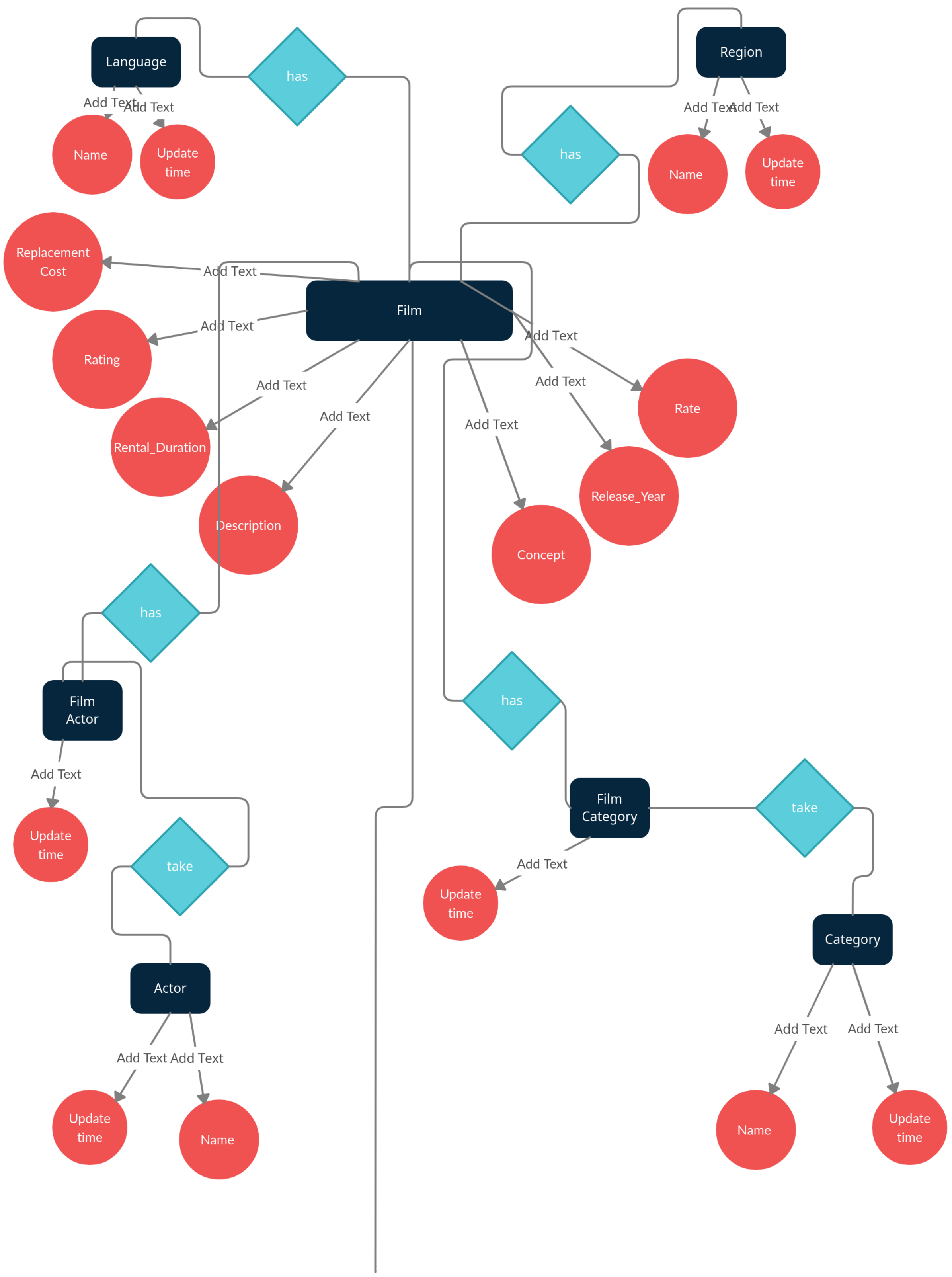
