Lewis University  
CPSC 50900: Database Systems  
Term Project

OSCAR\_WORLD

Radhakrishna Kambalapelly, [radhakrishnarkamba@lewisu.edu](mailto:radhakrishnarkamba@lewisu.edu)

UmaMahesh Pasumarthy, [umamaheshpasumarth@lewisu.edu](mailto:umamaheshpasumarth@lewisu.edu)

Rajashekhar reddy Beeravolu

https://github.com/maheshuma1/DBMS

Table of Contents

[Initial Proposal 2](#_Toc85814569)

[Data Sources 2](#_Toc85814570)

[Data Storage Alternatives 2](#_Toc85814571)

[Relational Database Design Process 3](#_Toc85814572)

[Relational Database Design 3](#_Toc85814573)

[Data Definition Language (DDL) Scripts 3](#_Toc85814574)

[Data Manipulation Language Scripts 4](#_Toc85814575)

[Indexes 5](#_Toc85814576)

[Views 5](#_Toc85814577)

[Triggers 5](#_Toc85814578)

[Transactions 5](#_Toc85814579)

[Database Security 6](#_Toc85814580)

[Locking and Concurrent Access 6](#_Toc85814581)

[Backing Up Your Database 6](#_Toc85814582)

[Python Programming 7](#_Toc85814583)

[PHP Programming 7](#_Toc85814584)

[Suggested Future Work 8](#_Toc85814585)

[Activity Log 8](#_Toc85814586)

# Initial Proposal

*Description: You will describe the data you aim to store. What data will be storing? Why are you interested in this data? Why is it important? Where will the data come from? Who will use this data? What kind of application do you plan to build with it?*

*Rubric: Your response to each of these six questions will be graded out of 3 points.*

* *3 points: clear, complete descriptions that convey the importance and meaning of your data*
* *2 points: mostly clear descriptions, although some additional data would have helped in some sections*
* *1 point: necessary details are lacking in many of your responses.*

*You will also earn 2 additional points for coming up with a descriptive title for your project.*

*Total points possible: 20*

Oscars are the most respectable and highly valued awards in the film fraternity and in the cinema lover community. I will be storing the information of all the people who won or got nominated for the Oscars in all categories. usually, there are five to 6 nominations and only one of them wins.

As a movie lover i watch and appreciate a lot of movies and i like many other movie lovers want to know about the technicians and artists involved. Some of them are exceptional and needs to be appreciated and awarded for their service in the film and for the society. The list of people nominated for the Oscar assures best performance and people look for their names in a movie or get interested in a movie solely because of the involvements of those guys.

The people who gets an Oscar or gets nominated for the Oscars are the people who are at their best in their respective fields and they have every right to be known and discovered by many others for the exceptional service they provide. People also will have an idea on whom to checkout for not to be disappointed.

The data is found on the official websites ofoscars and in various other websites created by film enthusiasts. The information is widely spread, and it takes time to collect all the information from the year of its formation.

The people who are film enthusiasts and who are generally into watching movies can use this data to explore the work of great artists. This data helps them to connect the people with the art undiscovered by them and the art finds the people for which it is intended to.

I want to create an application which curates the movies for the people to watch the best content and suggests the viewers the best. This allows the subscribers to watch the art which they are seeking and they become dependent on the application to curate the list of movies for them.

# Data Sources

*Description of the datafiles:*

* *Details: The data collected for the database file contains all of nomination on the Academy Awards Oscar. The information includes all the data regarding the Oscar nominations and details of the awardee.*
* Origin: The data was collect from the (Kaggle.com). The data was initially in the form of the csv file later it was filtered into Json and xml. The raw data is processed and removed some unwanted columns.
* Structure : The files are structured into fields ,subfields , categories and later divided according to the year of nomination

Additional files to be included: I would like to include the field named “previously nominated “and “no of Oscars “ in the datafiles.

* I will sort out the data into various tables which will be easy to access and edit data.
* I will filter all fields and replace winner column with no of awards section.
* The data was manually separated in the excel and separated according to their respective fields.

