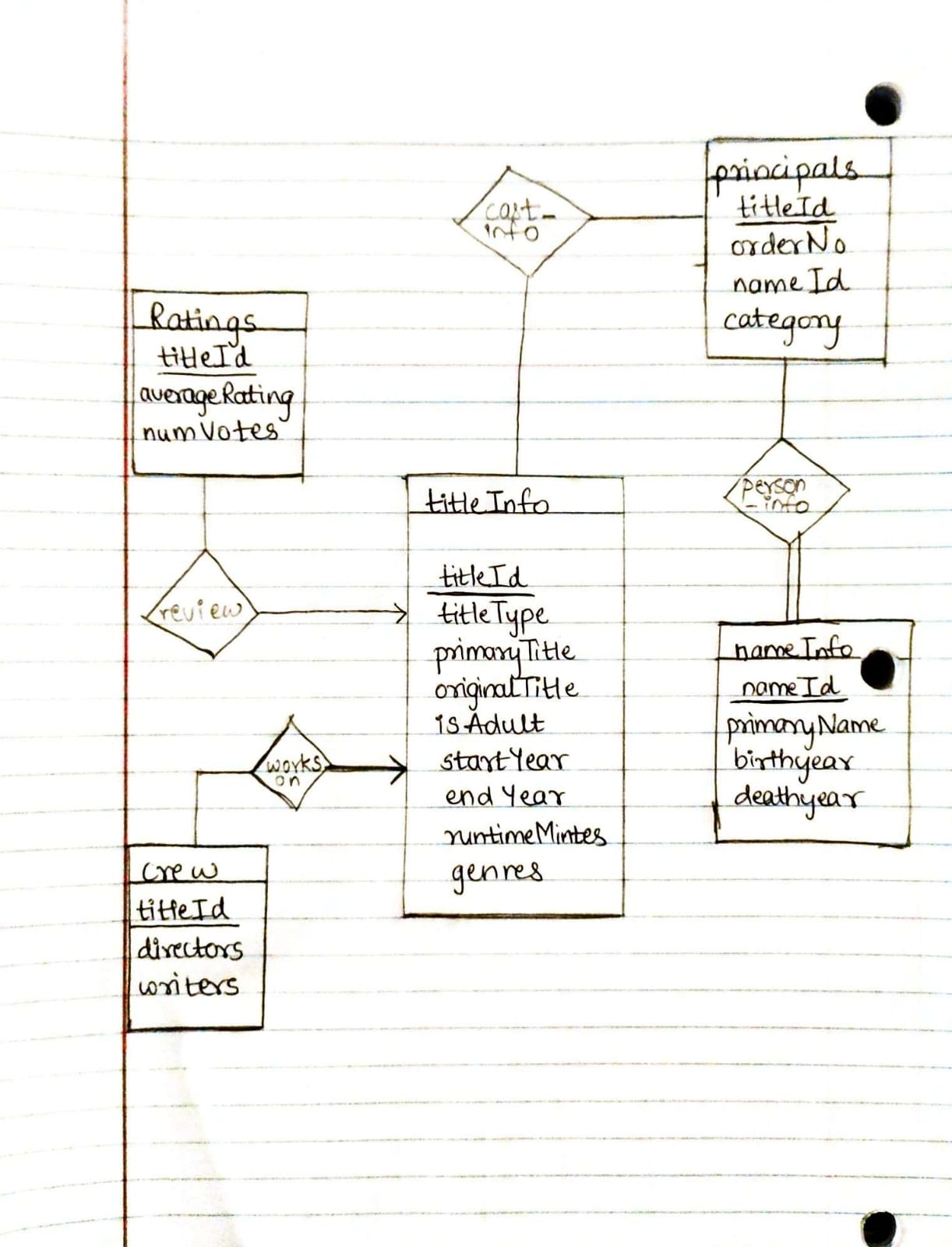
ASSIGNMENT 1- RELATIONAL MODEL

Q1 ER Diagram



Q2. SQL Scripts to CREATE TABLES:

1. Table 1: titleInfo

CREATE TABLE titleInfo(titleId VARCHAR(30),

titleType VARCHAR(50),

primaryTitle TEXT,

originalTitle TEXT,

isAdult BOOLEAN,

startYear INTEGER,

endYear INTEGER,

runTime INTEGER,

genres VARCHAR(250),

PRIMARY KEY(titleID));

1. Table 2: ratings

CREATE TABLE ratings(titleId VARCHAR(30),

averageRating FLOAT,

numVotes INTEGER,

PRIMARY KEY(titleId));