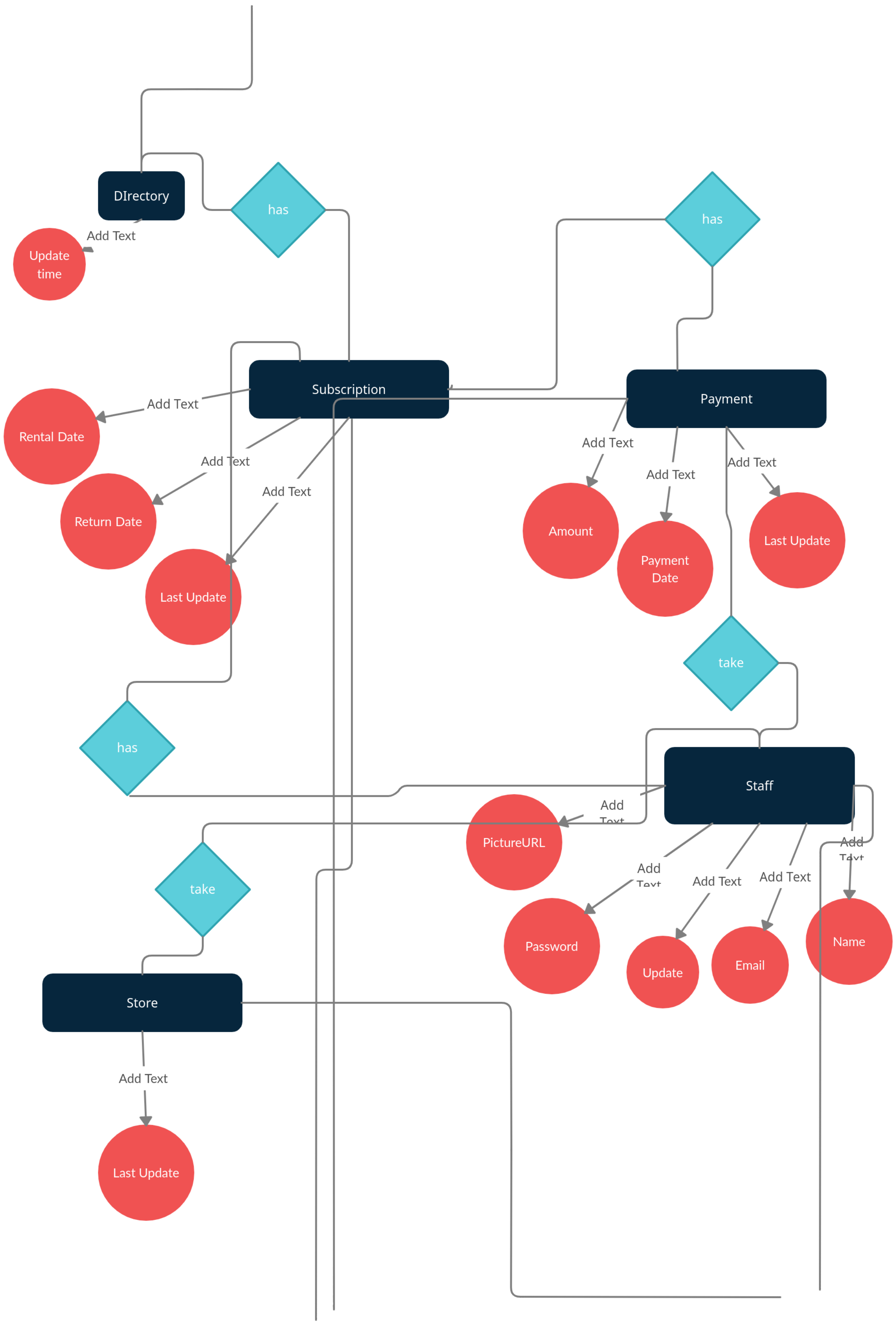