# Data Storage Alternatives

*Description: We will study alternatives to storing data in a relational database. Some of the alternatives come from several decades ago, including the hierarchical and network models. Some are newer options, such as NoSQL databases that use JSON or some other encoding. Describe in detail how to store the data using two alternatives to relational databases. Be sure to describe how you would implement the alternatives and the advantages and disadvantages of each.*

* *For the first alternative, I’ve chosen the NoSQL database in particular the MongoDB*

*Database. In this database, the data is stored in the form of documents where each document holds the data corresponding to a single record in a relational database. These documents are then stored in separate storage container called ‘Collection’.*

*Advantages:*

* *This is schema less i.e.; we don’t need to define schema for every data that is being stored into the database.*
* *In MongoDB, the structure of every single object is very clear.*
* *It is very easy to scale.*
* *There are no complex joins in this database.*

*Disadvantages:*

* *One of the main disadvantages is the absence of the transactions. It doesn’t support transactions.*
* *Implementing joins in the mongo DB is much harder than in rdbms.*
* *For the second alternative, I’ve chosen the column data store especially the Cassandra database to store my data.It works in the same way as the rdbms but the data is stored sequentially column wise rather than the traditional row-wise. In this case, all the Oscar year field values are stored first and then it stores the next field vales i.e., the Oscar ceremony year and so on.*

*Advantages:*

* *Peer-to-peer architecture i.e., it is masterless database, and any server can entertain a request from any client.*
* *Elastically scalable i.e., any number of nodes can be added or deleted without much disturbance.*
* *Highly efficient performance under large sets of data.*

*Disadvantages:*

* *It doesn’t support the join function.*
* *One of the main downsides is it isn’t optimized for ACID properties.*

# Relational Database Design Process

*Description: Consider the list of fields you identified in part c. Identify functional dependencies that exist among them. For each functional dependency, identify the determinants and the fields they determine. This becomes the basis for identifying your entity sets, which then become your tables. Give each entity set or table you identify in this way a unique and clear name, making sure that the names you use are singular nouns. Then list the relationships that exist among the various entity sets. For each relationship, identify its connectivity (one-to-one, one-to-many, many-to-many) and participation (optional or mandatory). Finally, make sure that none of the attributes you've assigned to each entity set are multi-valued. If they are, take the steps needed to break them down.*

*Rubric: Your work will be graded as follows:*

* *8 points for identifying all the functional dependencies, including determinants and the columns whose values they determine.*
* *2 points for naming the entity sets that make up your data with clear, easy-to-understand names.*
* *6 points for identify the relationships among the entity sets and identifying connectivity and participation for each.*
* *2 points for breaking down multi-valued attributes.*

*Total points possible: 18*

ENTER YOUR RELATIONAL DATABASE DESIGN DESCRIPTION HERE. INCLUDE SOURCE CODE AND SCREEN SHOTS.

Here the functional dependencies are actor id->actor name, film id->film name, movie id->movie name ,and the determinants include actor id, actress id, movie id, Oscar held year, film id.

Different entity sets are Actor, Award winning movies, Oscar held, films Nominated, .

Actor-filmcast : one-to-many

Film\_cast-Film : one-to-one

Fillms-Films nominated : : one-to-many

Oscar held-Films nominate : one-to-many

# Relational Database Design

*Description: This is where you will complete your database design. For each of the entity sets you identified in the preceding section, analyze them to make sure they pass 2nd, 3rd, 4th, and Boyce-Codd Normal Form. If they do not, introduce additional entity sets or key changes to make sure that they do. Then, add foreign keys to connect entity sets that are related. For many-to-many relationships, introduce bridge entity sets to convert them into two one-to-many relationships. Also, consider whether you should introduce surrogate keys to create a more efficient primary key for some of your entity sets. Finally, diagram your design in Vertabello. Make sure your ER diagram correctly shows all entity sets, their primary and foreign keys, the data types for each attribute, and the connectivity and participation characteristics of each entity set. Your final Vertabello design should be something you could actually implement in a relational database management system.*

*Rubric: Your work will be graded as follows:*

* *4 points for the normalization analysis of your entity sets.*
* *3 points for introducing bridge entity sets.*
* *3 points for choosing foreign keys and perhaps more efficient surrogate keys*
* *10 points for correctly depicting your physical database model in Vertabello*

*Total points possible: 20*

ENTER YOUR RELATIONAL DATABASE DESIGN HERE

Foreign keys are actor id, actress id,Diagram

Description automatically generated film id, Oscar held year, movie id.

