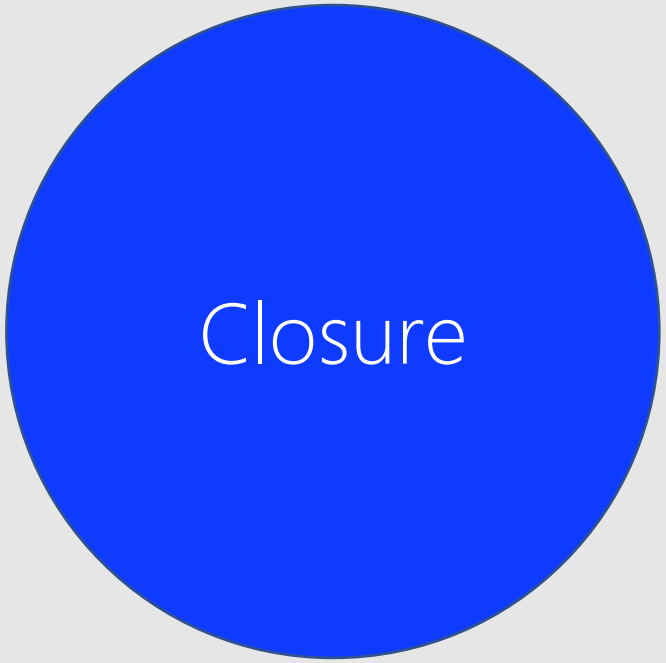
Relational Algebra

The result of relational operations on relations is also a relation.



Advanced SQL × Subquery

The result of SELECT on tables is also a table (temporary though) and can be used inside another query.

Subquery | Nested Query | Inner Query

UNION
INTERSECT
EXCEPT

INSERT
UPDATE

WHERE
FROM
SELECT

Advanced SQL × Subquery

The result of SELECT on tables is also a table (temporary though) and can be used inside another query.

Subquery | Nested Query | Inner Query

UNION
INTERSECT
EXCEPT

SQL × DML × SELECT

1 (SELECT ...)
3 UNION
2 (SELECT ...)

1 (SELECT ...)
3 INTERSECT
2 (SELECT ...)

1 (SELECT ...)
3 EXCEPT
2 (SELECT ...)

Advanced SQL × Subquery

The result of SELECT on tables is also a table (temporary though) and can be used inside another query.

Subquery | Nested Query | Inner Query

Subquery × Bulk × INSERT

Subquery can be used to do bulk INSERT.

INSERT INTO *TableName*(*c*, *c*', *c*'', ...) (SELECT ...);

Advanced SQL × Subquery

The result of SELECT on tables is also a table (temporary though) and can be used inside another query.

Subquery | Nested Query | Inner Query

Subquery × WHERE

Subquery can be used in WHERE clause of DML statements, i.e., SELECT, UPDATE, INSERT INTO, & DELETE FROM.

Here, we use WHERE clause in SELECT statement as it is the most common DML.

Subquery × WHERE

4 SELECT Columns
1 FROM Tables
3 WHERE (c_1, c_2, …, c_n) OP (SELECT c'_1, c'_2, …, c'_n FROM …)
2

- OP ∈ {=, >, >=, <, <=, <>}
- Subquery <u>cannot</u> have more than one row, BUT can have multiple columns
- Subquery always on right side of OP.

LIKE
NOT LIKE

Subquery × WHERE × ~~AGG~~

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

What is the oldest movie?

```
SELECT *
FROM Movie
WHERE ~~ReleaseDate = MIN(ReleaseDate)~~
```

Subquery × WHERE × AGG

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	2001: A Space Odyssey	English	1968	142
2	Rosemary's Baby	English	1968	NULL
3	The Birds	English	1963	119
4	Planet of the Apes	EN	1968	112

What is the oldest movie?

```
SELECT *
FROM Movie
WHERE ReleaseDate = (SELECT MIN(ReleaseDate) FROM Movie)
```

WHERE × IN

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Find the movies in English language?

```
SELECT *
FROM Movie
WHERE Language IN ('English', 'EN')
```

WHERE × IN

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	2001: A Space Odyssey	English	1968	142
2	Rosemary's Baby	English	1968	NULL
3	The Birds	English	1963	119
4	Planet of the Apes	EN	1968	112

Find the movies in non-English language?

```
SELECT *
FROM Movie
WHERE Language NOT IN ('English', 'EN')
```

WHERE × IN vs. FROM × INNER JOIN

What movies are directors' best movies?

```
SELECT *
FROM Movie
WHERE Id IN (SELECT BestMovieId FROM Director)
```

Which one?

```
SELECT M.*, D.*
FROM Movie AS M
INNER JOIN Director AS D ON M.Id = D.BestMovieId
```

WHERE × IN vs. FROM × INNER JOIN

What movies are directors' best movies?

```
SELECT *
FROM Movie
WHERE Id IN (SELECT BestMovieId FROM Director)
```

Which one? We can access Director info in below query.

```
SELECT M.*, D.*
FROM Movie AS M
INNER JOIN Director AS D ON M.Id = D.BestMovieId
```

WHERE × IN vs. FROM × INNER JOIN

What movies are directors' best movies?

```
SELECT *
FROM Movie
WHERE Id IN (SELECT BestMovieId FROM Director)
```

Which one? I don't need Director info! Which one is faster?

```
SELECT M.*
FROM Movie AS M
INNER JOIN Director AS D ON M.Id = D.BestMovieId
```

WHERE × IN vs. INTERSECT

FirstName	LastName
Clint	*Eastwood*

(SELECT FirstName, LastName FROM Director)
INTERSECT
(SELECT FirstName, LastName FROM Actor)

SQL-99
-2016

like

SELECT *
FROM Director
WHERE (FirstName, LastName) IN (SELECT FirstName, LastName FROM Actor)

MySQL does not support INTERSECT.

WHERE × IN vs. EXCEPT

FirstName	LastName
Stanley	*Kubrick*
Alfred	*Hitchcock*

LIKE
NOT LIKE

```
(SELECT FirstName, LastName FROM Director)
EXCEPT
(SELECT FirstName, LastName FROM Actor)
```

```
SELECT *
FROM Director
WHERE (FirstName, LastName) NOT IN (SELECT FirstName, LastName FROM Actor)
```

MySQL does not support EXCEPT.

Subquery × WHERE × ANY | SOME

4 SELECT Columns
1 FROM Tables
3 WHERE c_1 OP ANY (SELECT c'_1 FROM ...)
2

at least

The OP will be true if it is satisfied by one | more values in the subquery.

- OP ∈ {=, >, >=, <, <=, <>}
- Subquery can have multiple rows, BUT only one column.
- Subquery always on right side of ANY.
- ANY is same as SOME.