figure: Conceptual Diagram

Each Language has an identification unique number, name and last update. Last update will track when the information was change last. In the conceptual diagram above, it visualizes the relationship between Film and Language. A film need to have a language and this is why conceptual diagram represent 'has a' relationship between this two tables.

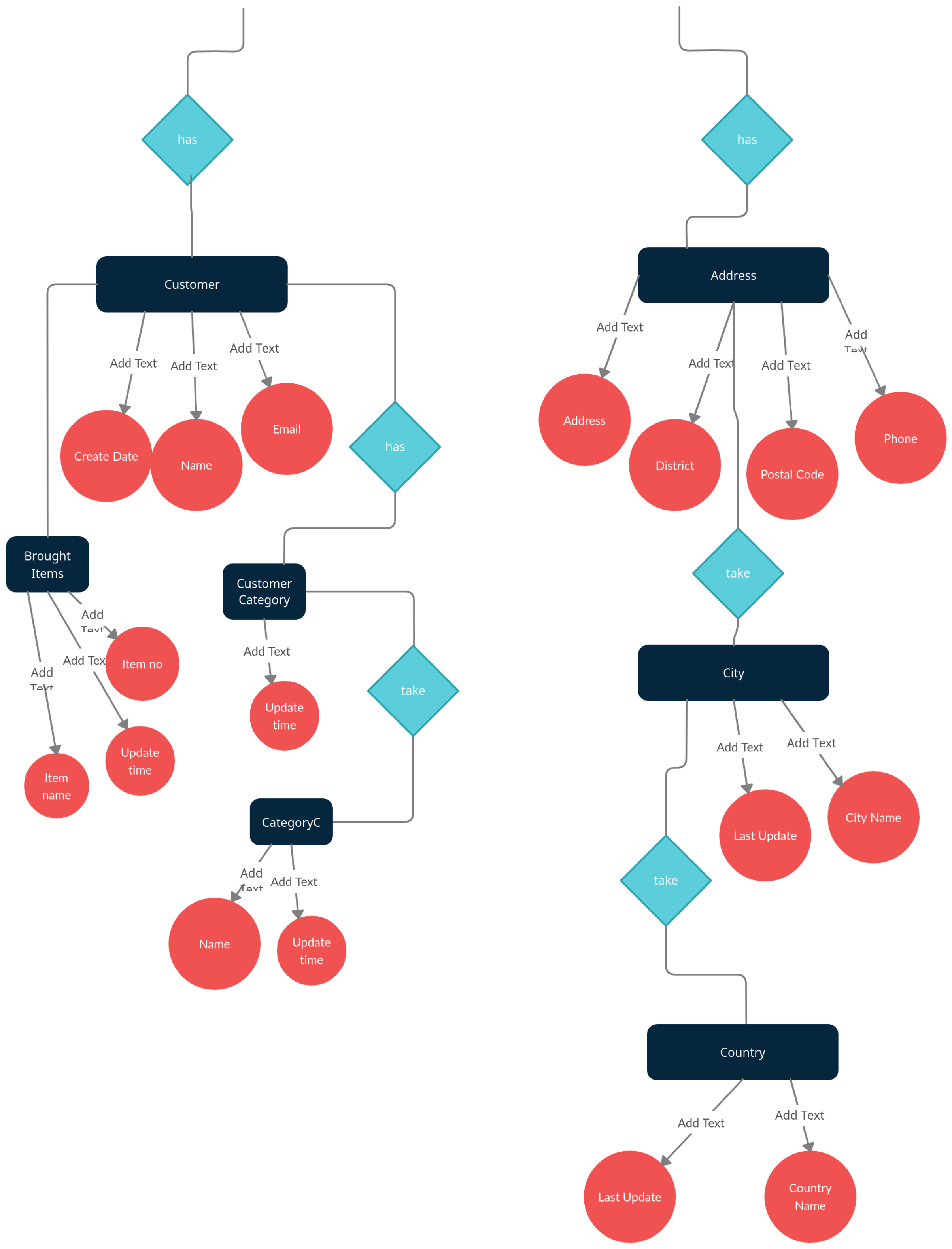
Same for the table Region which has an identification unique number, name and last update.

The diagram represents each type of records and their attributes. Film Actor has an identification number and last update information. Each actor and film table is assigned to a Film actor table. So film actor will basically hold the reference of film and actor table and can access them from film actor table by using their identification id. Actor entity has an identification number, name and last update time. Category entity has an identification number, name and last update time and film category has identification number and last update time.