1. Table 3: crew

CREATE TABLE crew(titleId VARCHAR(30),

director TEXT,

writer TEXT,

PRIMARY KEY(titleId));

1. Table 4: nameInfo

CREATE TABLE nameInfo(nameID VARCHAR(20),

primaryName TEXT,

birthYear VARCHAR(5),

deathYear VARCHAR(5),

primaryProfession TEXT,

knownForProfession TEXT,

PRIMARY KEY(nameID));

1. Table 5: principals

CREATE TABLE principals(titleId VARCHAR(20),

orderNo INTEGER,

nameID VARCHAR(20),

category VARCHAR(50),

job TEXT,

charactersName TEXT,

PRIMARY KEY(titleId,orderNo));

The issues that were faced during creation of tables :

The data files had some additional attribute columns which were not required. So while creating the tables those attributes were considered and in the further steps where the data loading was done in the tables these additional columns were dropped from the table.

For example, the principals table contains the columns job and charactersName which are not required according to the designed ER diagram so these columns are dropped in the later stage. Similarly, columns primaryProfession and knownForProfession from nameInfo table are also dropped.

The primary key of every table should be of Integer type. But the provided data was of type varchar. When the tables were created the primary key was initially set to varchar type and then it was updated to integer type. The primary key of every table was updated in the same manner.

The reserved keywords used had to be changed. Ordering was changed to orderNo.

Name was changed to nameID.

Columns like primaryTitle,originalTitle in the titleInfo table were given the Text type as the data of these attributes was large for the varchar type.

Q3.Description of files in the database:

1. title.akas.tz.gz:

**titleId:**

This acts a unique identifier for the title

**ordering:**

This column consists of integers that uniquely identify the titleId.

**title**:

This column contains the title name for the corresponding titleId.

**region:**

This column contains the name of the region the title belongs to.

**language:**

This column has the name of the language the title is in.

**attributes:**

This column contains the information that describes the alternative title.

**isOriginalTitle:**

This column contains the values in boolean format. If the title is original then it is represented as 1 else 0.

1. title.basics.tsv.gz :

This file contains the information about the various titles, the category to which they belong, the release and the end year for that particular title,the duration and the genres to which they belong.

**tconst:**

This acts as the primary key for the table as the data in this column is unique for each of the titles.

**titleType:**

The title can be a movie, short film, TV series, TV episode, video. This column tells us about the category to which the title belongs.

**primaryTitle:**

This column consists of the names that were referred for that title during the release of that title.

**originalTitle:**

This column contains the original name of the title in its original language.

**isAdult:**

The values in this column are of boolean type. If a title is adult then it is represented as 1 and 0 for non-adult titles.

**startYear:**

This column contains the year in which the titles were released.

**endYear:**

The values in this column are only related to TV series. It gives the year in which the TV series ended and for the other titles it is represented as ‘\N’.

**runtimeMinutes:**

This column gives us the time or duration of each title in minutes.

**genres**:

This column contains the genres associated with the title. Each title has a maximum of three genres associated with it.

1. Title.crew.tsv.gz

This file contains information about the directors and writers associated with the title.

**tconst:**

This acts as the primary key for the table as the data in this column is unique for each of the titles

**directors:**

This column consists of the directors associated with each of the titles. There can be more than one director associated with a particular title.

**writers:**

This column gives information about the writers of a particular title. Each title can have one or more than one writer associated with it.

1. title.episode.tsv.gz:

This file contains all the information related to a TV series like episode number, season number, unique ID of the episodes and the ID of the series to which the episodes belong.

**tconst:**

This value is associated with the episodes and it uniquely identifies the episodes.

**parentTconst:**

This is the value that is associated with the parent TV series. Each episode id that is the tconst is related to the parentTconst as each episode will belong to a particular series.

**seasonNumber:**

This column contains values that represent the season number that a particular episode belongs to.

**episodeNumber:**

This column gives the episode number of the associated tconst value for the TV series.

1. title.principals.tsv.gz:

It contains information about the cast and crew related to the titles.

**tconst:**

This column uniquely identifies each of the title

**ordering:**

This column consists of integers which act as a unique identifier for tconst(titleId).

**nconst:**

This is the unique identity related to the person.

**Category:**

This column contains the category of job that the particular person had for the associated titleId.

**Job:**

This column shows whether a particular job title is applicable or not.

**Characters:**

This represents the name of the character that is played by the person associated with it. If the person has not played any characters then it is denoted by ’\N’.

