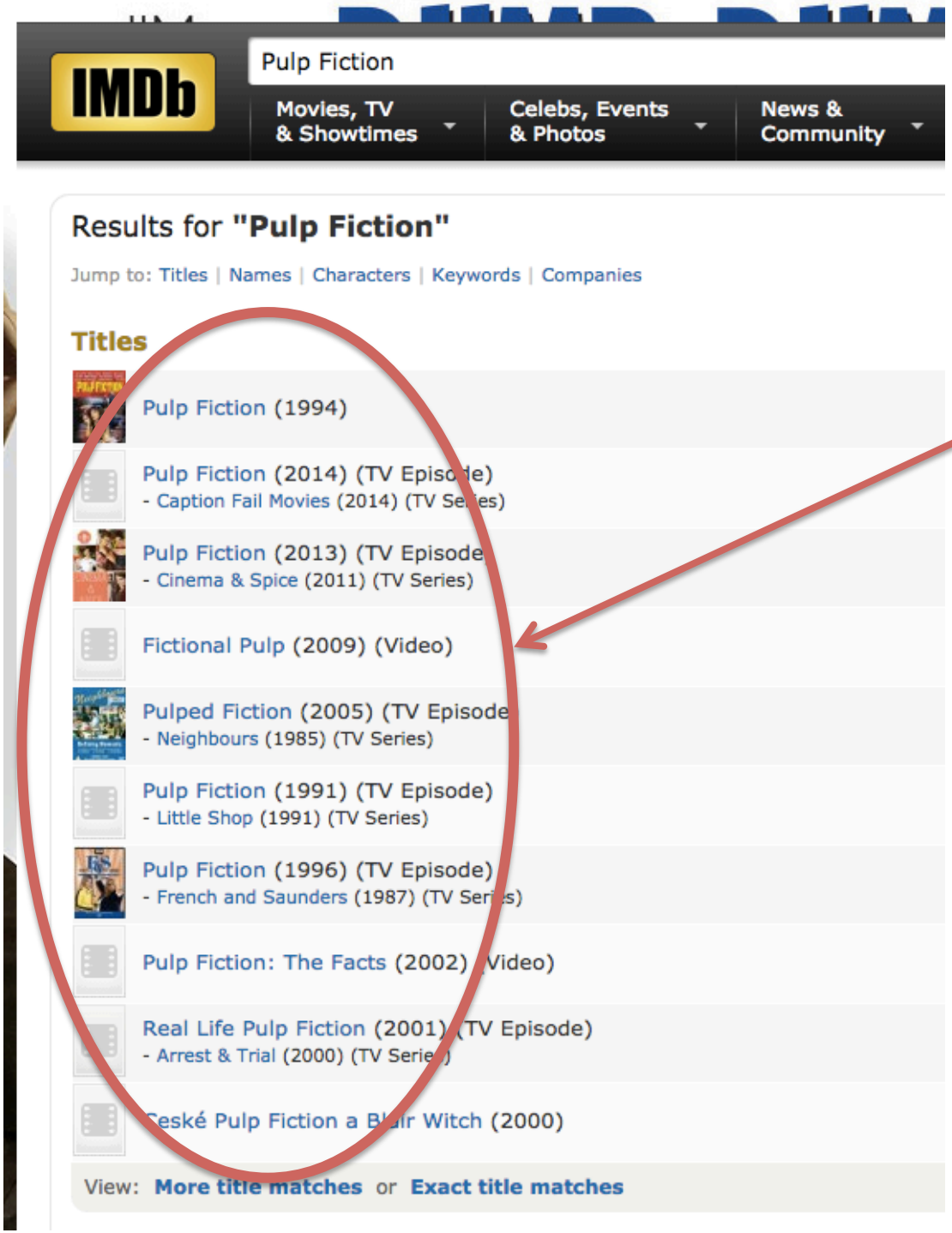
# CSE 344
# Section 8 [Srini]

1. Talking to Databases using Java
2. Surviving Homework 7




Where do these results come from?

And how are they retrieved?

