CS 211 : Tues 02/13 (lecture 12)

Prof. Hummel
(he/him)

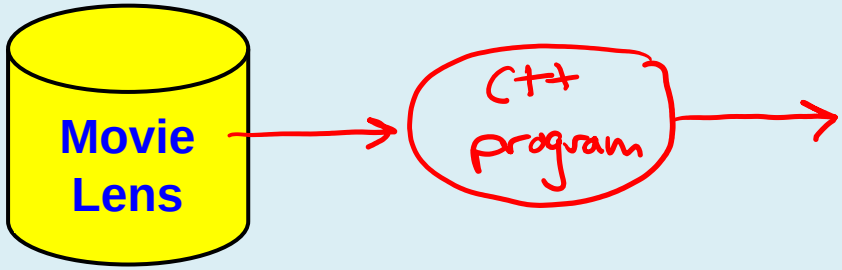
- **<u>Topics</u>: in-class C++ and SQL programming with classes, objects and MovieLens database**

Notes:

- *Lecture slides available on Canvas*
- *We are going to program in class today, and will **collect at the end of class** via Gradescope*
- ***HW 05** (intro to C++) due tonight (Tuesday)*
- ***Project 05** due Friday night (can submit as late as Sunday with late days)*
  - *Note that we are back on the EECS computers, replit not available*