DIrectory
Add Text
Update time
has
has
Subscription
Add Text
Rental Date
Add Text
Return Date
Add Text
Last Update
Payment
Add Text
Amount
Add Text
Payment Date
Add Text
Last Update
take
has
Staff
Add Text
PictureURL
take
Add Text
Password
Add Text
Update
Add Text
Email
Add Text
Name
Store
Add Text
Last Update

Each Film table is assigned to a Directory. Directory has two attributes which are update time and it's identification id.

The conceptual diagram visualizes the attributes and the relationship between the table subscription, payment staff and store in the above diagram. Subscription has identification number, rental date, return date and last update. Payment has identification number, ramount , payment date and last update. Staff has identification number, email, password, name and last update. Store has identification number and last update. Subsciption has staff and payment.



Customer has identification number, create date, name and email. Address has identification number, district , postal code and phone. Country has identification country number and last update. Brought items has item name and last update. The category of customer which is CategoryC has identification number, name, and update date.

The next diagram is Logical Entity relationship diagram. It shows logical information and relation between tables.

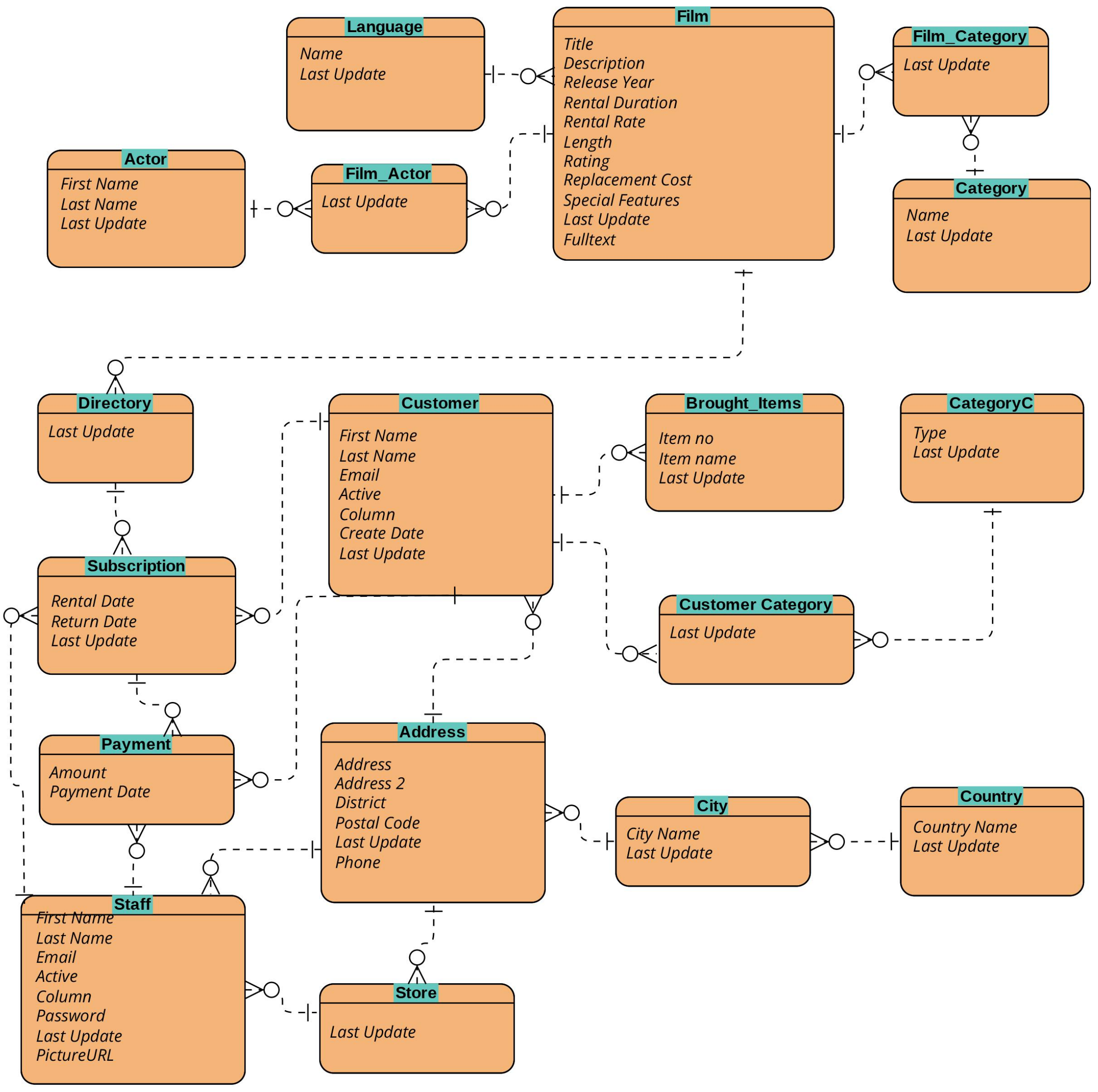


figure: Logical Diagram

The logical relationship of the diagrams is visualized in this logical diagram. Staff, directory and customer are assigned to subscription table. On the other hand, subscription is assigned to payment table. In the same way, customer and staff are also assigned in the payment table. City is assigned in the address entity and country is assigned in the city entity.

Customer category's main responsibility is to connect two table are customer and categoryc. But still for more specific informotion about the attributes and the tables we must need to look at the physical diagram below.