# Data Definition Language (DDL) Scripts

*Description: Use Vertabello to generate a script of SQL commands that build the database and its table structures. Write scripts or build Excel spreadsheets that take your data files and generate scripts of SQL insert statements from them. Use the MySQL source command to run the various scripts needed to build and populate the database in MySQL. Include the source code and / or Excel spreadsheets you use to manipulate and populate the data. Make sure all your tables have at least three records in them and that you've linked the tables through their foreign keys.*

*Rubric: Your work will be grades as follows:*

* *Database and table creation statements from Vertabello saved as ansql script file: 3 points*
* *Scripts you write or Excel spreadsheets you create to generate SQL commands for populating the tables, uploaded to GitHub: 8 points*
* *Descriptions of the scripts and Excel spreadsheets you wrote along with code excerpts included in the Word document: 5 points*
* *Screenshots of your successful attempts to use the MySQL source command to populate each table with at least three records: 4 points*

*Total points possible: 20*

ENTER YOUR DDL WORK HERE

the scripts contain records of various tables.

The actor tables insert statements, and all the remaining tables are created and populated using the insert statements.

-- Created by Vertabelo (http://vertabelo.com)

-- Last modification date: 2021-12-16 13:29:04.134

-- tables

-- Table: Actor

CREATE TABLE Actor (

Actor\_ID int NOT NULL,

Actor\_fname varchar(15) NOT NULL,

Actor\_lname varchar(10) NOT NULL,

Actor\_age int NOT NULL,

Actor\_Gender varchar(5) NOT NULL,

CONSTRAINT Actor\_pk PRIMARY KEY (Actor\_ID)

);

-- Table: Film

CREATE TABLE Film (

Film\_ID int NOT NULL,

Release\_year int NOT NULL,

CONSTRAINT Film\_pk PRIMARY KEY (Film\_ID)

);

-- Table: Film\_cast

CREATE TABLE Film\_cast (

Filmcast\_ID int NOT NULL,

Actor\_ID int NOT NULL,

Film\_ID int NOT NULL,

CONSTRAINT Film\_cast\_pk PRIMARY KEY (Filmcast\_ID)

);

-- Table: Films\_nominated

CREATE TABLE Films\_nominated (

nom\_id int NOT NULL,

Oh\_year int NOT NULL,

Film\_ID int NOT NULL,

CONSTRAINT Films\_nominated\_pk PRIMARY KEY (nom\_id)

);

-- Table: Oscar\_held

CREATE TABLE Oscar\_held (

Oh\_year int NOT NULL,

Awards\_given int NOT NULL,

People\_attended int NOT NULL,

Oh\_location varchar(20) NOT NULL,

CONSTRAINT Oscar\_held\_pk PRIMARY KEY (Oh\_year)

);

-- foreign keys

-- Reference: Film\_cast\_Actor (table: Film\_cast)

ALTER TABLE Film\_cast ADD CONSTRAINT Film\_cast\_Actor FOREIGN KEY Film\_cast\_Actor (Actor\_ID)

REFERENCES Actor (Actor\_ID);

-- Reference: Film\_cast\_Film (table: Film\_cast)

ALTER TABLE Film\_cast ADD CONSTRAINT Film\_cast\_Film FOREIGN KEY Film\_cast\_Film (Film\_ID)

REFERENCES Film (Film\_ID);

-- Reference: Films\_nominated\_Film (table: Films\_nominated)

ALTER TABLE Films\_nominated ADD CONSTRAINT Films\_nominated\_Film FOREIGN KEY Films\_nominated\_Film (Film\_ID)

REFERENCES Film (Film\_ID);

-- Reference: Films\_nominated\_Oscar\_held (table: Films\_nominated)

ALTER TABLE Films\_nominated ADD CONSTRAINT Films\_nominated\_Oscar\_held FOREIGN KEY Films\_nominated\_Oscar\_held (Oh\_year)

REFERENCES Oscar\_held (Oh\_year);

insert into Actor values(125,'Mahesh','Babu',45,'M');

insert into Actor values(126,'Leonardo','DeCaprio',56,'M');

insert into Actor values(127,'Tony','Stark',50,'M');

insert into Actor values(100,'heli','margarate',39,'F');

insert into Actor values(101,'Kate','Winslet',30,'F');

insert into Actor values(102,'molly','james',25,'F');

insert into Film values(1,'titanic',2000);

insert into Film values(2,'return of the apes',2005);

insert into Film values(3,'beast',2002);

insert into Film values(4,'The Jungle Book',2010);

insert into film\_cast values(120,126,1);

insert into film\_cast values(122,102,3);

insert into film\_cast values(123,100,2);

insert into film\_cast values(124,125,4);

insert into Oscar\_held values(2001,30,5000,'sydney');

insert into Oscar\_held values(2004,30,4500,'Hollywood');

insert into Oscar\_held values(2006,30,6000,'paris');

insert into Films\_nominated values(12,2004,1);

insert into Films\_nominated values(13,2006,3);

insert into Films\_nominated values(14,2001,2);

Graphical user interface, application

Description automatically generated

-- End of file.