Northwestern University

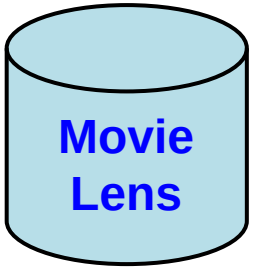
C++ Programming Demo: MovieLens

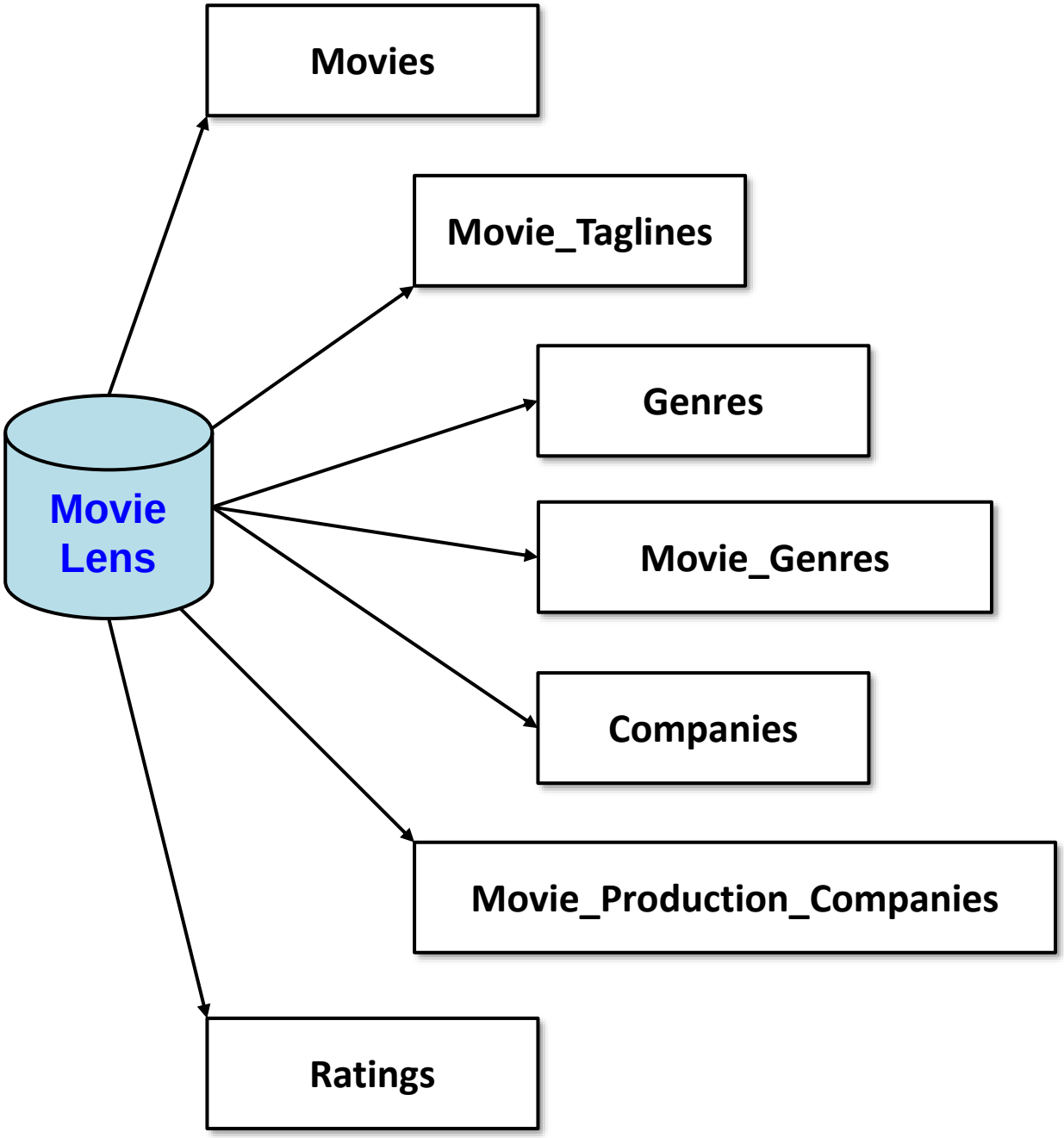


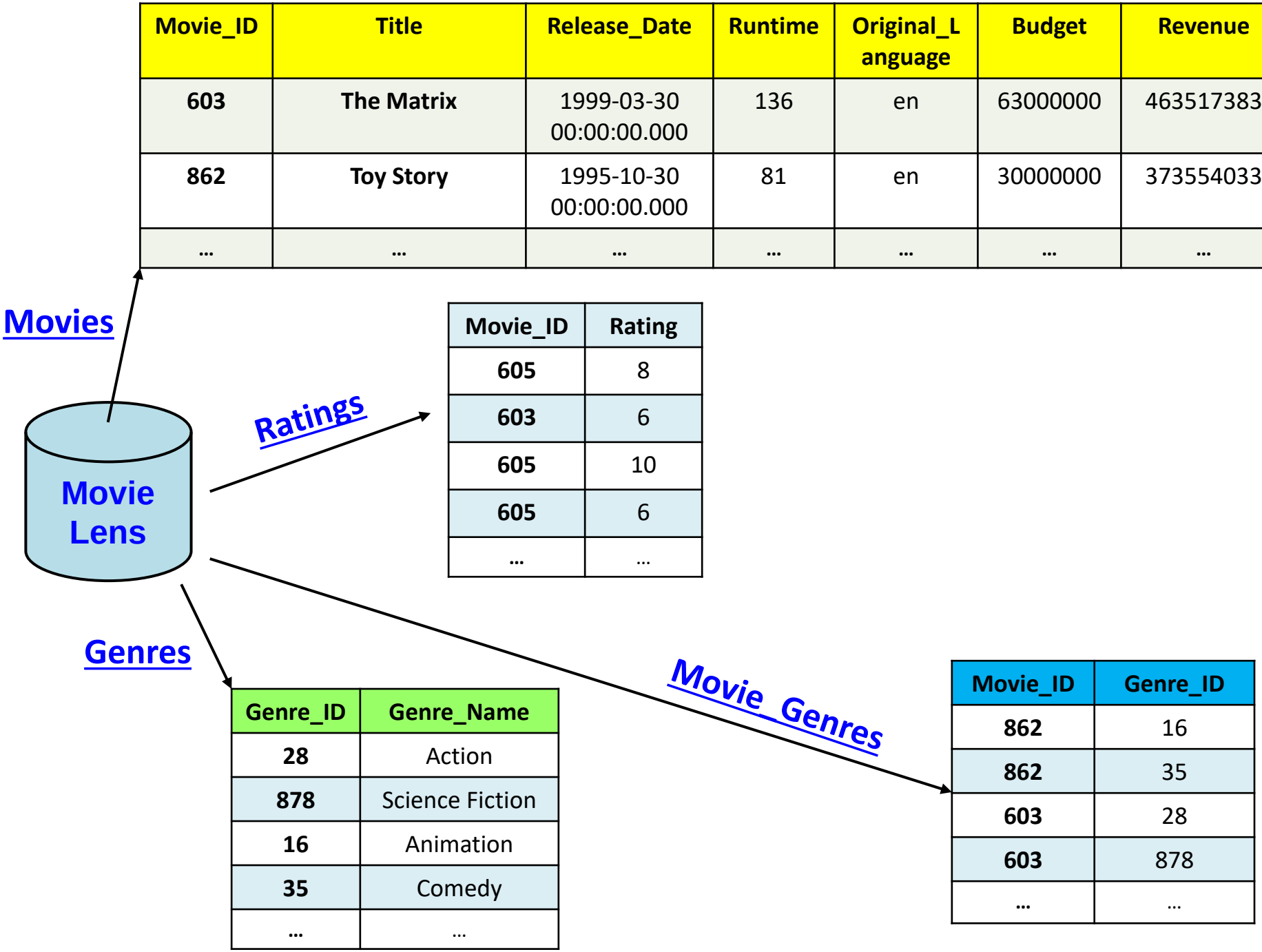

```
** MovieLens **

# of movies: 836

949: Heat ($187436818), 7.81
Genres:
  Drama
  Action
  Thriller
  Crime

710: GoldenEye ($352194034), 5.48
Genres:
  Adventure
  Action
  Thriller

1408: Cutthroat Island ($10017322), 7.41
Genres:
  Adventure
  Action

524: Casino ($116112375), 7.04
Genres:
  Drama
  Crime

5: Four Rooms ($4300000), 6.14
Genres:
  Comedy
  Crime
```

