Database Design, CSCI 340, Spring 2016
SQL, Transactions, April 15

Previously everyone in the class used the mysql account:

Username: csci340User
Password: csci340Pass

Personal mysql accounts have been created for you. I used the first letter of your first name followed by your last name for both the username and password. For example, for me:

Username: cschahczenski
Password: cschahczenski

Your personal mysql account has access to a database, with the same name. Within that database, you can create tables, views, indexes and grant privileges to other mysql users, on tables and views in the database.

Login to katie. From katie, login to your personal mysql account, and see what databases you have access to.

```
mysql –u cschahczenski –p
cschahczenski

SHOW DATABASES;
```

Use the database with your name and check that you can create tables within it.

```
USE cschahczenski;
CREATE TABLE TEST (testID INT PRIMARY KEY);

SHOW TABLES;
```

Recall that transactions are a logical unit of work which takes a database from a consistent state to a consistent state. (The database may be in an inconsistent state during the transaction.)

1. DBMS typically have multiple ways in which data can be stored. They can be stored using ordered/unordered flat files, ISAM, heap files, hash buckets, or B+ trees. Each form has its own particular advantages and disadvantages. The most commonly used forms are B+ trees and ISAM.

   To see the storage options for MySQL, give the following command within mysql.

   ```
   SHOW ENGINES;
   ```

   How may storage options are there for MySQL?

   9

2. To see what storage engine is used for a table:

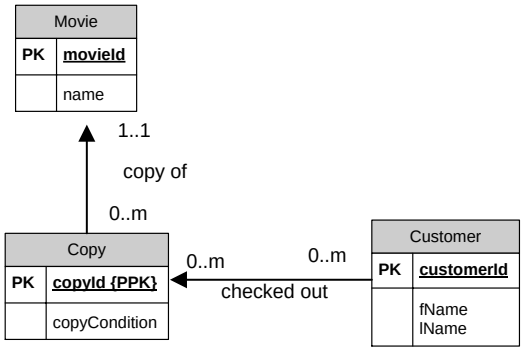
```
SELECT ENGINE
FROM information_schema.TABLES
WHERE TABLE_SCHEMA = 'databaseName'
AND TABLE_NAME = 'tableName'
```

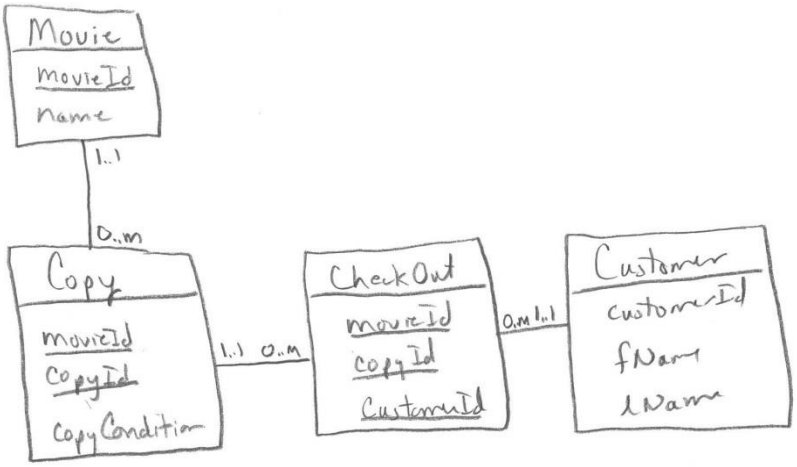
What is the default storage engine for our implementation of mysql?

InnoDB

3. Create a logical model for the following conceptual model.

Video Store –
Only storing who
checked out what





4. Get the script from the website and use it to create tables in your database. Check that the tables are inserted correctly. Add two new movies into the database, also with a copy of each of these movies.

   What happens if you try inserting a copy of a movie into the database where the movie doesn't exist? Write the error statement here.

   ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`cmschahczenski`.`Copy`, CONSTRAINT `movieConst` FOREIGN KEY (`movieId`) REFERENCES `Movie` (`movieId`) ON UPDATE CASCADE)

Say that you are the boss and database administrator for a video rental store. When new copies of movies are purchased, you are the one who will add these to the database. Other employees of the video store can see if a copy of a movie is available and they can check movies out and back in.

5. You have decided to create a view for your employees to see a list of the movies in the store and the number of copies. Write the SELECT query which you would use to obtain the information below. Note that in rare cases, you have entered a movie into the database, but you don't yet have copies of the movie. These movies should appear in the list.

   Here are the results using the database extension given in the script.

```
+------------+------------------+
| Movie      | Number of Copies |
+------------+------------------+
| Inception  |                0 |
| The Butler |                1 |
| Star Wars  |                2 |
+------------+------------------+
3 rows in set (0.00 sec)
```

```
SELECT name as 'Movie', COUNT(copyId) as 'Number of Copies'
FROM Movie LEFT OUTER JOIN Copy ON Movie.movieId=Copy.movieId
GROUP BY Movie.movieId;
```

6. Write the SQL statements to create the view called AllMovies.

```
CREATE VIEW AllMovies AS
       SELECT name as 'Movie', COUNT(copyId) as 'Number of Copies'
       FROM Movie LEFT OUTER JOIN Copy ON
              Movie.movieId=Copy.movieId
      GROUP BY Movie.movieId;
```

7. Since employees will be helping customers check out and return movies, you want to create a view for them to see what copies of movies are available for checkout. Write the SELECT query to obtain this information.

   Here are the results using the database extension given in the script.

```
+------------+---------+--------+
| name       | movieId | copyId |
+------------+---------+--------+
| The Butler |       2 |      1 |
+------------+---------+--------+
1 row in set (0.00 sec)
```

```
SELECT name AS 'name', Movie.movieId as "movieId", copyId as "copyId"
FROM Movie LEFT OUTER JOIN Copy ON Movie.movieId=Copy.movieId
WHERE (Movie.movieId, copyId) NOT IN
       (SELECT movieId, copyId FROM CheckOut);
```

8. Write the SQL statements to create the view called MoviesAvailable.

```
CREATE VIEW MoviesAvailable AS
SELECT name AS 'name', Movie.movieId as 'movieId', copyId as 'copyId'
FROM Movie LEFT OUTER JOIN Copy ON Movie.movieId=Copy.movieId
WHERE (Movie.movieId, copyId) NOT IN (SELECT movieId, copyId FROM
CheckOut);
```

9. Select someone else in the class to work with. Say that this person is one of your employees. Try granting SELECT privileges to that person on the AllMovies view suing the following.

```
GRANT SELECT
ON databaseName.tableOrViewName
TO otherUserName;
```

   Is your partner able to see your tables? How are the tables identified?

```
GRANT SELECT
ON cmschahczenski.AllMovies
TO partner;
```

10. Grant the same person SELECT, INSERT, UPDATE and DELETE privileges on the CheckOut table using the following statement.

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON databaseName.tableOrViewName
TO otherUserName;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON cmschahczenski.MoviesAvailble
```

```
TO partner;
```

Experiment with transactions where you and your partner both perform check outs on the same movies. Use the commands:

- BEGIN WORK; (rather than BEGIN TRANSACTION;)
- COMMIT;
- ROLLBACK;

Other useful commands:

- SAVE savePointName;
- ROLLBACK TO savePointName;

BEGIN WORK;
….
SAVEPOINT savePointName;
….

How are conflicts handled?