SQL Programming 2

Sections to read:

-”The Database language SQL” chapter
- - SQLite tutorial https://www.sqlitetutorial.net/

Revision of SQL1

The (small) Movies database:

- Movie(<u>mvID</u>, Title, Rating, Rel_date, Length, Studio)
- Classification(<u>mvID*, Genre</u>)
- Cast(<u>mvID*, Actor</u>)
- Direct(<u>mvID*, Director</u>)

All our SQL queries will be based on the (small) Movies database

Revision of SQL1 ...

- Queries on one relation
  - List the title and length of movies with title starting with ‘T’. Length should be in hours with the header “Length in Hours”.
- Queries on multiple relations - Join queries.
  - List the title of movies having “Tom Hanks” and produced by “Roadshow”.

Revision of SQL1...

- You need to research on
  - Datetime function: DATETIME()
  - String function: UPPER() and LOWER()
  - Number function: ROUND()
- SQLite online tutorial - functions

https://www.sqlitetutorial.net/sqlite-functions/

Multi-relation Queries

- Interesting queries often combine data from more than one table.
- There are several ways to compose such queries in SQL.
  - Join: List all tables in the FROM clause or use JOIN operators.
  - Subquery: A subquery is nested inside the WHERE (or FROM) clause of a query.

The JOIN query using WHERE

- The TABLE.COLUMN notation to distinguish columns from different tables.
- A wrong query - “columns ambiguously defined”:

```
SELECT mvid, title, director
FROM Movie, Direct
WHERE mvid = mvid
```

Explicit Tuple-Variables

- Sometimes a query needs to use two copies of the same relation.
- Distinguish copies by following the relation name by the name of a tuple-variable, in the FROM clause.

A Previous Example

- Which movies have both “Marie Gillain” and “Audrey Tautou”?

```
SELECT mvID
FROM cast
WHERE actor=‘Marie Gillain’
      AND actor =‘Audrey Tautou’;
```

Solution: Self-join

Cast C1

```
MVID ACTOR
---- --------------------
   1 Tom Hanks
   2 Alessandro Nivola
   2 Audrey Tautou
   2 Benolt Poelvoorde
   2 Marie Gillain
   3 Daniel Radcliffe
   3 Emma Watson
   3 Rupert Grint
   4 Betty White
   4 Malin Akerman
   4 Mary Steenburgen
   4 Ryan Reynolds
   4 Sandra Bullock
   5 Chris Wedge
   5 Denis Leary
   5 John Leguizamo
   5 Queen Latifah
   5 Ray Ramono
```

Cast C2

```
MVID ACTOR
---- --------------------
   1 Tom Hanks
   2 Alessandro Nivola
   2 Audrey Tautou
   2 Benolt Poelvoorde
   2 Marie Gillain
   3 Daniel Radcliffe
   3 Emma Watson
   3 Rupert Grint
   4 Betty White
   4 Malin Akerman
   4 Mary Steenburgen
   4 Ryan Reynolds
   4 Sandra Bullock
   5 Chris Wedge
   5 Denis Leary
   5 John Leguizamo
   5 Queen Latifah
   5 Ray Ramono
```

```
SELECT C1.mvID
FROM Cast C1, Cast C2
WHERE C1.mvID=C2.mvID
  AND C1.actor='Marie Gillain'
  AND C2.actor='Audrey Tautou';

MVID
--------
  2
```

✓

Natural Join

- Natural join: Tuples from two relations are joined conditioned on that they have the same values for the common attributes.
- Common attributes appear only once in the result of Natural Join.

The NATURAL JOIN Operator

List all movies, including their mvID, title rating, studio and director information.

```
SELECT  mvID, title, rating, studio, director
FROM Movie natural join Direct
```

```
 MVID TITLE                                      RA STUDIO               DIRECTOR
---------- ---------------------------------------- -- -------------------- ----------------------------------------
         1 Angels and Demons                        M  Sony Pictures        Ron Howard
         2 Coco Avant Chanel                        PG Roadshow             Anne Fontaine
         3 Harry Potter and the Half-Blood Prince   M  Roadshow             David Yates
         4 The Proposal                             PG Disney               Anne Fletcher
         5 Ice Age: Dawn of the Dinosaurs           PG 20th Century Fox     Carlos Saldanha
         5 Ice Age: Dawn of the Dinosaurs           PG 20th Century Fox     Mike Thurmeier
```

Natural Join ...