MovieLens database

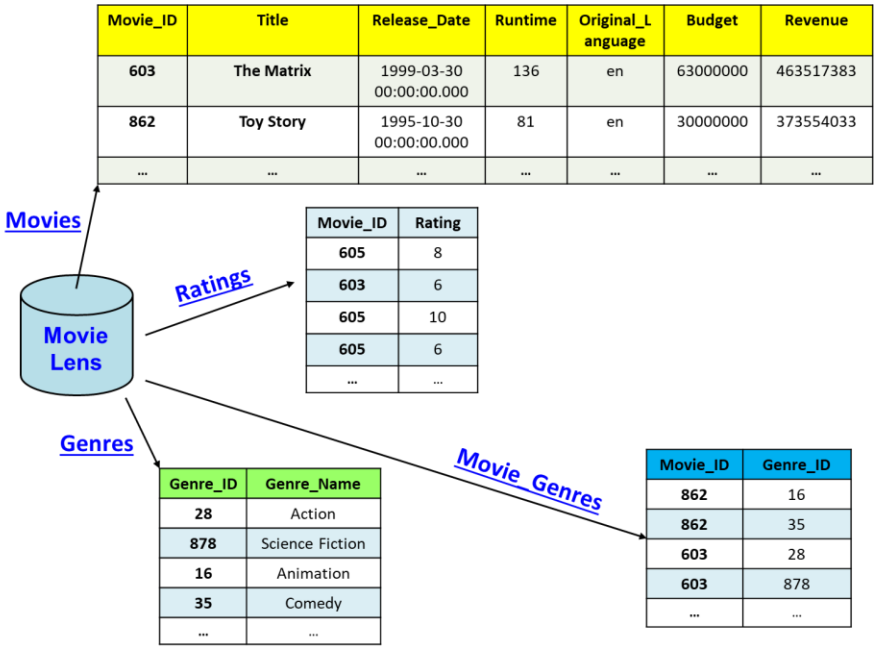
- **MovieLens**
  - *https://movielens.org/*
  - *45K movies*
  - *26M reviews*