Subquery × WHERE × ANY | SOME

SELECT Columns
FROM Tables
WHERE c_1 = ANY (SELECT c'_1 FROM ...)

v_1 v_1

---------- same as ----------

SELECT Columns
FROM Tables
WHERE c_1 IN (SELECT c'_1 FROM ...)

v_1 v_1

Subquery × WHERE × ANY | SOME

Director						
Id	FirstName	LastName	DateOfBirth	PlaceOfBirth	BestMovieId	MovieCount
1	Stanley	Kubrick	Jul. 26, 1928	USA	1	13
2	Alfred	Hitchcock	Aug. 13, 1899	England	NULL	47
3	Clint	Eastwood	May 31, 1930	USA	NULL	35

What question this SQL statement is trying to answer?

```
(SELECT * FROM Director)
EXCEPT
(SELECT * FROM Director
 WHERE MovieCount < ANY (SELECT MovieCount FROM Director))
```

all directors

13 ✗ ✗ ✗
47 ✓ ✗ ✓
35 ✗

Subquery × WHERE × ANY | SOME

Director						
Id	FirstName	LastName	DateOfBirth	PlaceOfBirth	BestMovieId	MovieCount
1	*Stanley*	*Kubrick*	*Jul. 26, 1928*	*USA*	*1*	*13*
2	*Alfred*	*Hitchcock*	*Aug. 13, 1899*	*England*	NULL	*47*
3	*Clint*	*Eastwood*	*May 31, 1930*	*USA*	NULL	*35*

What question this SQL statement is trying to answer?

```
SELECT * FROM Director
WHERE MovieCount = (SELECT MAX(MovieCount) FROM Director)
```

So, we can re-write MAX by other operations!

AVG

Subquery × WHERE × ALL

4 SELECT Columns
1 FROM Tables
3 WHERE c_1 OP ALL (SELECT c'_1 FROM ...)
2

The OP will be true if it is satisfied by ALL values in the subquery.

- OP ∈ {=, >, >=, <, <=, <>}
- Subquery can have multiple rows, BUT only <u>one column</u>.
- Subquery always on right side of ALL.

Subquery × WHERE × ALL

What question this SQL statement is trying to answer?

```
SELECT * FROM Director
WHERE MovieCount > All (SELECT MovieCount
                        FROM Director
                        WHERE PlaceOfBirth = 'USA')
```

Subquery × WHERE × ALL

What question this SQL statement is trying to answer?

```
SELECT * FROM Director
WHERE MovieCount > All (SELECT MovieCount
                        FROM Director
                        WHERE PlaceOfBirth = 'USA')
```

```
SELECT * FROM Director
WHERE MovieCount > (SELECT MAX(MovieCount)
                    FROM Director
                    WHERE PlaceOfBirth = 'USA')
```

Subquery × WHERE × EXISTS

4 SELECT Columns
1 FROM Tables
3 WHERE [NOT] EXISTS (SELECT 1 | * | … FROM …)
2

The condition is true if |subquery|>0, i.e., at least one row in subquery.

- Subquery can have multiple rows and columns.
- Subquery always on right side of EXISTS.

The main use case is with *correlated* subquery, ending slides ☺

Advanced SQL × Subquery

The result of SELECT on tables is also a table (temporary though) and can be used inside another query.

Subquery | Nested Query | Inner Query

Subquery × FROM

3 SELECT Columns
1 FROM Tables, (SELECT …) AS S1, (SELECT …) AS S2, …
2 WHERE θ

Each subquery is assumed to be a Table.

- Subquery can have multiple rows and columns.
- Subquery always with alias, i.e., AS.

Subquery × FROM

5 SELECT Columns
1 FROM Table
2 INNER | LEFT | RIGHT OUTER JOIN (SELECT ...) AS S ON θ'
3 WHERE θ

Each subquery is assumed to be a Table.

- Subquery can have multiple rows and columns.
- Subquery always with alias, i.e., AS.

Subquery × FROM

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Movies longer than the average time of movies in the same release year?

Subquery × FROM

Movies longer than the average time of movies in the same release year?

ReleaseDate	AvgTime
1968	*127*
1963	*119*

Subquery × FROM

Movies longer than the average time of movies in the same release year?

```
SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
FROM Movie
GROUP BY ReleaseDate
```

ReleaseDate	AvgTime
1968	*127*
1963	*119*

Subquery × FROM

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

ReleaseDate	AvgTime
1968	*127*
1963	*119*

Movies longer than the average time of movies in the same release year?

```
SELECT *
FROM Movie
```

```
SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
FROM Movie
GROUP BY ReleaseDate
```

Subquery × FROM

Movie					S	
Id	Title	Language	ReleaseDate	RunningTime	ReleaseDate	AvgTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*	*1968*	*127*
2	*Rosemary's Baby*	*English*	*1968*	NULL	*1968*	*127*
3	*The Birds*	*English*	*1963*	*119*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*	*1968*	*127*

Movies longer than the average time of movies in the same release year?

```
SELECT *
FROM Movie AS M, (SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
                  FROM Movie
                  GROUP BY ReleaseDate) AS S
WHERE M.ReleaseDate = S.ReleaseDate
```

Subquery × FROM

M					S	
Id	Title	Language	ReleaseDate	RunningTime	ReleaseDate	AvgTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*	*1968*	*127*
2	*Rosemary's Baby*	*English*	*1968*	NULL	*1968*	*127*
3	*The Birds*	*English*	*1963*	*119*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*	*1968*	*127*

Movies longer than the average time of movies in the same release year?

```
SELECT *
FROM Movie AS M
INNER JOIN (SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
            FROM Movie
            GROUP BY ReleaseDate) AS S ON M.ReleaseDate = S. ReleaseDate
```

Subquery × FROM

M					S	
Id	Title	Language	ReleaseDate	RunningTime	ReleaseDate	AvgTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*	*1968*	*127*
2	*Rosemary's Baby*	*English*	*1968*	NULL	*1968*	*127*
3	*The Birds*	*English*	*1963*	*119*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*	*1968*	*127*

Movies longer than the average time of movies in the same release year?

```
SELECT *
FROM Movie AS M, (SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
                  FROM Movie
                  GROUP BY ReleaseDate) AS S
WHERE M.ReleaseDate = S.ReleaseDate
```

Subquery × FROM

Movie						
Id	Title	Language	ReleaseDate	RunningTime	ReleaseDate	AvgTime
1	2001: A Space Odyssey	English	1968	142	1968	127
2	Rosemary's Baby	English	1968	NULL	1968	127
3	The Birds	English	1963	119	1963	119
4	Planet of the Apes	EN	1968	112	1968	127

Movies longer than the average time of movies in the same release year?

```
SELECT *
FROM Movie AS M
INNER JOIN (SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
            FROM Movie
            GROUP BY ReleaseDate) AS S ON M.ReleaseDate = S. ReleaseDate
WHERE M.RunningTime > S.AvgTime
```

Correlated Subquery

A correlated subquery (aka synchronized subquery) is a subquery that uses columns of tables from the outer query.