# Data Manipulation Language Scripts

*Description: Write the SQL commands for twelve queries. Two queries should be insert statements, two should update statements, one should be a delete statement, one should be a simple select statement that selects a subset of the rows and columns from one table, two should be a select statements that select data from a joining of two tables, two should use summary functions to generate statistics about the data, one should be a multi-table query, and one should be another query of your choice. Show the queries and screenshots of the results in your Word document, and save your queries in a commented sql script to GitHub.*

*Rubric: Your work will be graded as follows:*

* *1 point each for the two insert statements*
* *1 point each for the two update statements*
* *1 point for the delete statement*
* *1 point for the simple select statement*
* *2 points each for the 2 join statements*
* *2 points each for the two that use summary statements*
* *2 points for the multi-table query*
* *2 points for the query of your choice.*
* *12 points for showing the query and a screenshot of the corresponding result set back-to-back for each of these queries in your Word document.*

*Total points possible: 30*

ENTER DML WORK HERE

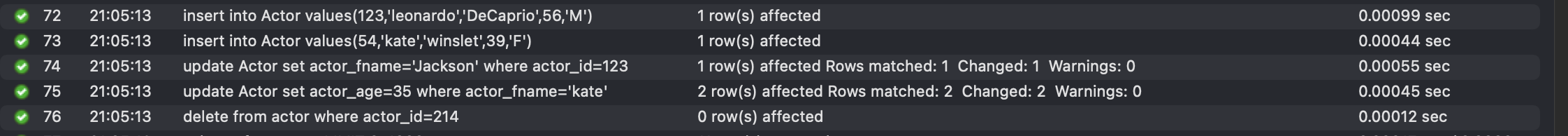
insert into Actor values(123,'leonardo','DeCaprio',56,'M');

insert into Actor values(54,'kate','winslet',39,'F');

update Actor set actor\_fname='Jackson' where actor\_id=123;

update Actor set actor\_age=35 where actor\_fname='kate';

delete from actor where actor\_id=214;



select \* from actor;

Table

Description automatically generated

SELECT a.actor\_id, a.actor\_fname

from Actor a

INNER join film\_cast fc on a.Actor\_id=fc.Actor\_ID;

Table

Description automatically generated

select a.actor\_fname, a.actor\_gender

from actor a

INNER join film\_cast fc on a.Actor\_id=fc.Actor\_ID and a.actor\_gender='F';

Table

Description automatically generated

select actor\_id,oh\_year from Actor a,Oscar\_held oh;

Background pattern

Description automatically generated

# Indexes

*Description: Improve the performance of your design by adding indexes to various tables. Show the SQL needed to add the indexes. Explain why you chose the ones you added. Explain how you would demonstrate the impact the indexes had on the performance of various queries.*

*Rubric: Your work will be graded as follows:*

* *6 points for clearly defining at least three indexes and explaining why you chose them.*
* *3 points for showing the sql needed to generate the indexes*
* *3 points for explaining how you would demonstrate the performance improvement afforded by the indexes.*

*Total points possible: 12*

ENTER YOUR INDEX WORK HERE

I’ve chosen my three indexes on actor first name,Film id and Oscar held year since I think that these will be the most searched in the database.

Everyone is interested in knowing the actor name who has done execellent acting hence placing an index on it will be much efficient to search them. Likewise the various Films attached to their Film id helps in faster fetching. Oscar held helps more efficient searches on the diversely held Oscar ceremonies.

Graphical user interface, application, Teams

Description automatically generated

# Views

*Description: Add two views to your database to provide easy access to combinations of data from multiple tables.*

*Rubric: Your work will be graded as follows:*

* *2 points for including the SQL for generating the two views in your Word document*
* *2 points for including screenshots for the data contained in each view in your Word document*
* *2 points for explaining why each view is a valuable addition to your database*
* *2 points for explaining who might benefit most from having access to each view.*

*Total points possible: 8*

ENTER YOUR WORK WITH VIEWS HERE

create view [Oscar\_movies] as

select awm.film\_name, a.act\_fname, at.actress\_fname, awm.Oscar\_held\_oh\_year

from Award\_Winning\_movies awm, Actor a, Actress at, Oscar\_held o, film\_cast fc

where awm.movie\_id=fc.film\_id and a.act\_id=fc.Actor\_act\_id and at.Actress\_id=fc.Actress\_Actress\_id;

create view [Oscar\_movies\_premiere] as

select awm.film\_name, fn.premiere\_yr

from Award\_Winning\_movies awm, Films\_nominated fn

where awm.movie\_id=fn.Award\_Winning\_movies\_movie\_id;

I’ve created views on award winning movie’s cast and its premiere dates

Graphical user interface, text, application

Description automatically generated

These views are useful for viewing the films and their premiere dates. While the 1st view is for viewing the various films that are awarded and their cast.