Movies
Movie_Taglines
Genres
Movie Lens
Movie_Genres
Companies
Movie_Production_Companies
Ratings


Movie_ID
Title
Release_Date
Runtime
Original_Language
Budget
Revenue
603
The Matrix
1999-03-30 00:00:00.000
136
en
63000000
463517383
862
Toy Story
1995-10-30 00:00:00.000
81
en
30000000
373554033
...
Movies
Movie Lens
Ratings
Movie_ID
Rating
605
8
603
6
605
10
605
6
...
Genres
Genre_ID
Genre_Name
28
Action
878
Science Fiction
16
Animation
35
Comedy
Movie_Genres
Movie_ID
Genre_ID
862
16
862
35
603
28
603
878


SQL



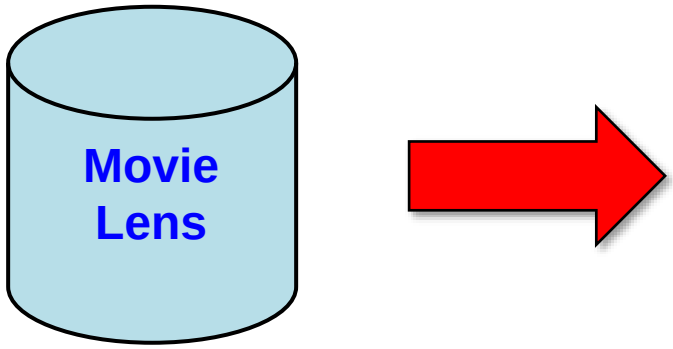

```
SELECT Title FROM Movies ORDER BY Title;
SELECT Count(Title) FROM Movies;
```

```
SELECT * FROM Movies WHERE Title = 'The Matrix';
```

```
SELECT Title, Avg(Rating)
FROM   Movies
JOIN   Ratings
ON     Movies.Movie_ID = Ratings.Movie_ID
WHERE  Title = 'The Matrix';
```

ORM

- ORM = Object-Relational mapping
- Most database software works with objects…




```
class Movie
{
public:
  string Title;
  int    ID;
  double Revenue;

  Movie(string title,
        int id,
        double revenue);

  double getAvgRating();
  vector<string> getGenres();
};
```

Replit

- Login to replit.com
- Open team…
- Open project "Lecture 12"

EECS Computers

```
mkdir movies
cd movies
cp -r /home/cs211/w2024/lecture12/* .
make
./a.out
```

(1) Define a Movie class

- **Open "movie.h"**
- **Data members:** ID (int), Title (string), Revenue (double)
- **Declare a constructor to initialize the data members**
- **Declare a destructor? No…**
  1. *Why not?*
  2. *Constructor does not allocate memory, so nothing to free*

- **Open "movie.cpp"**
- **Implement constructor**
  - *If you need help, solution code in Lectures folder (see link on Canvas under today's lecture)*

(2) "main.cpp" getMovies( ) function

- **Open "main.cpp"**
- **Define getMovies( ) function with db_name parameter, returning a vector<Movie>**
- **Implement the function:**
  1. *Open database*
  2. *Define SQL query*
  3. *Execute query*
  4. *For each row, create a Movie object and push into vector*
  5. *Return vector*

  – *If you need help, solution code in Lectures folder (see link on Canvas under today's lecture)*

(3) write main() to call and test

- **Switch to function main()**
- **Call getMovies( ) function, pass "MovieLens.db"**
- **Assign the returned vector, output the size**
- **Run and test --- vector size should be 836**

  – *If you need help, solution code in Lectures folder (see link on Canvas under today's lecture)*