Correlated Subquery

Movies longer than the average time of movies in the same release year?

```
SELECT *
FROM Movie AS M1
WHERE RunningTime > (?)
```

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Correlated Subquery

Movies longer than the average time of movies in the same release year?

```
SELECT *
FROM Movie AS M1
WHERE RunningTime > (?)
```

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Correlated Subquery

Movies longer than the average time of movies in the same release year?

```
SELECT *
FROM Movie AS M1
WHERE RunningTime > (?)
```

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Correlated Subquery

Movies longer than the average time of movies in the same release year?

```
SELECT *
FROM Movie AS M1
WHERE RunningTime > (?)
```

M1. ReleaseDate

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Movie				
Id	Title	Language	ReleaseDate	RunningTime
1	*2001: A Space Odyssey*	*English*	*1968*	*142*
2	*Rosemary's Baby*	*English*	*1968*	NULL
3	*The Birds*	*English*	*1963*	*119*
4	*Planet of the Apes*	*EN*	*1968*	*112*

Correlated Subquery

Movies longer than the average time of movies in the same release year?

```
SELECT *
FROM Movie AS M1
WHERE RunningTime > (SELECT AVG(RunningTime)
                     FROM Movie AS M2
                     WHERE M2.ReleaseDate = M1.ReleaseDate)
```

Correlated Subquery

Movies longer than the average time of movies in the same release year?

```
SELECT *
FROM Movie AS M1
WHERE RunningTime > (SELECT AVG(RunningTime)
                     FROM Movie AS M2
                     WHERE M2.ReleaseDate = M1.ReleaseDate)
```

Which one?

```
SELECT *
FROM Movie AS M
INNER JOIN (SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
            FROM Movie
            GROUP BY ReleaseDate) AS S ON M.ReleaseDate = S. ReleaseDate
WHERE M.RunningTime > S.AvgTime
```

Correlated Subquery

The subquery is evaluated once for each row processed by the outer query, it is inefficient!

```
SELECT *
FROM Movie AS M1
WHERE RunningTime > (SELECT AVG(RunningTime)
                     FROM Movie AS M2
                     WHERE M2.ReleaseDate = M1.ReleaseDate)
```

```
SELECT *
FROM Movie AS M
INNER JOIN (SELECT ReleaseDate, AVG(RunningTime) AS AvgTime
            FROM Movie
            GROUP BY ReleaseDate) AS S ON M.ReleaseDate = S. ReleaseDate
WHERE M.RunningTime > S.AvgTime
```

Correlated Subquery

Update movie count for each director.

Correlated Subquery

Update movie count for each director.

UPDATE Director SET MovieCount = ? WHERE Id = ??

Director						
Id	FirstName	LastName	DateOfBirth	PlaceOfBirth	BestMovieId	MovieCount
1	Stanley	Kubrick	Jul. 26, 1928	USA	1	13 → 5
2	Alfred	Hitchcock	Aug. 13, 1899	England	NULL	47
3	Clint	Eastwood	May 31, 1930	USA	NULL	35

MovieDirector		
Id	MovieId	DirectorId
1	1	1
2	2	1
3	3	2
4	4	2
5	5	1
6	5	2
7	6	1
8	7	1

Correlated Subquery

Update movie count for each director.

UPDATE Director SET MovieCount = ? WHERE Id = ??

Director						
Id	FirstName	LastName	DateOfBirth	PlaceOfBirth	BestMovieId	MovieCount
1	Stanley	Kubrick	Jul. 26, 1928	USA	1	13
2	Alfred	Hitchcock	Aug. 13, 1899	England	NULL	47 → 3
3	Clint	Eastwood	May 31, 1930	USA	NULL	35

MovieDirector		
Id	MovieId	DirectorId
1	1	1
2	2	1
3	3	2
4	4	2
5	5	1
6	5	2
7	6	1
8	7	1

Correlated Subquery

Update movie count for each director.

UPDATE Director SET MovieCount = ? WHERE Id = ??

Director						
Id	FirstName	LastName	DateOfBirth	PlaceOfBirth	BestMovieId	MovieCount
1	Stanley	Kubrick	Jul. 26, 1928	USA	1	13
2	Alfred	Hitchcock	Aug. 13, 1899	England	NULL	47
3	Clint	Eastwood	May 31, 1930	USA	NULL	35 → 0

MovieDirector		
Id	MovieId	DirectorId
1	1	1
2	2	1
3	3	2
4	4	2
5	5	1
6	5	2
7	6	1
8	7	1

Correlated Subquery

Update movie count for each director.

Id	MovieId	DirectorId
1	1	1
2	2	1
3	3	2
4	4	2
5	5	1
6	5	2
7	6	1
8	7	1

Correlated Subquery

Update movie count for each director.

```
SELECT D.Id, COUNT(*)
FROM Director AS D
INNER JOIN MovieDirector AS MD ON D.Id = MD.DirectorId
GROUP BY D.Id
```

Which one?

```
SELECT DirectorId COUNT(*)
FROM MovieDirector
GROUP BY DirectorId
```

Id	MovieId	DirectorId
1	1	1
2	2	1
3	3	2
4	4	2
5	5	1
6	5	2
7	6	1
8	7	1

Correlated Subquery

Update movie count for each director.

```
UPDATE Director AS D SET MovieCount = (
    SELECT COUNT(*)
    FROM MovieDirector AS MD
    GROUP BY MD.DirectorId
    HAVING MD.irectorId = D.Id)
```

where

MD.DirectorId = D.Id

MovieDir

Director						
Id	FirstName	LastName	DateOfBirth	PlaceOfBirth	BestMovieId	MovieCount
1	Stanley	Kubrick	Jul. 26, 1928	USA	1	13 → 5
2	Alfred	Hitchcock	Aug. 13, 1899	England	NULL	47
3	Clint	Eastwood	May 31, 1930	USA	NULL	35

Id	MovieId	DirectorId	
1	1	1	✓
2	2	1	✓
3	3	2	✗
4	4	2	✗
5	5	1	✓
6	5	2	✗
7	6	1	✓
8	7	1	✓

Correlated Subquery × WHERE × EXISTS

What does it do?

E

```
DELTE FROM Director AS D WHERE NOT EXISTS (
    SELECT 1 FROM MovieDirector AS MD ON D.Id = MD.DirectorId)
```

Subquery × Final Notes

- The ORDER BY clause may not be used in a subquery.
- The subquery cannot be used inside BETWEEN for outer query. However, the BETWEEN operator can be used in subquery.
- Subquery can be used in HAVING clause.
- There might be some SQL-92 support issues by different DBMSs.