1. title.ratings.tsv.gz:

This file contains information about the votes and ratings related to the titles

**tconst:**

This column uniquely identifies each of the title

**averageRating:**

This gives the weighted average that is calculated from the individual user ratings.

**numVotes:**

This column gives us the number of votes that are associated with a particular title.

1. name.basics.tsv.gz:

This file contains information about the people who are associated with the titles

**nconst:**

This column contains values that uniquely identify a person.

**primaryName:**

This contains the name of the person associated with the unique Id(nconst).

**birthYear:**

It gives the year in which the associated person was born.

**deathYear:**

It gives the death year of the person only if it is applicable else it is in the form ‘\N’.

**primaryProfession:**

This column gives the top three professions that an individual is associated with.

**knownForTitles:**

This column contains the titleIds (tconst) that a person has worked for

Q4 Loading data from files into database

1. Table 1: titleInfo

**Copying the title.tsv file into titleInfo table**

copy titleInfo from 'C:/Users/Revaa/Desktop/title.tsv' DELIMITER E'\t' NULL '\N' CSV HEADER ENCODING 'UTF8' QUOTE E'\b';

**Time:**

Query returned successfully in 2 min 18 secs.

**Updating the primary key titleId.This removes the ‘tt’ present in the beginning and gives only the integer part**

UPDATE titleInfo

SET titleId=substring(titleId,3);

**Time:**

Query returned successfully in 2 min 33 secs.

**Changing the type of titleId column from varchar to integer**

ALTER TABLE titleInfo

ALTER COLUMN titleId TYPE INTEGER

USING (titleId::INTEGER);

**Time:**

Query returned successfully in 1 min 5 secs.

1. Table 2: ratings

**Copying the rating.tsv file into titleInfo table**

copy ratings from 'C:/Users/Revaa/Desktop/rating.tsv' DELIMITER E'\t' NULL '\N' CSV HEADER ENCODING 'UTF8' QUOTE E'\b';

**Time:**

Query returned successfully in 15 secs 623 msec.

**Updating the primary key titleId.**

UPDATE ratings

SET titleId=substring(titleId,3);

**Time:**

Query returned successfully in 14 secs 991 msec.

**Changing the type of titleId column from varchar to integer**

ALTER TABLE ratings

ALTER COLUMN titleId TYPE INTEGER

USING (titleId::INTEGER);

**Time:**

Query returned successfully in 4 secs 322 msec.

**Setting the Foreign key**

The titleId of ratings column acts as a foreign key in reference to the titleId key of titleInfo table. First deleting the rows which do not have matching Id from the ratings table and then setting the foreign key.

DELETE from ratings where not exists(select null from titleinfo where ratings.titleId=titleInfo.titleId);

ALTER TABLE ratings ADD constraint fk\_ratings\_titleInfo

FOREIGN KEY (titleId)

REFERENCES titleInfo(titleId);

**Time:**

Query returned successfully in 20 secs 909 msec.

1. Table 3: crew

**Copying the crew.tsv file into crew table**

copy crew from 'C:/Users/Revaa/Desktop/crew.tsv' DELIMITER E'\t' NULL '\N' CSV HEADER ENCODING 'UTF8' QUOTE E'\b';

**Time:**

Query returned successfully in 1 min 47 secs.

**Updating the primary key titleId of the crew table.**

UPDATE crew

SET titleId=substring(titleId,3);

**Time:**

Query returned successfully in 2 min 8 secs.

**Changing the type of titleId column from varchar to integer**

ALTER TABLE crew

ALTER COLUMN titleId TYPE INTEGER

USING (titleId::INTEGER);

**Time:**

Query returned successfully in 40 secs 308 msec.

**Setting the Foreign key**

**The titleId of the crew table acts as a foreign key in reference to the titleId of the titleInfo table. First deleting the rows that do not match and then setting the foreign key.**

DELETE from crew where not exists(select null from titleinfo where crew.titleId=titleInfo.titleId);

ALTER TABLE crew ADD constraint fk\_crew\_titleInfo

FOREIGN KEY (titleId)

REFERENCES titleInfo(titleId);

**Time:**

Query returned successfully in 23 secs 821 msec.

1. Table 4: nameInfo

**Copying the name.tsv file into nameInfo table**

copy nameInfo from 'C:/Users/Revaa/Desktop/name.tsv' DELIMITER E'\t' NULL '\N' CSV HEADER ENCODING 'UTF8' QUOTE E'\b';

**Time:**

Query returned successfully in 2 min 39 secs.