Are the two queries below equivalent?

– Almost “yes” but with subtle difference --- they have the same number of tuples in the output but subtle difference in the number of attributes. What is it?

```
SELECT *
FROM Movie NATURAL JOIN Direct
```

mvID is included once in the output.

```
SELECT *
FROM Movie, Direct
WHERE Movie.mvID = Direct.mvID
```

Both Movie.mvID and Direct.mvID are included in the output.

Theta(Θ) Join

- Theta join: Join two relations on any condition:

  <*relation 1*> JOIN <*relation 2*> ON *Condition.*

The JOIN …ON… Operator

- Find the movies that have at least two directors.

```
select D1.mvID
from Direct D1 JOIN Direct D2 on
      D1.mvID = D2.mvID and D1.Director < D2.Director
```

Why not '<>'?

Direct

```
 MVID DIRECTOR
---------- --------------------
         1 Ron Howard
         2 Anne Fontaine
         3 David Yates
         4 Anne Fletcher
         5 Carlos Saldanha
         5 Mike Thurmeier
```

Output

```
      MVID
----------
         5
```

Join: A Complex Example

- Find the movies that have at least one genre that is the same as that of “Harry Potter and the Half-Blood Prince”.

```
select M2.mvID, M2.title, C2.genre
from movie M1, classification C1, movie M2, classification C2
where M1.mvID = C1.mvID
      and M1.title='Harry Potter and the Half-Blood Prince'
      and M2.mvID=C2.mvID
      and M1.mvID != M2.mvID
      and C1.genre=C2.genre
```

M2-C2 are about the movies to be found and output.

M2 should not be Harry Potter itself.

M1-C1 is about Harry Potter.

Join: A Complex Example

- Natural JOIN --- easier to understand.

```
select MC2.mvID, MC2.title, MC2.genre
from (select * from movie natural join classification) MC1,
       (select * from movie natural join classification) MC2
where
            MC1.mvID != MC2.mvID
     and MC1.genre=MC2.genre
     and MC1.title='Harry Potter and the Half-Blood Prince'
```

MC1

```
MVID TITLE              GENRE
---------- ------------------------------
1 Angels and Demons  Drama
2 Coco Avant Chanel   Drama
3 Harry Potter …          Action
3 Harry Potter …          Adventure
3 Harry Potter …          Drama
4 The Proposal             Comedy
5 Ice Age …                  Animated
5 Ice Age …                  Comedy
```

MC2

```
MVID TITLE              GENRE
---------- ------------------------------
1 Angels and Demons  Drama
2 Coco Avant Chanel   Drama
3 Harry Potter …          Action
3 Harry Potter …          Adventure
3 Harry Potter …          Drama
4 The Proposal             Comedy
5 Ice Age …                  Animated
5 Ice Age …                  Comedy
```

Output

```
MVID TITLE              GENRE
---------- ------------------------------
1 Angels and Demons  Drama
2 Coco Avant Chanel   Drama
```

Exercises

- Explain what the following query is doing.

```
select distinct m1.mvID, m1.title
from movie m1 join movie m2
        on m1.length > m2.length
```

The below literal interpretation is not acceptable:
“ Join the Movie table with itself conditioned on that the length of a movie is greater than that of another movie. Output the movie ID and title of movies”
What movies are in the output? Find movies that ….

Exercises …

- Write an SQL query to find the movies directed by Ron Howard. Output the title of these movies.