# Triggers

*Description: Add a trigger to a table so that data will be updated when a certain event occurs*

*Rubric: Your work will be graded as follows:*

* *2 points for including the SQL for the trigger in your Word document*
* *2 points for clearly explaining the purpose of the trigger*
* *2 points for a screenshot and explanation that shows the trigger in action.*

*Total points possible: 6*

ENTER YOUR WORK WITH TRIGGERS HERE

delimiter //

create trigger del\_rec after delete

on Film

for each row

select old.Film\_name+" deleted!" as output;

delimiter ;

this trigger is for informing whether the selected record is deleted or not.



# Transactions

*Description: Demonstrate that you know how to define and use a transaction. Why are transactions important for ensuring ACID behavior?*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly explaining the importance of transactions to ensuring ACID behavior*
* *3 points for including a screenshot and accompanying explanation of a MySQL transaction.*

*Total points possible: 6*

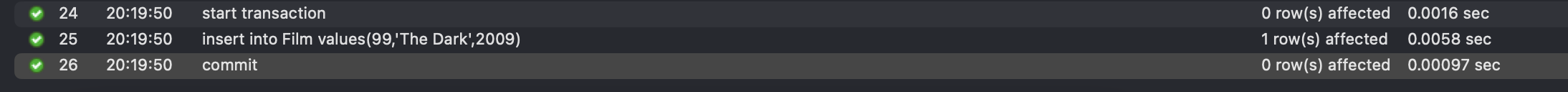
ENTER YOUR WORK WITH TRANSACTIONS HERE

Transactions are very important in operating or accessing the database because by default a lock will be placed on the working table whenever we start a transaction. This helps in maintain the acid properties which are

* Atomicity – there are no sequential changes. All are done in a single step
* Consistency – the data doesn’t changes before and after a transaction.
* Isolation – every transactions occurring is isolated. Hence no interruptions will take place.
* Durability – after the transaction the data is durable.

Graphical user interface, text

Description automatically generated



In this transaction, a record is being inserted into film table using the transaction.

# Database Security

*Description: Identify the different kinds of users who will use your database. Write GRANT statements to define the privileges for these different kinds of users.*

*Rubric: Your work will be graded as follows:*

* *6 points for clearly identifying and describing the various kinds of users who will use the databases and identifying and justifying what privileges each should have.*
* *4 points for writing GRANT statements that assign privileges to these different kinds of users.*
* *4 points for demonstrating with screenshots that your GRANT statements do distinguish among different kinds of users in regard to what they can do with the database.*

*Total points possible: 14*

ENTER YOUR WORK WITH DATABASE SECURITY HERE

The various users are

Oscar administrator, who manages the whole oscarInfo database and sees to the proper working of it.

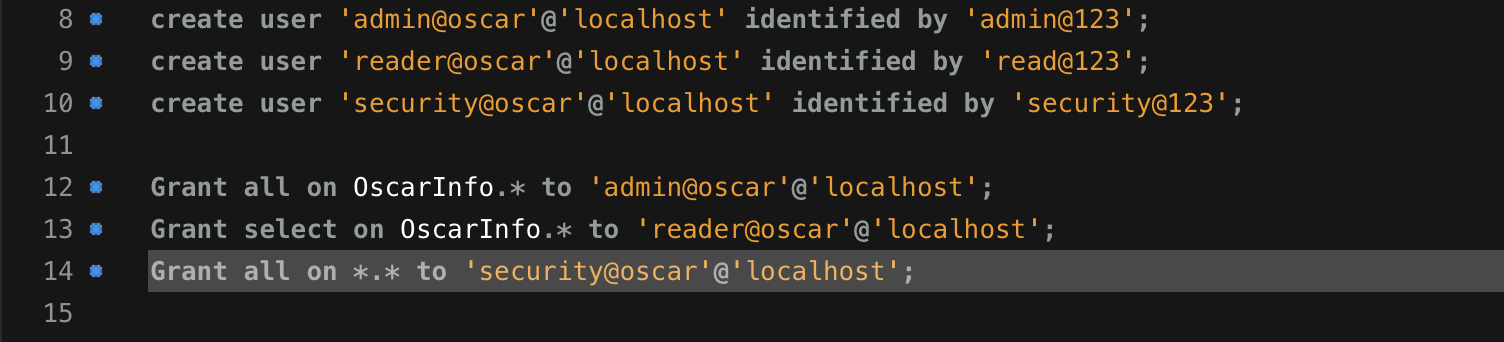
Grant all on OscarInfo.\* to ‘admin@oscar’@’localhost’;