(4) Output first 10 movies

- **As a test, output the first 10 movies in the vector**

  1. *Use an index-based for loop*
  2. *Access a movie using* ***movies[i]***
  3. *Output ID, Title, and Revenue*

```
** MovieLens **

# of movies: 836
949: Heat ($187436818)
710: GoldenEye ($352194034)
1408: Cutthroat Island ($10017322)
524: Casino ($116112375)
5: Four Rooms ($4300000)
902: The City of Lost Children ($1738611)
63: Twelve Monkeys ($168840000)
2054: Mr. Holland's Opus ($106269971)
880: Antonia's Line ($0)
688: The Bridges of Madison County ($182016617)

** done **
```

```
#include <iomanip>


cout << setprecision(12)
     << ...
```

(5) Movie::getAvgRating( )

- **We want to output a movie's average rating based on reviews in the database…**
- **Define getAvgRating( ) in "movie.h" to return a double**
- **Implement the function in "movie.cpp"**
  1. *Open database – hard-code "MovieLens.db"*
  2. *Define SQL query to select reviews for THIS movie*
  3. *Execute query, binding movie's ID with the query*
  4. *Loop through the results and sum the ratings (1..10)*
  5. *Compute average --- beware of divide by 0*
  6. *Return average*
  7. *Back in "main.cpp", modify main( ) to output average rating as well*
  8. *Run and test*

(6) refactor…

- **Hard-coding database name is a bad idea…**
- **Instead, let's refactor as follows:**
  1. *Add a private DB_name data member to Movie class*
  2. *Modify constructor to accept and store*
  3. *Modify getMovies( ) to pass its db_name parameter*
  4. *Use private data member in getAvgRating( )*
  5. *Run and test…*

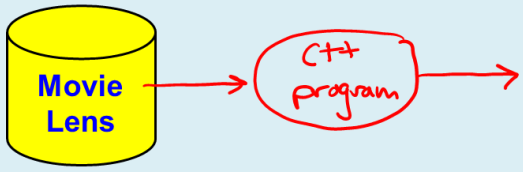
(7) Movie::getGenres( )

```
** MovieLens **

# of movies: 836

949: Heat ($187436818), 7.81
Genres:
  Drama
  Action
  Thriller
  Crime

710: GoldenEye ($352194034), 5.48
Genres:
  Adventure
  Action
  Thriller

1408: Cutthroat Island ($10017322), 7.41
Genres:
  Adventure
  Action

524: Casino ($116112375), 7.04
Genres:
  Drama
  Crime

5: Four Rooms ($4300000), 6.14
Genres:
  Comedy
  Crime
```

- **We want to output a movie's genres…**
- **Define getGenres( ) in "movie.h" to return vector<string>**
- **Implement the function in "movie.cpp"**
  1. *Open database using private DB_*
  2. *Define SQL query to select Genre_Name using a join…*
  3. *Execute query*
  4. *Loop through the results and push each Genre_Name into vector*
  5. *Return vector*
  6. *Back in "main.cpp", after each movie call to get genres and output*
  7. *Run and test*

Submit your work to Gradescope

- **EECS computers?**
  - *make submit*
- **Replit?**
  - *download main.cpp, movie.cpp, movie.h*
  - *upload to Gradescope*




```
** MovieLens **

# of movies: 836

949: Heat ($187436818), 7.81
Genres:
  Drama
  Action
  Thriller
  Crime

710: GoldenEye ($352194034), 5.48
Genres:
  Adventure
  Action
  Thriller

1408: Cutthroat Island ($10017322), 7.41
Genres:
  Adventure
  Action

524: Casino ($116112375), 7.04
Genres:
  Drama
  Crime

5: Four Rooms ($4300000), 6.14
Genres:
  Comedy
  Crime
```

What's due?

HW 05 *due tonight*

Project 05 *due Friday night… Note that we are back on the EECS computers, replit not available*