Subqueries

- A parenthesized SELECT-FROM-WHERE statement (*subquery*) can be used as a value in the WHERE (or FROM) clause of another query.

```
SELECT ...
FROM ...
WHERE ....    .... op (SELECT ...
                       FROM ...
                       WHERE ...)
```

Returns True or False!

Subqueries that return a relation

- Generally a subquery returns a relation --- a set of tuples.
- To use the result of the subquery in the WHERE clause, we need operators (NOT) IN and (NOT) EXISTS. The result of IN and EXISTS expressions is TRUE or FALSE.

The IN Operator

- <tuple> IN (<subquery>) is true if and only if the tuple is a member of the relation produced by the subquery.
  - Opposite: <tuple> NOT IN (<subquery>).

Example: IN

- Find the movies that are dramas.

```
SELECT mvID, title
FROM Movie
WHERE mvID IN (SELECT mvID
               FROM Classification
               WHERE Genre='Drama')
```

Loops over tuples of Movie!

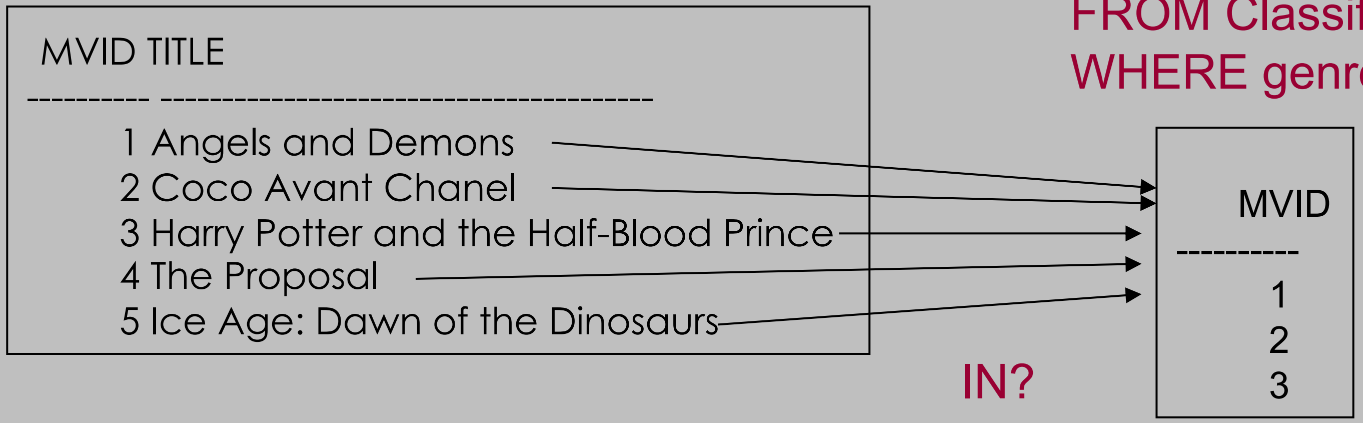
Loop over the tuples of Classification!

•Movie is the outer relation, and Classification is the inner relation.
•Only attributes of outer relation can be output.

Example: IN

Movie

SELECT mvID
FROM Classification
WHERE genre='Drama‘;



```
 MVID TITLE
---------- ----------------------------------------
         1 Angels and Demons
         2 Coco Avant Chanel
         3 Harry Potter and the Half-Blood Prince
         4 The Proposal
         5 Ice Age: Dawn of the Dinosaurs
```