Today

Data Modeling in RDBMS

Real World Entity

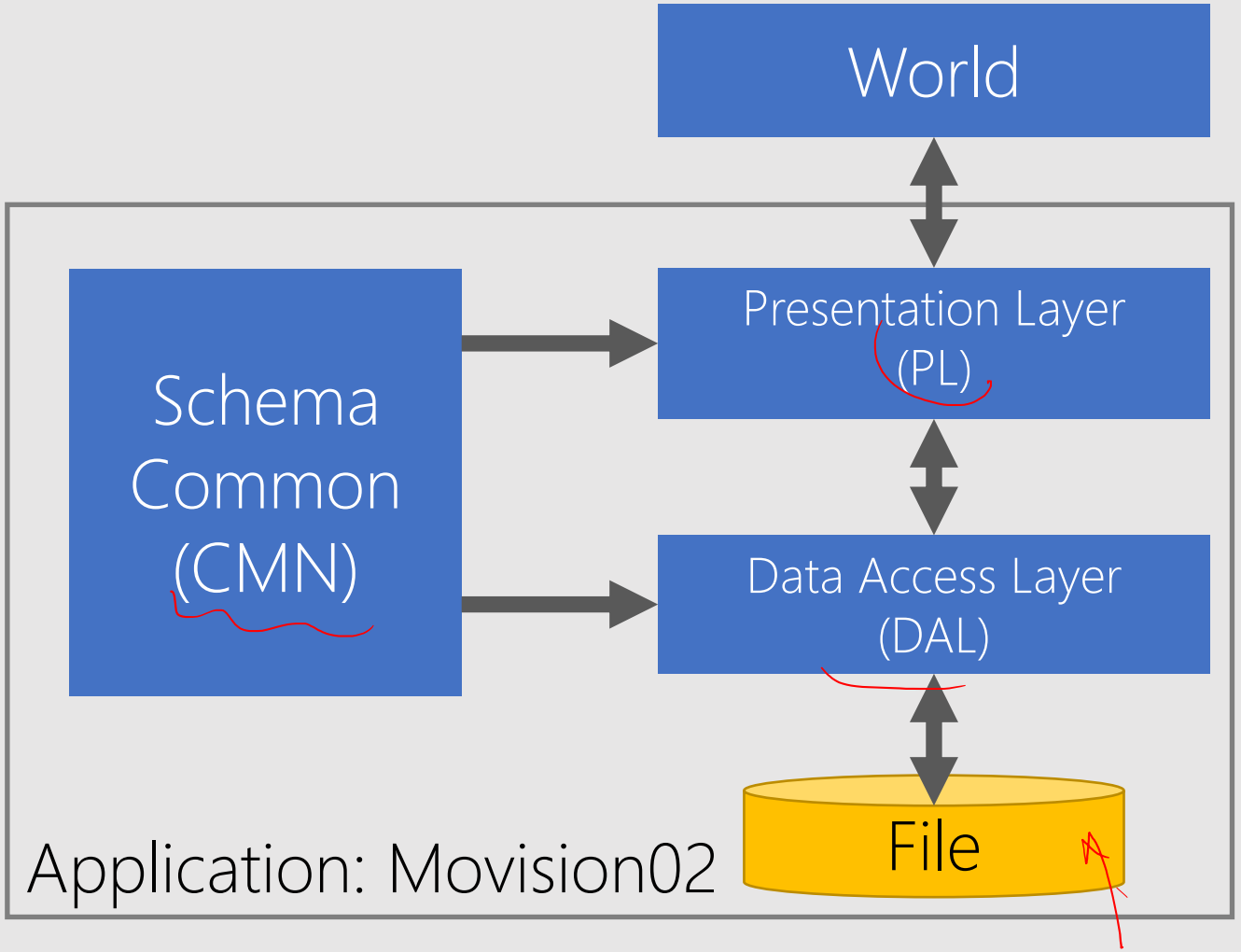
Conceptual Level | Entity-Relationship Model (E/R)