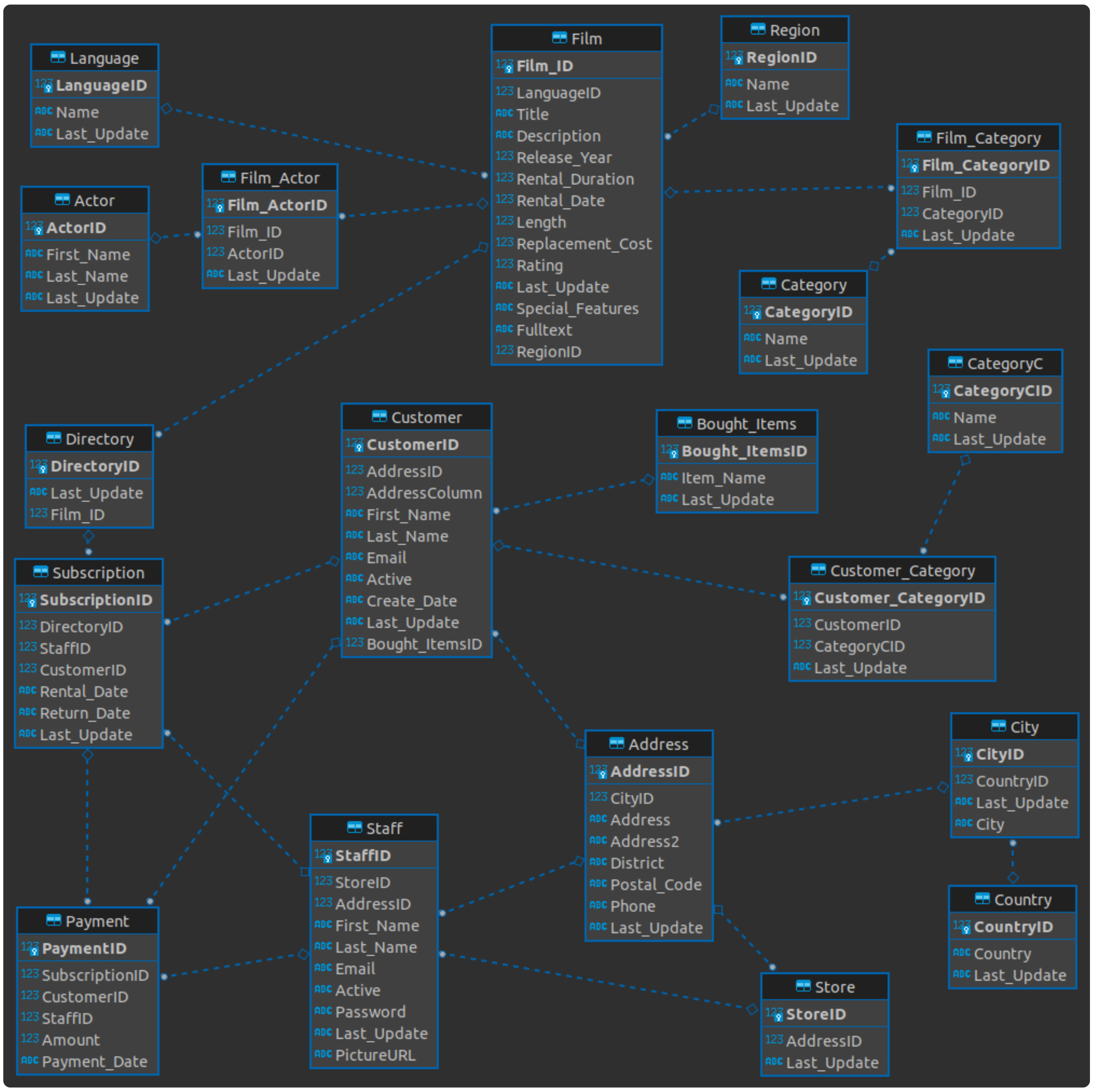


Figure: Physical Diagram

The logical diagram shows the identification number which is basically the Primary key of a table. For instance, Film_ID is the primary key of the table Film, LanguageID is the primary key of the table Language.

We can see LanguageID and RegionID are assigned in the table Film as foreign key. This is how two table connect logically. Film_Actor and Film_Category has two foreign key each. So these table can connect two different table logically. The physical diagram also represent the datatype of the attributes.

In entity relationship diagram, there are multiple type of data type. For instance, numeric, integer, datetime, varchar, char etc. In staff table, First_Name, Last_name, email and password data type is varchar and the StoreID, StaffID are integer. Which means when data will be inserted in the database, it will only accept specific type of data type for specific attribute. For example, last update should need to be datetime datatype which is date.

SQL

```sql
INSERT INTO Language VALUES ('English','2012-12-31');
```

So if other type of data type are inserted then it will not save that data and ask for date type of data if the attribute is not null attribute. For the attribute active in the customer table, the data type is char.

But still there are a lot of information is missing in a physical diagram. If a attribute can be null or not, is defined by DDL which stands for Database Definition Language. Lets have a look on the entity relation diagram's DDL below.