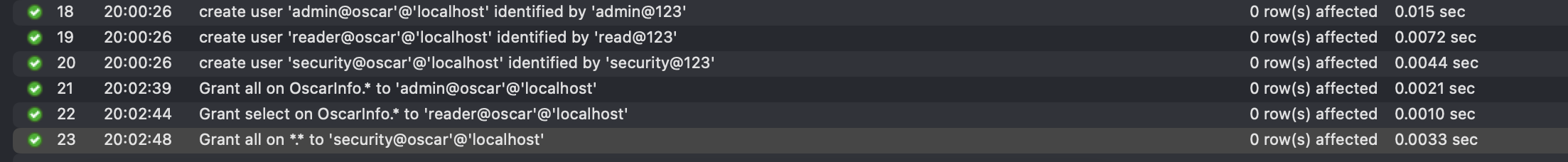
Reader, who proofreads the total database and points to any data errors.

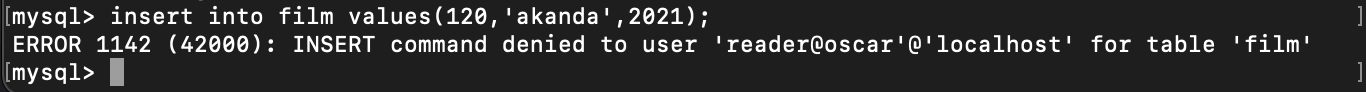
Grant select on OscarInfo.\* to ‘reader@oscar’@’localhost’;

Security manager, who sees to the security of the all databases.

Grant all on \*.\* to ‘security@oscar’@’localhost’;







Here, for user read@oscar insert is denied as we have specified in the grant operations.

# Locking and Concurrent Access

*Description: Explain the purpose of locking tables and show how to do that to prevent inconsistencies that may arise in your data when concurrent transactions take place.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly explaining an example that shows why you should lock tables to prevent inconsistencies.*
* *3 points for providing a screenshot and accompanying explanation of locking tables.*

*Total points possible: 6*

ENTER YOUR WORK WITH LOCKING AND CONCURRENT ACCESS HERE

Suppose there are two delete actions that are being executed on a particular table, but the table contains only one record. Hence, in this case an error occurs which is there will be no more records to delete after the other delete action executes.

Text

Description automatically generated

Here the actor table and film\_cast table are locked and open for only reading. Hence whenever we try to insert into these tables it shows an error.

# Backing Up Your Database

*Description: How you will back up your database. What commands will you issue? How frequently will the commands run? How can they be automated? Where will the backups be stored?*

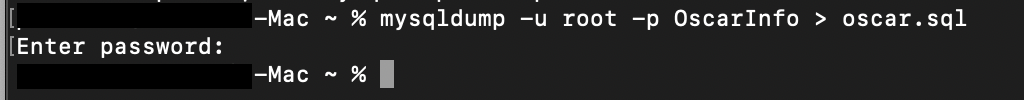
*Rubric: Your work will be graded as follows:*

* *12 points for clearly explaining and justifying your database backup strategy, including the frequency with which you will back up the database, how you will automate backups, where you will store them, and how you will secure them. You will earn three points for addressing each factor (frequency, location, automation, and security)*
* *3 points for providing a screenshot of the command you would issue to back up the database and for including a portion of the resulting file.*

*Total points possible: 15*

ENTER YOUR WORK ON DATABASE BACKUPS HERE

I’ve would back up my database once for every week and to a location different than where the actual database resides. Because if the disk containing the actual database crashes, this will wont crash.