| Logical Level | Relational Model

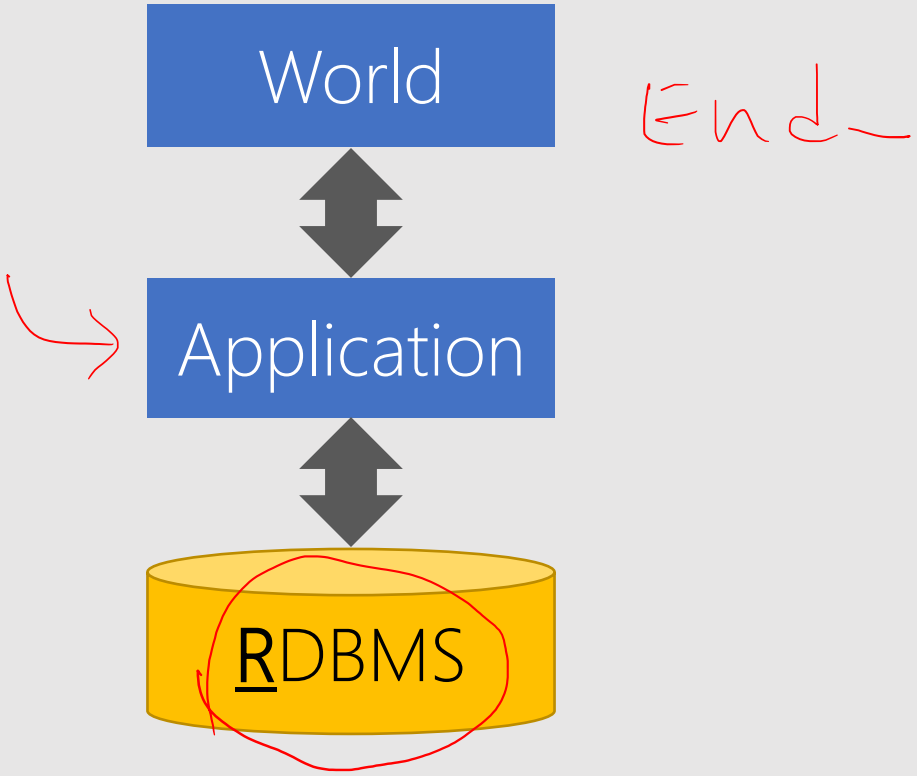
| Physical Level | SQL

Computable Entity

Physical Level × File

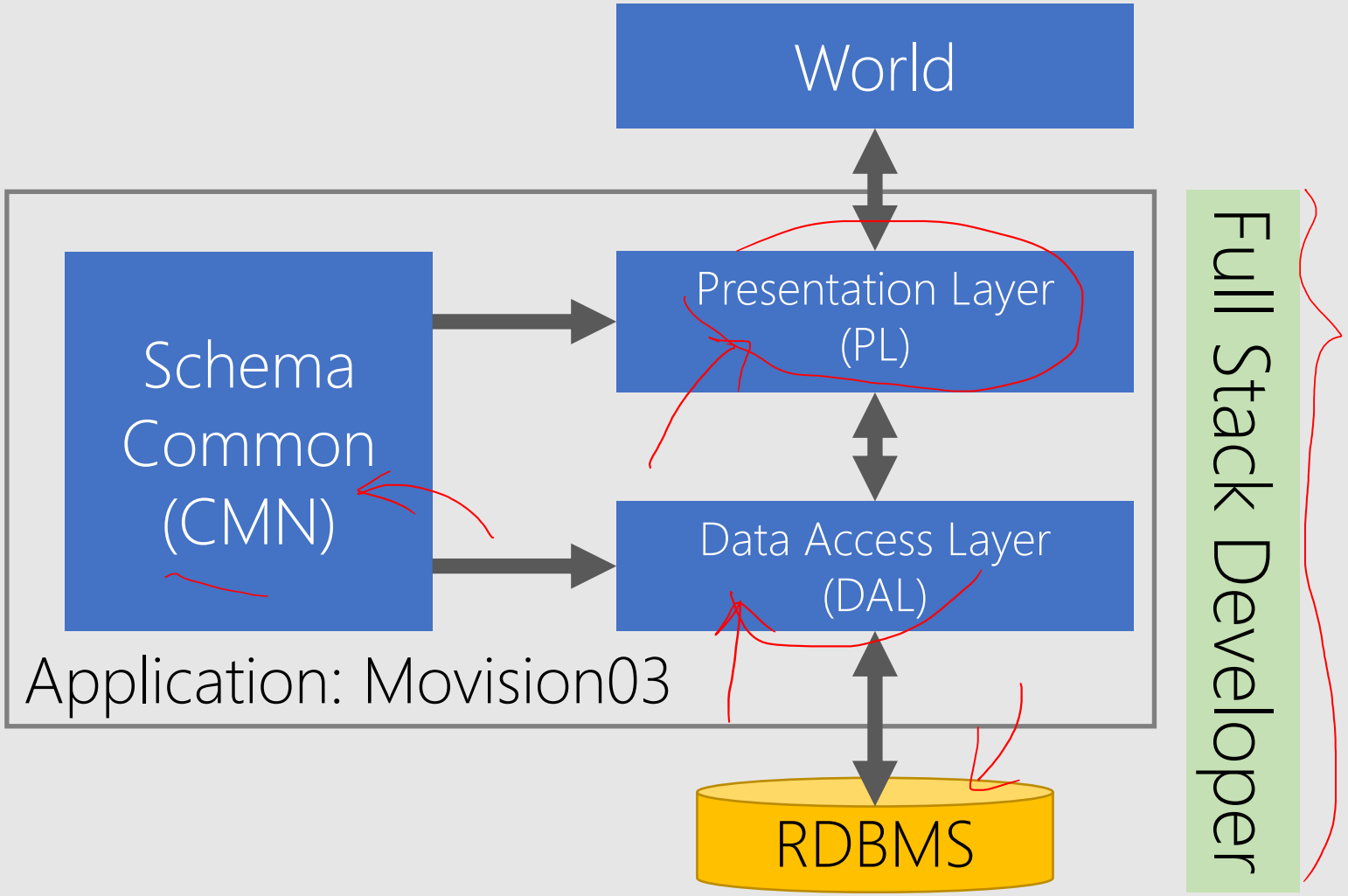


Physical Level × RDBMS

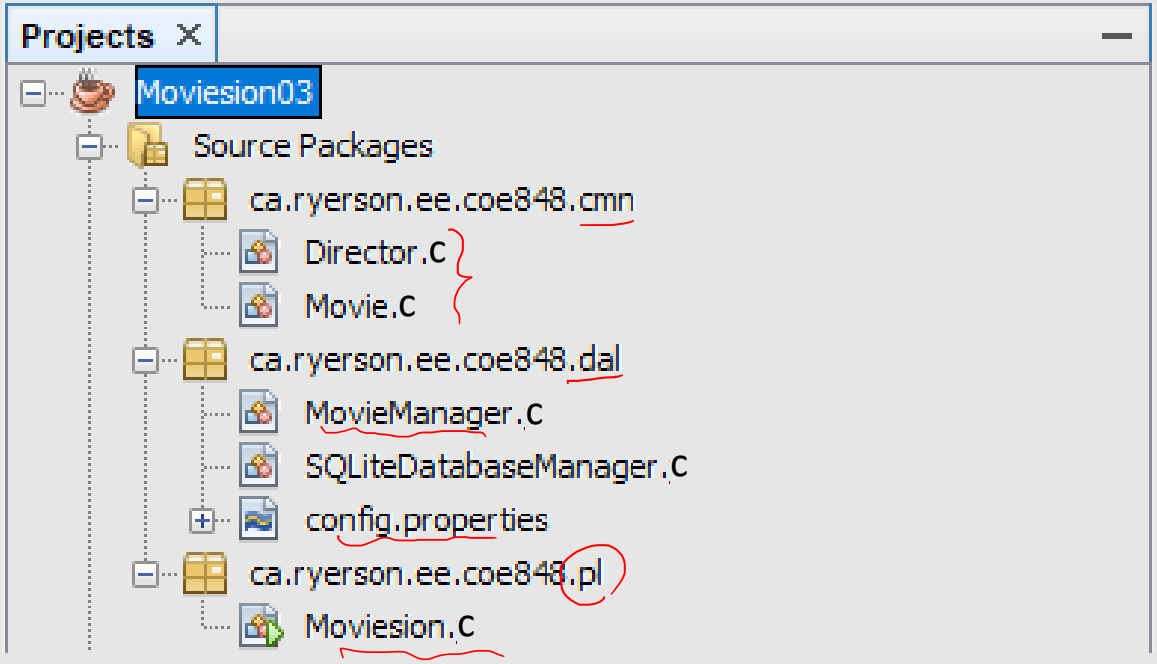