The text of the Film Subscription SQL script is as follows:

JavaScript

```javascript
--
--  Script is used to create tables, views, procedures,
--  functions, triggers, etc.
--
--  Start new transaction commit all or nothing
--
BEGIN;
--
--  Create and load tables used in the documentation examples.
--
--  The command below will create the 'Film' table

CREATE TABLE Film (
        Film_ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        LanguageID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        RegionID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Title varchar(255) NOT NULL,
        Description varchar(255),
        Release_Year INTEGER NOT NULL,
        Rental_Duration INTEGER NOT NULL,
```

```sql
        Rental_Rate NUMERIC(19, 0) NOT NULL,
        "Length" INTEGER,
        Replacement_Cost NUMERIC(19, 0) NOT NULL,
        Rating INTEGER,
        Special_Features varchar(255),
        FullText varchar(255),
        Last_Update datetime NOT NULL,
        CONSTRAINT Film_PK PRIMARY KEY (Film_ID),
        CONSTRAINT Film_FK FOREIGN KEY (LanguageID) REFERENCES "Language"(LanguageID),
        CONSTRAINT Film_FK_1 FOREIGN KEY (RegionID) REFERENCES Region(RegionID)
);

```

In the sql script, we can see more specific data type, auto increament, not null different type of comands has been used. Autoincrement comand will create unique id number each time when the table is used.