# Homework 7 (Out today)

- 1. Video Rental Store
- 2. Create a command line program
- 3. Have user accounts
- 4. Allow users to search for movies from IMDB and rent them out

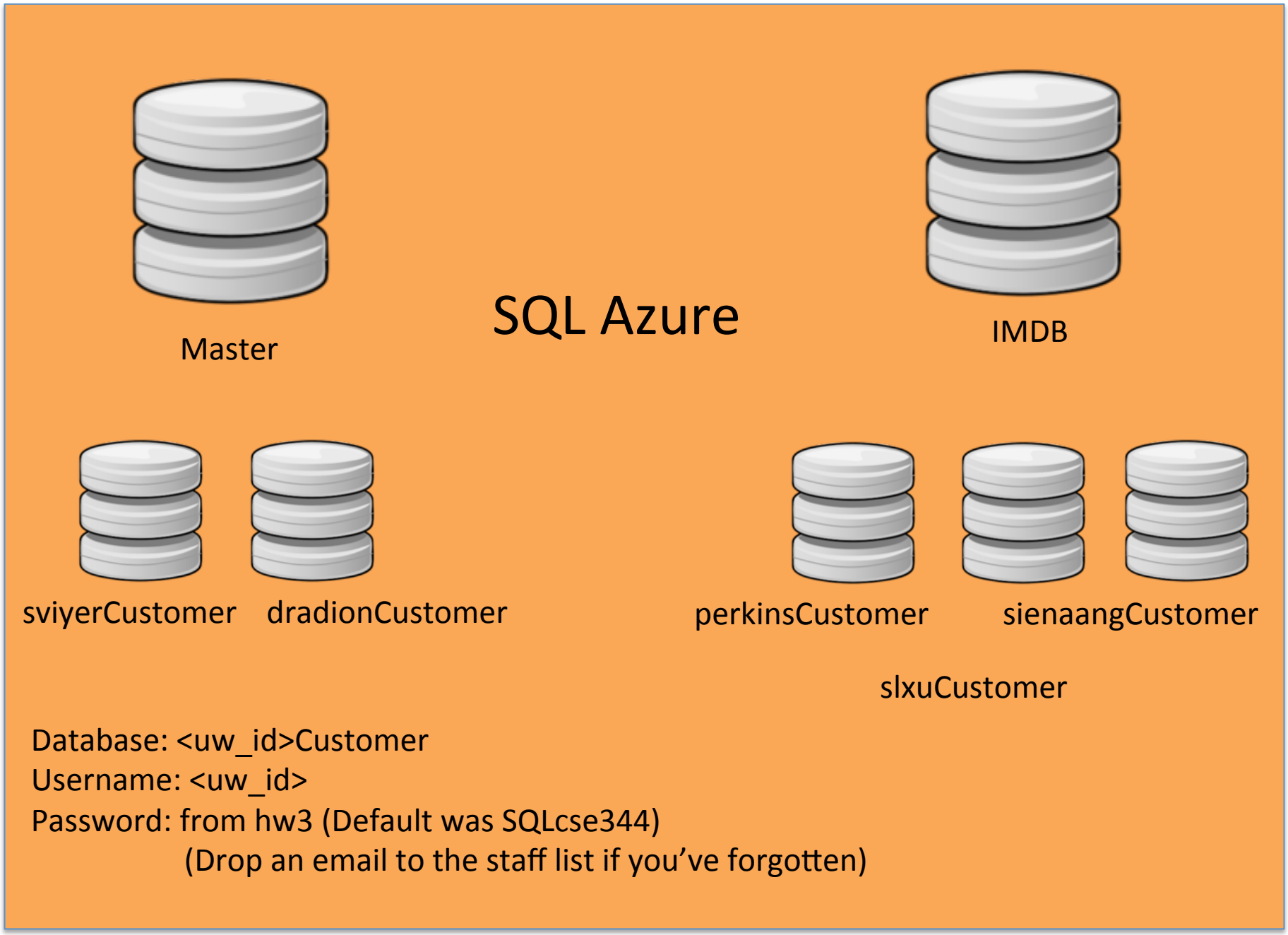
# Homework introduction

## The starter code contains:

- **VideoStore.java**: the command-line interface to your video store; calls into Query.java to run customer transactions
- **Query.java**: code to run customer transactions against your database, such as renting and returning movies
- **dbconn.properties:** a file containing settings to connect to the customer and IMDB databases. You need to edit it before running the starter code.
- **sqljdbc4.jar**: the JDBC to SQL Server driver. This tells Java how to connect to a SQL Azure database server, and needs to be in your CLASSPATH (see below)
- **sqljdbc.jar**: the JDBC to SQL Server driver for older versions of Java. Use this driver only if the other one does not work for you.