End

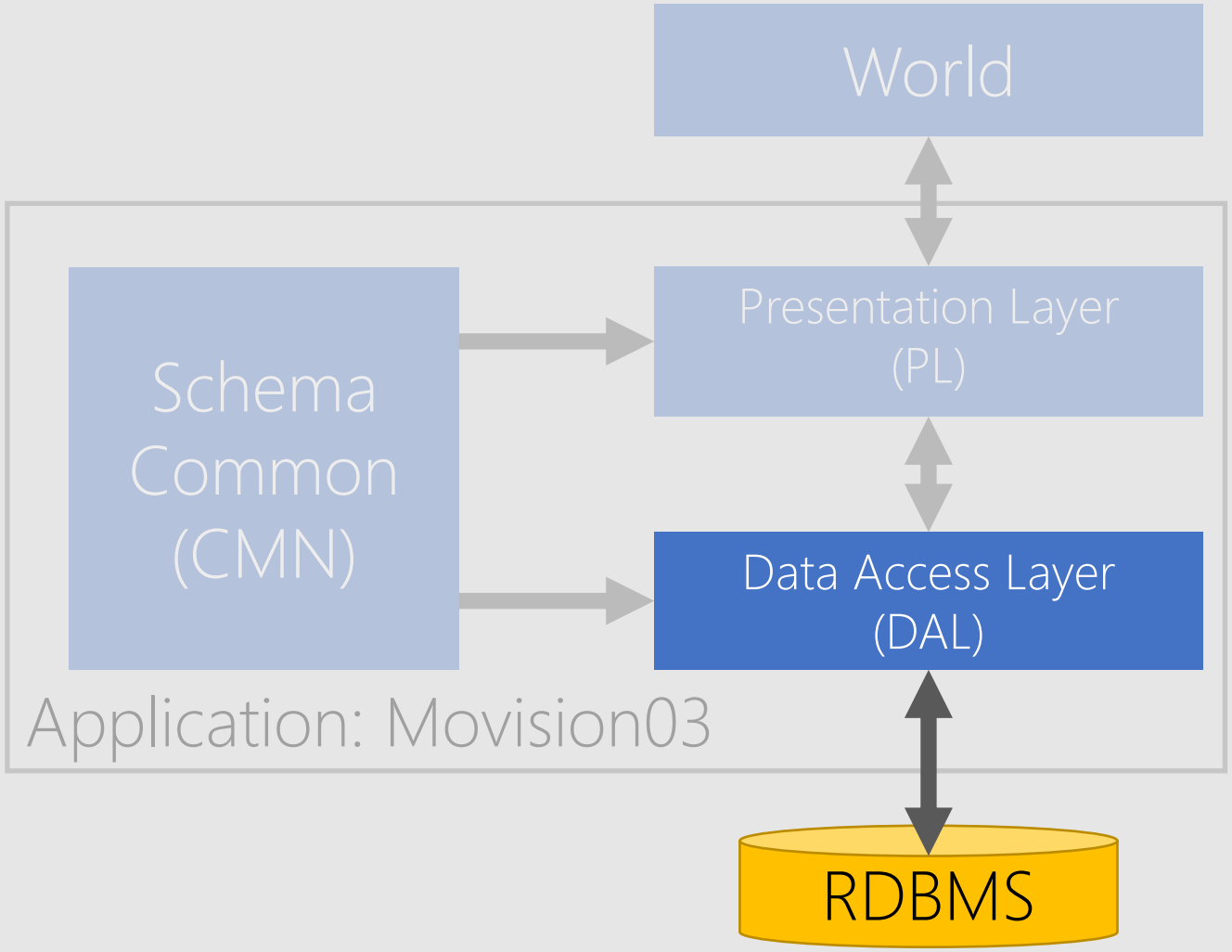
Physical Level × RDBMS



Physical Level × RDBMS



Physical Level × ORM

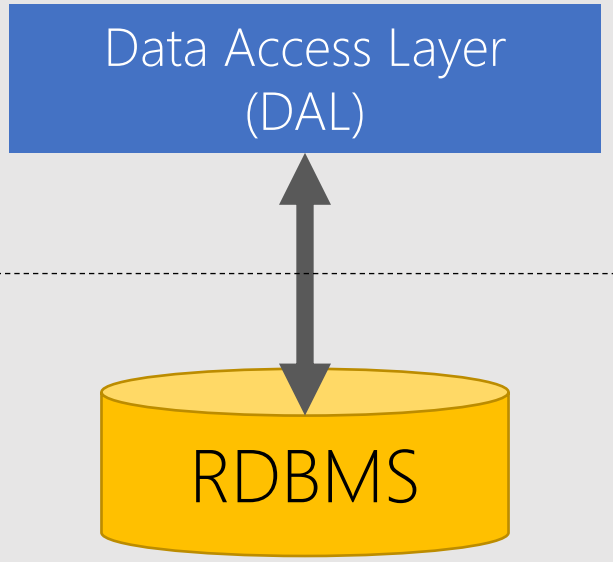


Physical Level × ORM

Object Relational Mapping (ORM)