Below I am going to add all DDL to create tables:

- Film

```
CREATE TABLE Film (
        Film_ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        LanguageID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        RegionID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Title varchar(255) NOT NULL,
        Description varchar(255),
        Release_Year INTEGER NOT NULL,
        Rental_Duration INTEGER NOT NULL,
        Rental_Rate NUMERIC(19, 0) NOT NULL,
        "Length" INTEGER,
        Replacement_Cost NUMERIC(19, 0) NOT NULL,
        Rating INTEGER,
        Special_Features varchar(255),
        FullText varchar(255),
        Last_Update datetime NOT NULL,
        CONSTRAINT Film_PK PRIMARY KEY (Film_ID),
        CONSTRAINT Film_FK FOREIGN KEY (LanguageID) REFERENCES "Language"
(LanguageID),
        CONSTRAINT Film_FK_1 FOREIGN KEY (RegionID) REFERENCES
Region(RegionID)
);
```

- Language

```
CREATE TABLE "Language" (
        LanguageID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Name varchar(20) NOT NULL,
        Last_Update datetime NOT NULL,
        CONSTRAINT Language_PK PRIMARY KEY (LanguageID)
);
```

- Region

```
CREATE TABLE Region (
        RegionID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Name varchar(20) NOT NULL,
        Last_Update datetime NOT NULL,
        CONSTRAINT Region_PK PRIMARY KEY (RegionID)
);
```

- Film_Category

```
CREATE TABLE Film_Category (
        Film_CategoryID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Film_ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        CategoryID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Last_Update datetime NOT NULL,
        CONSTRAINT Film_Category_PK PRIMARY KEY (Film_CategoryID),
        CONSTRAINT Film_Category_FK FOREIGN KEY (Film_ID) REFERENCES
Film(Film_ID),
        CONSTRAINT Film_Category_FK_1 FOREIGN KEY (CategoryID) REFERENCES
Category(CategoryID)
);
```

- Category

```
CREATE TABLE Category (
        CategoryID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
         Last_Update datetime NOT NULL,
         Name varchar(20) NOT NULL,
        CONSTRAINT Category_PK PRIMARY KEY (CategoryID)
         
);
```

- Film_Actor

```
CREATE TABLE Film_Actor (
        Film_ActorID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Film_ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        ActorID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Last_Update datetime NOT NULL,
        CONSTRAINT Film_Actor_PK PRIMARY KEY (Film_ActorID),
        CONSTRAINT Film_Actor_FK FOREIGN KEY (Film_ID) REFERENCES 
Film(Film_ID),
        CONSTRAINT Film_Actor_FK_1 FOREIGN KEY (ActorID) REFERENCES 
Actor(ActorID)
);
```

- Actor

```
CREATE TABLE Actor (
        ActorID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        First_Name varchar(255) NOT NULL,
        Last_Name varchar(255) NOT NULL,
        Last_Update datetime NOT NULL,
        CONSTRAINT Actor_PK PRIMARY KEY (ActorID)
);
```

- Directory

```
CREATE TABLE Directory (
        DirectoryID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Film_ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Last_Update datetime NOT NULL,
        CONSTRAINT Directory_PK PRIMARY KEY (DirectoryID),
        CONSTRAINT Directory_FK FOREIGN KEY (Film_ID) REFERENCES 
Film(Film_ID)
);
```

- Subscription

```
CREATE TABLE Subscription (
        SubscriptionID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        DirectoryID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        StaffID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        CustomerID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Rental_Date datetime NOT NULL,
        Return_Date datetime NOT NULL,
        Last_Update datetime NOT NULL,
        CONSTRAINT Subscription_PK PRIMARY KEY (SubscriptionID),
        CONSTRAINT Subscription_FK FOREIGN KEY (DirectoryID) REFERENCES 
Directory(DirectoryID),
        CONSTRAINT Subscription_FK_1 FOREIGN KEY (StaffID) REFERENCES 
Staff(StaffID),
        CONSTRAINT Subscription_FK_2 FOREIGN KEY (CustomerID) REFERENCES 
Customer(CustomerID)
);
```

- Payment

```
CREATE TABLE Payment (
        PaymentID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        SubscriptionID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        CustomerID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        StaffID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Amount NUMERIC NOT NULL,
        Payment_Date datetime NOT NULL,
        CONSTRAINT Payment_PK PRIMARY KEY (PaymentID),
        CONSTRAINT Payment_FK FOREIGN KEY (SubscriptionID) REFERENCES 
Subscription(SubscriptionID),
        CONSTRAINT Payment_FK_1 FOREIGN KEY (StaffID) REFERENCES
Staff(StaffID),
        CONSTRAINT Payment_FK_2 FOREIGN KEY (CustomerID) REFERENCES
Customer(CustomerID)
);
```