# Technical Requirements

- 1. Read access to the IMDB database
- 2. Your own database of user accounts, passwords and rented out videos.
- 3. Query these databases from a Java Program. (starter code)


SQL Azure
Master
IMDB
sviyerCustomer
dradionCustomer
perkinsCustomer
sienaangCustomer
slxuCustomer
Database: <uw_id>Customer
Username: <uw_id>
Password: from hw3 (Default was SQLcse344)
(Drop an email to the staff list if you've forgotten)

# Demo: Create tables (setup.sql)

Demo: Run starter code

1. dbconn.properties
2. Execute program

# The Hello Database program!

- see helloDatabase.java

# Transaction Review

## ACID!

- A Atomicity: all-or-nothing
- C Consistency: integrity of the database must be maintained after a transaction
- I Isolation: each transaction must appear to have executed as if no other transaction is executing at the same time
- D Durability: effect of a transaction must never be lost once it has been completed

# Transaction Review

## Transactions

begin transaction;

... Statement1…

... Statement2…

... Statement3…

commit; /rollback;

# Transaction Review

## Isolation Levels in SQL

- SET TRANSACTION ISOLATION LEVEL
  READ UNCOMMITTED
- SET TRANSACTION ISOLATION LEVEL
  READ COMMITTED
- SET TRANSACTION ISOLATION LEVEL
  REPEATABLE READ
- SET TRANSACTION ISOLATION LEVEL
  SERIALIZABLE

See Query.java

# Transaction fails?

- SQLException is thrown
- try {

```
} catch (SQLException e) {
    // Rollback transaction
}
```

# Common Questions and Mistakes

- Plan is a keyword in SQL, so we cannot create a table name Plan, change it to any name that is relevant.
- If we have a primary key in the table, the clustered indices are created automatically.
- Minimum 8 tuples means 8 tuples in total not 8 tuples for each table.
- Make sure the statements in setup.sql are in the right order.
- The date field in rental database should be a valid SQL date type that includes time.

# Common Questions and Mistakes

- Fastsearch is faster but it is for single word only.
- Print customerinfo = at least the customer name and the number of movies the user can still rent.

# Extra comments on fast search

Magda: “In fast search, you should really execute three queries only (forget about movie availability): the first query should compute the movie metadata for all movies that match the keyword search, the second query should find the directors for all movies that match the keyword search, and the third one should similarly find the actors for all the movies that match the keyword search. Execute each of these three queries separately. You then need to merge the results of the three queries *in* the Java code. The merge will be easier if you sort the results of the three queries.

There is also a way to actually merge all of this info in a single SQL query but don't worry about that because it's similarly easy to write a very expensive single SQL query. Best to try writing three queries exactly."

# Extra comments on fast search

Vaspol: "If you take a close look at the "search" method, you will see that the "search" method is iterating over all the mids returned when you search for the movie. Then, for each of the mids, it will issue 2 queries to the database: getting the actors for that mid, getting the director(s) for that mid. This results in number_of_mids * 2 queries when we do the "search" method, which is expensive.

The idea of "fastsearch" is that we want to reduce the queries being issued to the database. This will give you only a few queries (way less than that of "search“ method.) Therefore, it will be faster in the sense that we don't need to connect to the database and run a lot of queries."

# SQL Injection

- String name_of_movie = “Pulp Fiction”;

- String sql =
- "SELECT * FROM movie WHERE name LIKE '%” + name_of_movie + “%’”;
- Sql = "SELECT * FROM movie WHERE name LIKE ’%Pulp Fiction %’”

- when name_of_movie = “ ’; drop movie;  “

- Sql = "SELECT * FROM movie WHERE name LIKE '%’; drop movie; %’”;

Start early and have fun!

Questions?