Graphical user interface, text, application

Description automatically generated

# Python Programming

*Description: Write a Python program that generates a report that contains a subset of the data from your database. Include the code for your Python program in your Word document, and also post the program to your GitHub repository.*

*Rubric: Your work will be graded as follows:*

* *12 points for writing a Python script (and including its code in the Word doc) that will pull data from a database and store it to a text file and present it to the screen. Your code must have comments in it that explain how it works. You will be awarded 3 points for successfully connecting to the database, 3 points for successfully querying it, and 4 points for presenting the data to the screen and to a file. Internal comments count for 2 points.*
* *2 points for posting the code to GitHub*
* *4 points for showing a screenshot of your running the script and showing the results it produces on the screen.*

*Total points possible: 18*

ENTER YOUR PYTHON DATABASE PROGRAMMING WORK HERE

import mysql.connector

from mysql.connector import errorcode

from mysql.connector.errors import DataError

# Method to get the records from the database to store in a file and to present it.

*def* WriteRows(*conn*,*Query*,*file*):

cur = *conn*.cursor()

cur.execute(*Query*)

q = *Query*.split(' ')

*file*.write(q[3]+" Table:\n")

print(q[3]+" Table:")

for rec in cur.fetchall():

line=""

for r in rec:

line+=str(r)+","

*file*.write(line+"\n")

print(line)

*file*.write("\n")

print("\n")

try:

# Connecting to Database

conn = mysql.connector.connect(

*user* = "root",

*password* = "root@123",

*host* = "localhost",

*database* = "OscarInfo"

)

print("database successfully connected!")

# Opening a file to write

file = open('Dataset.txt','w')

print("File successfully opened!")

# Querying the Tables

WriteRows(conn,"select \* from Actor",file)

WriteRows(conn,"select \* from Film",file)

WriteRows(conn,"select \* from Film\_cast",file)

WriteRows(conn,"select \* from Films\_nominated",file)

WriteRows(conn,"select \* from Oscar\_held",file)

print("Dataset created into the file successfully!")

except mysql.connector.Error as err:

print("Error connecting to database")

exit()

except IOError:

print("File not found or path incorrect")

exit()

finally:

conn.close()

output:

Text

Description automatically generated Text

Description automatically generated

# PHP Programming or Java Program

*Description: Either*

1. *Build an HTML form that enables the user to specify criteria to search by. Use PHP to show the results of the query on a resulting web page. Make sure you include protections against an SQL injection attack. Include your HTML and PHP code in your Word document, and also post the files to your GitHub repository.*

*Rubric: Your work will be graded as follows:*

* + *4 points for writing an HTML form the user will use to enter search criteria*
  + *8 points for a PHP script that uses the search criteria and returns results*
  + *4 points for an HTML page that shows the results*
  + *4 points for explaining what SQL injection might be run on your website and explaining how you prevented it.*
  + *4 points for providing screen shots of your PHP website in action.*
  + *2 points for posting your code to GitHub*

*or*

1. *Build a Java program that manages the content of your database. It should be a menu-driven program that gives the user the option to insert a new record into one of your tables, update a record from one of your tables, delete a record from one of your tables, or select records from one of your tables according to multiple search criteria the user enters.*

*Rubric: Your work will be graded as follows:*

* + *4 points for repeatedly showing the menu of options until the user decides to quit*
  + *4 points for implementing the insert operation*
  + *4 points for implementing the update operation*
  + *4 points for implementing the delete operation*
  + *8 points for implementing the select operation with multiple criteria*
  + *2 points for posting your code to GitHub*

*Total points possible: 26*

ENTER YOUR PHP or JAVA DATABASE APP PROGRAMMING WORK HERE

import **java**.**io**.\*;

import **java**.**sql**.\*;

public class **DatabaseConn** {

public static void **main** (**String** args[]) throws **Exception** {

**BufferedReader** br = new **BufferedReader**(new **InputStreamReader**(**System**.in));

**String** url = "jdbc:mysql://localhost:3306/OscarInfo?useSSL=false";

**String** user = "root";

**String** pass = "root@123";

**Connection** conn=null;

try {

**Class**.**forName**("com.mysql.cj.jdbc.Driver");

conn = **DriverManager**.**getConnection**(url, user, pass);

**Statement** stt = conn.**createStatement**();

int op;

do {

**System**.out.**println**();

**System**.out.**println**("1.Inserting record");

**System**.out.**println**("2.Updating record");

**System**.out.**println**("3.deleting record");

**System**.out.**println**("4.Selecting record");

**System**.out.**println**("5.Exit");

**System**.out.**print**("Option:");

op = **Integer**.**parseInt**(br.**readLine**());

switch(op) {

case 1:{

**System**.out.**println**("Inserting into Film table record: \nFilm name : Spider-Man");

**String** query = "insert into Film values(10,'Spider-Man',2010)";

int n = stt.**executeUpdate**(query);

**System**.out.**println**(n>0?"Successfully inserted the record":"insertion failed");

break;