**Updating the primary key titleId of the crew table.**

UPDATE nameInfo

SET nameId=substring(nameId,3);

**Time:**

Query returned successfully in 2 min 53 secs.

**Dropping the columns from nameInfo table**

ALTER TABLE nameInfo

DROP COLUMN primaryProfession,

DROP COLUMN knownForProfession;

**Time:**

Query returned successfully in 1 secs 40 msec.

**Changing the type of nameId column from varchar to integer**

ALTER TABLE nameInfo

ALTER COLUMN nameId TYPE INTEGER

USING (nameId::INTEGER);

**Time:**

Query returned successfully in 1 min 6 secs.

1. Table 5: principals

**Copying the principals.tsv file into principal table**

copy principals from 'C:/Users/Revaa/Desktop/principal.tsv' DELIMITER E'\t' NULL '\N' CSV HEADER ENCODING 'UTF8' QUOTE E'\b';

**Time:**

Query returned successfully in 6 min 35 secs.

**Updating the primary key nameId and titleId of the principal table.**

UPDATE principals

SET nameId=substring(nameId,3), titleId=substring(titleId,3);

**Time:**

Query returned successfully in 13 min 49 secs.

**Changing the type of titleId column and nameID column from varchar to integer**

ALTER TABLE principals

ALTER COLUMN titleId TYPE INTEGER

USING (titleId::INTEGER);

ALTER TABLE principals

ALTER COLUMN nameID TYPE INTEGER

USING (nameID::INTEGER);

**Time:**

Query returned successfully in 7 min.

**Dropping job and charactersName column**

ALTER TABLE principals

DROP COLUMN job,

DROP COLUMN charactersName;

**Time:**

Query returned successfully in 120 msec.

**This table has two foreign keys.**

**The titleId of the principals table is the foreign key in reference to the titleId of titleInfo table.**

DELETE from principals where not exists(select null from titleinfo where principals.titleId=titleInfo.titleId);

ALTER TABLE principals ADD constraint fk\_principals\_titleInfo

FOREIGN KEY (titleId)

REFERENCES titleInfo(titleId);

**Time:**

Query returned successfully in 1 min 31 secs.

**The nameId of the principals table is the foreign key in reference to the nameId of the nameInfo table.**

DELETE from principals where not exists(select null from nameInfo where principals.titleId=nameInfo.nameID);

ALTER TABLE principals ADD constraint fk\_principals\_nameInfo

FOREIGN KEY (nameID)

REFERENCES nameInfo(nameID);

**Time:**

Query returned successfully in 21 min 47 secs.

The issue faced during this stage was during the creation of foreign keys.

The attributes could not be directly assigned as foreign keys as there were some rows that were absent in the data for the matching key id from another table.

This issue was solved by first deleting the rows that had missing values and then creating the foreign key

The attributes which were not needed in the tables were dropped. The primary key of every table was earlier of the type varchar which was then updated to type integer.

Q5.

The query1 is for inserting the values into the titleInfo table. The values to be inserted in this query are valid and they do not exist in the table. This query is executed. The query2 tries to insert values into the titleInfo table but the titleId that is to be inserted already exists in the table.

So it would result in failure as duplicate keys are not allowed. The query3 would not get executed as the transaction is stopped. The changes that are made due to execution of query1 are rolled back and the database is in the same state as before.

import psycopg2

connection = None

try:

connection = psycopg2.connect(

host='localhost',

dbname='postgres',

user='postgres',

password='Revaa@25',

port=5432)

cur = connection.cursor()

query1 = '''insert into titleInfo(titleId,titleType,primaryTitle,originalTitle,isAdult,startYear,endYear,runTime,genres) values (10144565,'short','shortMovie','movie','false',1999,2003,22,'short'); '''

cur.execute(query1)

query2 = '''insert into titleInfo(titleId,titleType,primaryTitle,originalTitle,isAdult,startYear,endYear,runTime,genres) values (10144400,'movies','Frozen','Frozen','false',1992,2000,20,'action'); '''

cur.execute(query2)

query3 = '''insert into titleInfo(titleId,titleType,primaryTitle,originalTitle,isAdult,startYear,endYear,runTime,genres) values (10144555,'video','Fun','Fun','false',1990,1991,15,'drama'); '''

cur.execute(query3)

updated\_rows = cur.rowcount

connection.commit()

cur.close()

except Exception as error:

print(error)

print("Transaction Failed")

connection.rollback()

finally:

if connection is not None:

connection.close()