- Staff

```
CREATE TABLE Staff (
        StaffID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        StoreID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        AddressID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        First_Name varchar(255) NOT NULL,
        Last_Name varchar(255) NOT NULL,
        Email varchar(50) NOT NULL,
        Active Char,
        Password varchar(40) NOT NULL,
        PictureURL varchar(80),
        Last_Update datetime NOT NULL,
        CONSTRAINT Staff_PK PRIMARY KEY (StaffID),
        CONSTRAINT Staff_FK FOREIGN KEY (StoreID) REFERENCES
Store(StoreID),
        CONSTRAINT Staff_FK_1 FOREIGN KEY (AddressID) REFERENCES
Address(AddressID)
);

```

- Store

```
CREATE TABLE Store (
        StoreID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        AddressID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Last_Update datetime NOT NULL,
        CONSTRAINT Store_PK PRIMARY KEY (StoreID),
        CONSTRAINT Store_FK FOREIGN KEY (AddressID) REFERENCES
Address(AddressID)
);
```

- Customer

```
CREATE TABLE Customer (
        CustomerID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        AddressID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Bought_ItemsID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        AddressColumn INTEGER,
        First_Name varchar(255) NOT NULL,
        Last_Name varchar(255) NOT NULL,
        Email varchar(50) NOT NULL,
        Active Char(1),
        Create_Date datetime NOT NULL,
        Last_Update datetime NOT NULL,
        CONSTRAINT Customer_PK PRIMARY KEY (CustomerID),
        CONSTRAINT Customer_FK FOREIGN KEY (Bought_ItemsID) REFERENCES
Bought_Items(Bought_ItemsID),
        CONSTRAINT Customer_FK_1 FOREIGN KEY (AddressID) REFERENCES
Address(AddressID)
);
```

- Address

```
CREATE TABLE Address (
        AddressID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        CityID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Address1 varchar(50) NOT NULL,
        Address2 varchar(50),
        District INTEGER NOT NULL,
        Postal_Code varchar(10) NOT NULL,
        Phone varchar(20),
        Last_Update datetime NOT NULL,
        CONSTRAINT Address_PK PRIMARY KEY (AddressID),
        CONSTRAINT Address_FK FOREIGN KEY (CityID) REFERENCES City(CityID)
);
```

- Bought_Items

```
CREATE TABLE Bought_Items (
        Bought_ItemsID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Item_Name varchar(255) NOT NULL,
        Last_Update datetime NOT NULL,
        CONSTRAINT Bought_Items_PK PRIMARY KEY (Bought_ItemsID)
);
```

- Cusotmer_Category

```
CREATE TABLE Customer_Category (
        Customer_CategoryID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        CustomerID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        CategoryCID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Last_Update datetime NOT NULL,
        CONSTRAINT Customer_Category_PK PRIMARY KEY (Customer_CategoryID),
        CONSTRAINT Customer_Category_FK FOREIGN KEY (CategoryCID) 
REFERENCES CategoryC(CategoryCID),
        CONSTRAINT Customer_Category_FK_1 FOREIGN KEY (CustomerID) 
REFERENCES Customer(CustomerID)
);
```

- CategoryC

```
CREATE TABLE CategoryC (
        CategoryCID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Name varchar(255) NOT NULL,
        Last_Update datetime NOT NULL,
        CONSTRAINT CategoryC_PK PRIMARY KEY (CategoryCID)
);
```

- City

```
CREATE TABLE City (
        CityID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        CountryID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        City_Name varchar(255) NOT NULL,
        Last_Update datetime NOT NULL,
        CONSTRAINT City_PK PRIMARY KEY (CityID),
        CONSTRAINT City_FK FOREIGN KEY (CountryID) REFERENCES
Country(CountryID)
);
```

- Country

```
CREATE TABLE Country (
        CountryID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
        Country_Name varchar(255) NOT NULL,
        Last_Update datetime NOT NULL,
        CONSTRAINT Country_PK PRIMARY KEY (CountryID)
);
```

Thank you for reading.