}

case 2: {

**System**.out.**println**("Updating the film table record\nchanging the release year of spider\_man movie");

**String** que = "update Film set release\_year = 2012 where Film\_ID=10";

int x = stt.**executeUpdate**(que);

**System**.out.**println**(x>0?"Successfully updated the record":"updation failed");

break;

}

case 3:{

**System**.out.**println**("Deleting the film table record\nFilm name : spider\_man");

**String** que = "delete from Film where Film\_ID=10";

int x = stt.**executeUpdate**(que);

**System**.out.**println**(x>0?"Successfully deleted the record":"deletion failed");

break;

}

case 4:{

**System**.out.**println**("Selecting the actor table records where actor is female and age<35");

**String** que = "select f.Film\_name, a.Actor\_fname, a.Actor\_lname, f.release\_year from Film f, Actor a, Film\_cast fc where fc.Actor\_ID=a.Actor\_ID and fc.Film\_ID=f.Film\_ID;";

**System**.out.**println**("Film name\tFirst Name\tLast name\tPremiere year");

**ResultSet** rs = stt.**executeQuery**(que);

while(rs.**next**()){

**System**.out.**println**(rs.**getString**(1)+"\t"+rs.**getString**(2)+"\t"+rs.**getString**(3)+"\t"+rs.**getInt**(4));

}

*//System.out.println(n>0?"Successfully updated the record":"updation failed");*

break;

}

}

}while(op != 5);

}

catch(**SQLException** e) {

**System**.out.**println**(e);

}

catch(**ClassNotFoundException** e) {

**System**.out.**println**(e);

}

finally{

conn.**close**();

br.**close**();

}

}

}

Text

Description automatically generated

# Suggested Future Work

*Description: Describe the limitations of your current database and explain how you or someone else could improve the design to address these shortcomings. Also describe how you might take advantage of leverage cloud services to increase the performance and availability of your database. Finally, explain the advantages and disadvantages of storing your data in a NoSQL format instead.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly describing the limitations of your databases*
* *3 points for explaining how you would address these shortcomings*
* *3 points for explaining how you might migrate the database to the cloud and describing what advantages you might gain from doing that.*
* *3 points for explaining the advantages and disadvantages of storing your data in a document-based NoSQL format instead.*

*Total points possible: 12*

ENTER YOUR SUGGESTED FUTURE WORK IDEAS HERE

* The limitations of my database are that it is not properly connected and assimilated correctly.
* These shortcomings can be resolved by implementing correct views and creating additional tables if necessary.
* I’ve would move the database as is into the cloud. Doing this results in low operational costs, high availability, end-to-end security and automation.
* The advantages include
  + Easy updates to tables and schema
  + Can store huge amounts of data
  + Developer friendly
* The disadvantages are
  + No proper backup
  + Lack of ample consistency

# Activity Log

*Description: As an appendix, the team will keep a daily diary or log of their activity. What did you or your team study in this class each day? What did you learn? What did you accomplish or build or design? You don't have to enter something every day, but there should be at least three entries each week. Since we have eight weeks, that means you should make 3 posts to the Activity Log each week, for a total of at least 24 posts. Each post will be worth 1 point.*

*If you are working as part of a team, make sure you clearly identify which team member worked on which tasks. The Activity Log should help me figure out how each team member contributed to the project. If I cannot discern who worked on what aspects of the project from the activity log, no points will be awarded for it.*

*Total points possible: 24*

1. Radha Krishna has successfully collected the data from the sources.

2. Uma mahesh is working on cleaning the data

3. Rajashekar studied about data storage alternatives

4. Radha Krishna is working on relational database design process

5.Uma Mahesh is working on relational database design process

6.Rajashekar chose MongoDB and Graph as databases

7. Uma Mahesh created relational database design using vertabelo

8. Radha Krishna learning Vertabelo software and working on creation of relational database design

9. Rajashekar created relational database design using vertabelo

10. Radha Krishna is working on DDL Scripts

11. Uma Mahesh is working on DML Scripts

12. Rajashekar learnt index part and adding indexes to the tables

13. Radha Krishna added views to the table

14. Uma Mahesh is working on adding triggers to the table

15. Rajashekar adding transactions to tables

16. Radha Krishna is working on looking and concurrent access

17. Uma Mahesh is writing GRANT statements for Database Security

18.Rajashekar is working on backup of the database

19. Radha Krishna is working on python source code

20. Rajashekar working on java code

21. Uma Mahesh gathered information on suggested future works

22. Radha Krishna successfully executed the python code

23. Uma Mahesh has completed the documentation part

24. Rajashekar has done the java code part successfully