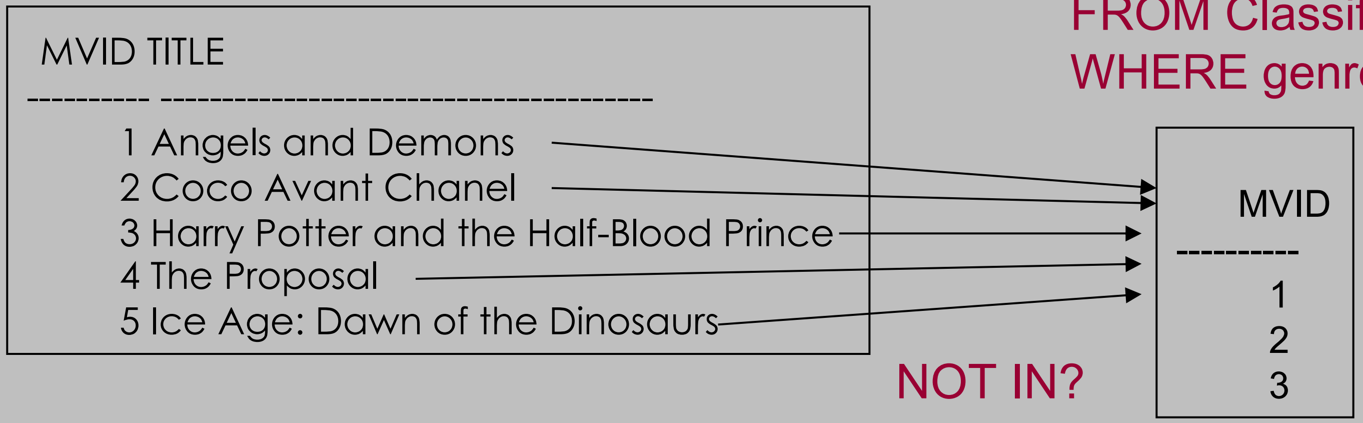
```
      MVID
----------
         1
         2
         3
```

```
SELECT mvID, title
FROM Movie
WHERE mvID IN (SELECT mvID
               FROM Classification
               WHERE Genre=‘Drama’)
```

```
 MVID TITLE
---------- ----------------------------------------
         1 Angels and Demons
         2 Coco Avant Chanel
         3 Harry Potter and the Half-Blood Prince
```

Example: NOT IN

Movie



```
 MVID TITLE
---------- ----------------------------------------
         1 Angels and Demons
         2 Coco Avant Chanel
         3 Harry Potter and the Half-Blood Prince
         4 The Proposal
         5 Ice Age: Dawn of the Dinosaurs
```

```
SELECT mvID
FROM Classification
WHERE genre='Drama‘;
```

```
      MVID
----------
         1
         2
         3
```

NOT IN?

```
SELECT mvID, title
FROM Movie
WHERE mvID NOT IN (SELECT mvID
                   FROM Classification
                   WHERE Genre=‘Drama’)
```

```
 MVID TITLE
---------- ----------------------------------------
         4 The Proposal
         5 Ice Age: Dawn of the Dinosaurs
```

The EXISTS Operator

- EXISTS(<subquery>) is true if and only if the subquery result is not empty.
  - Opposite: NOT EXISTS (<subquery>)

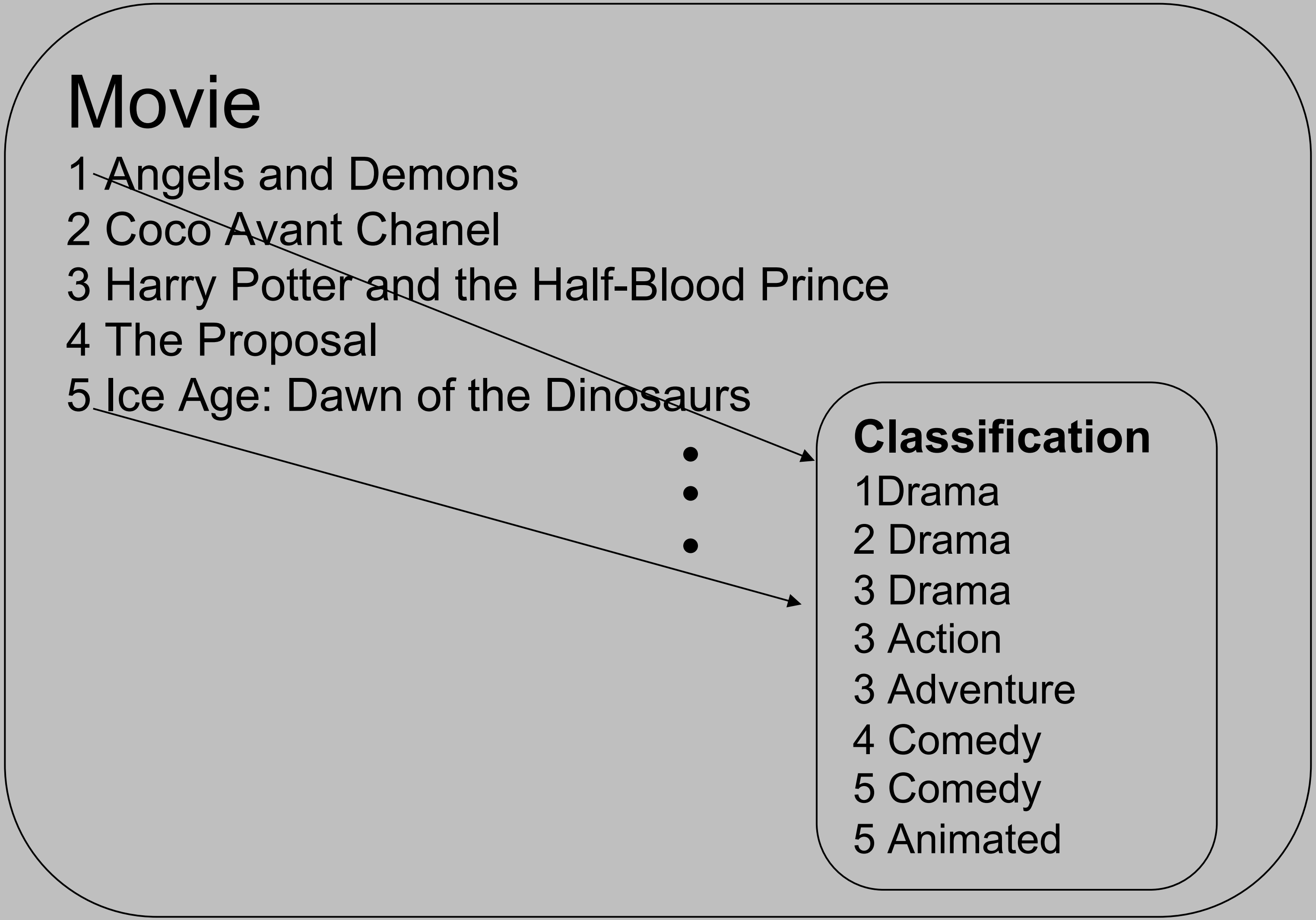
Example: EXISTS

```
SELECT mvID, title
FROM Movie
WHERE
    EXISTS (SELECT *
    FROM Classification
            WHERE mvID = Movie.mvID AND
                    Genre = 'Drama');
```

Variable scope rule:
The inner-most relation by default.

Scope: The outer relation Movie.

Example: EXISTS



Movie

1 Angels and Demons
2 Coco Avant Chanel
3 Harry Potter and the Half-Blood Prince
4 The Proposal
5 Ice Age: Dawn of the Dinosaurs

•
•
•

Classification

1Drama
2 Drama
3 Drama
3 Action
3 Adventure
4 Comedy
5 Comedy
5 Animated

Example: NOT EXISTS

```
SELECT mvID, title
FROM Movie
WHERE
      NOT EXISTS (SELECT *
      FROM Classification
            WHERE mvID = Movie.mvID AND
                        Genre = ‘Drama’);
```

Subqueries that return a value

- When a subquery returns a value, the expression

  `<value> = (<subquery>)`

  can be used in the WHERE clause of a query.
  – Other relational operators can also be used, including <>, !=, >, >=, <, <=.
- Errors will occur if the relational operators are used for subqueries that do not return a value.

Subqueries that return a value: Example

The following query is correct only if it is known that Tom Hanks only performs in one movie.

```
select mvID, title
from Movie
where mvID = (select mvID
              from Cast
              where Actor='Tom Hanks');
```