Object

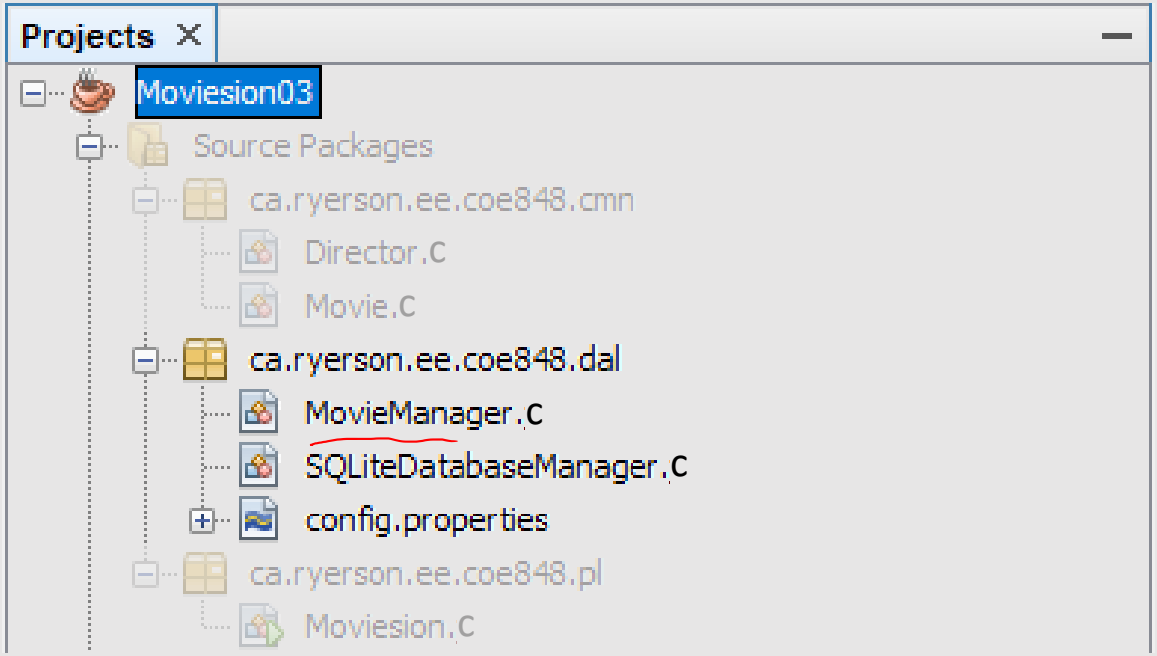
Function



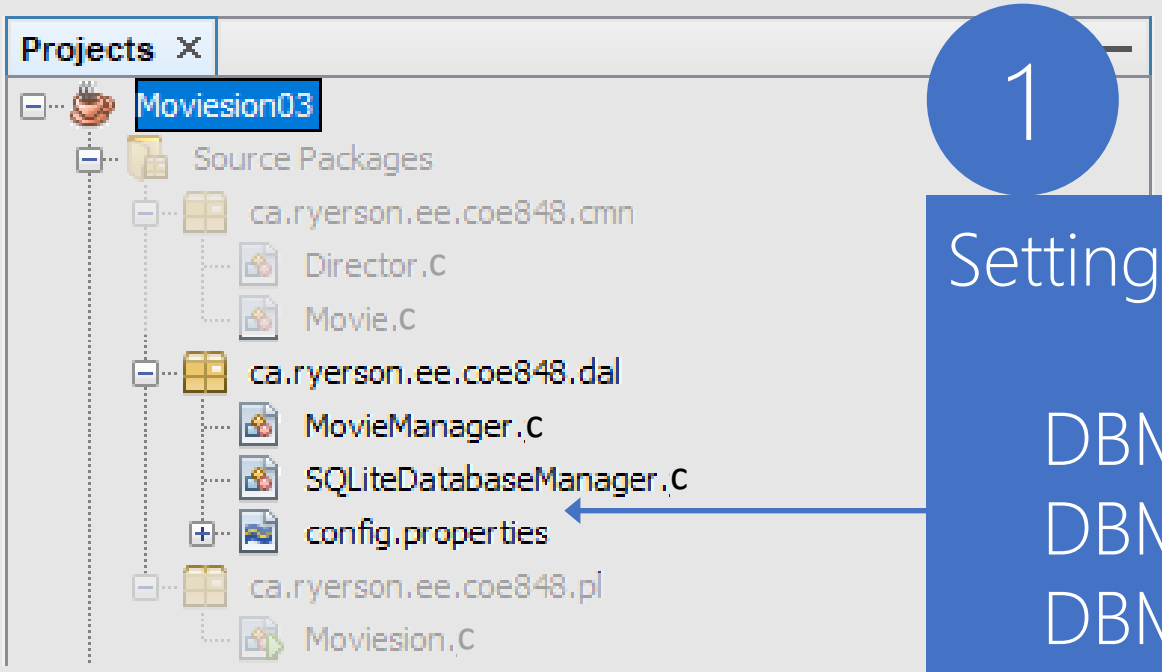
Relation
(Table)

SQL
(DML | DDL)

Physical Level × ORM



Physical Level × ORM

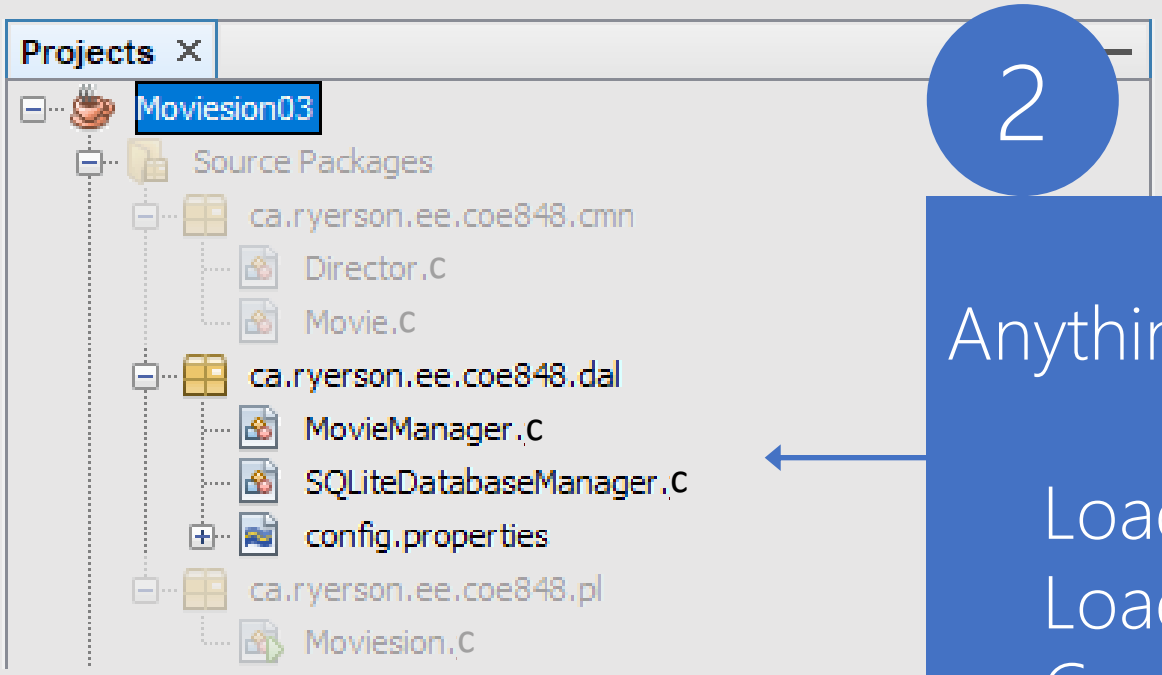


1

Settings to connect to DBMS:

- DBMS Location
- DBMS Username
- DBMS Password
- DB Name

Physical Level × ORM

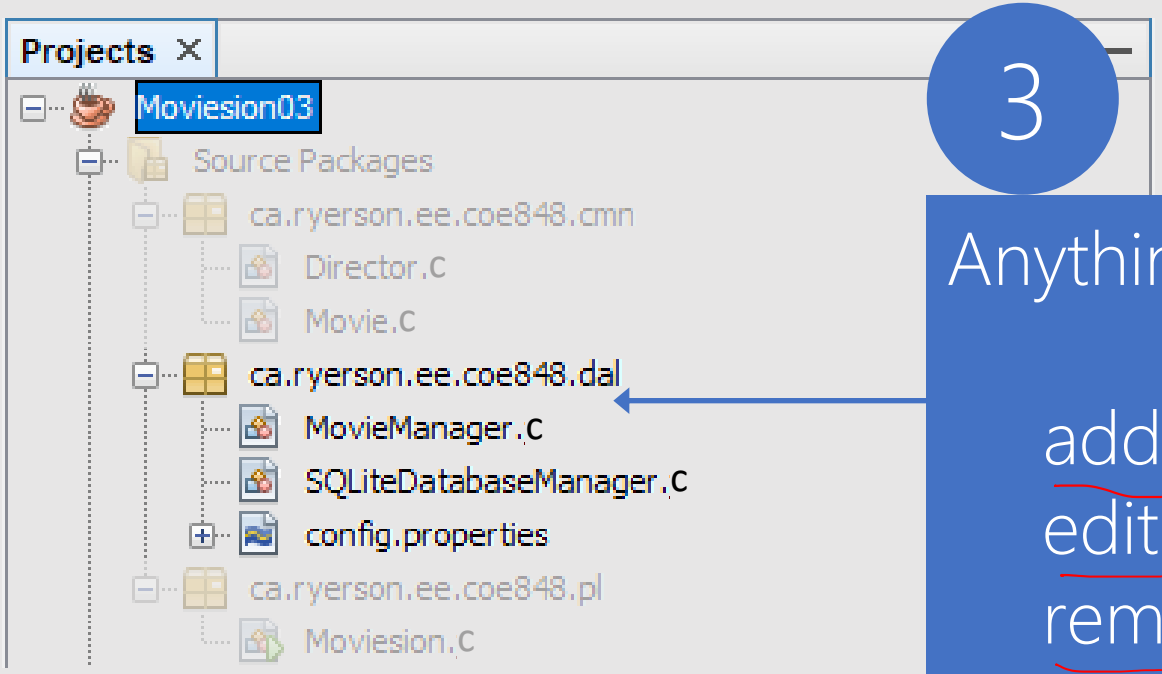


2

Anything related to DBMS connection:

- Loading SQLite library
- Loading settings from config file
- Create a connection

Physical Level × ORM

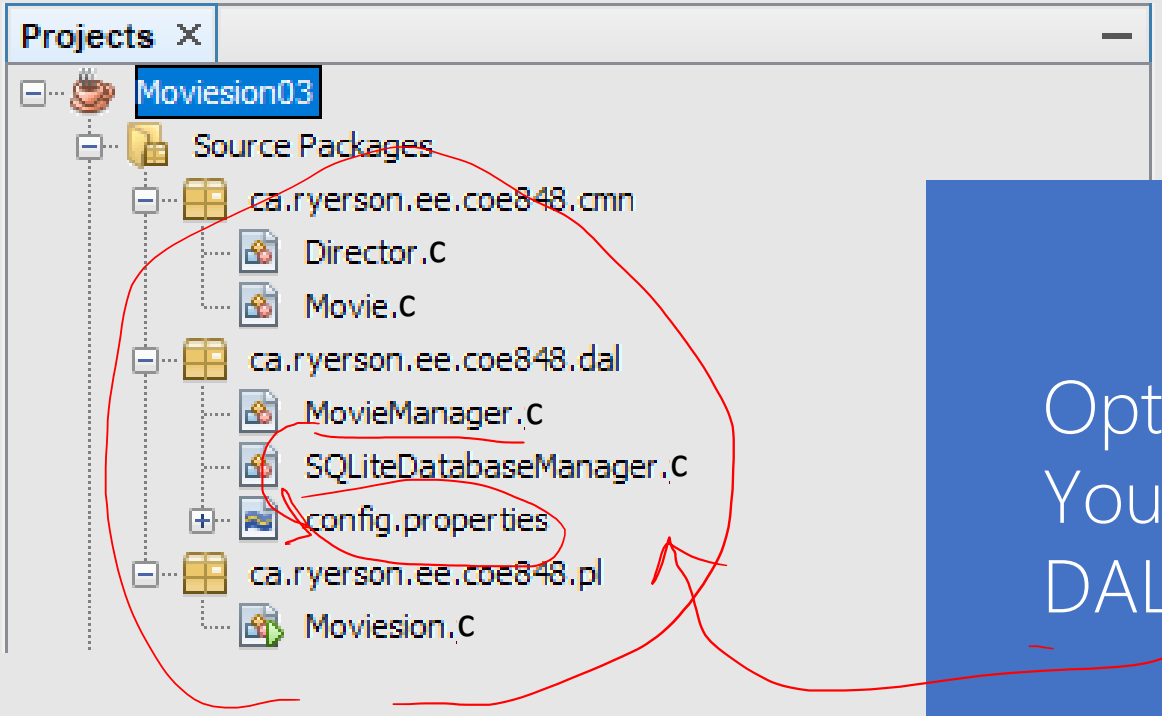


3

Anything related to Movie (ORM):

addMovie()	INSERT
editMovie()	UPDATE
removeMovie()	DELETE
getMovie() byId(int)	SELECT

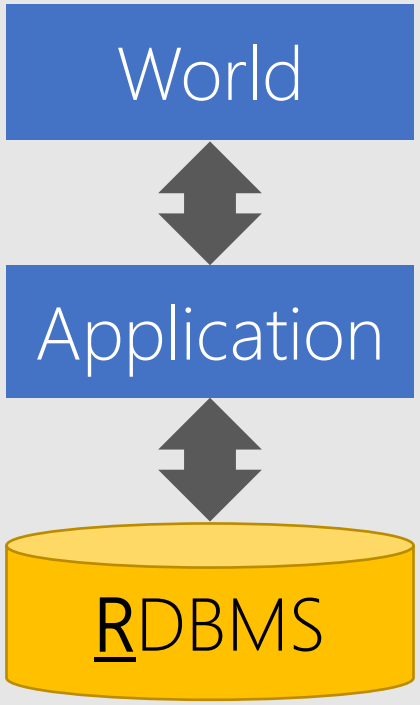
Physical Level × ORM



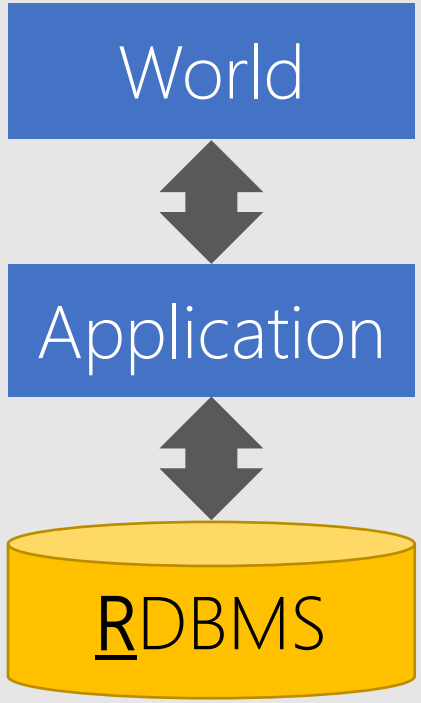
Optional:
You can follow 3-layered architecture (PL, DAL, DB) for your Lab5 project.

DB vs. APP Level Processing

How many movies do we have?



DB Level Processing

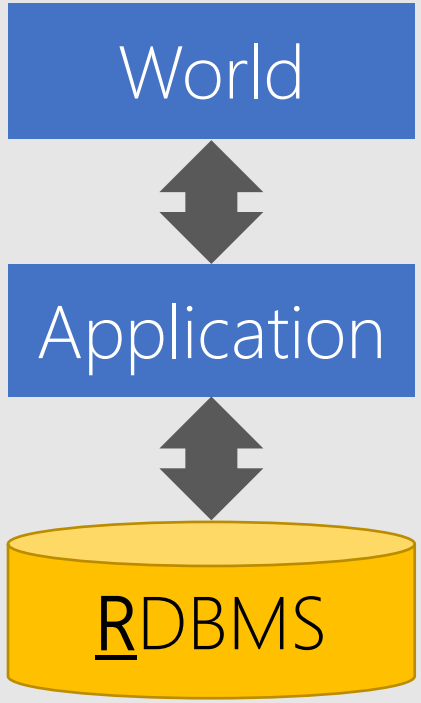


How many movies do we have?

In SQL: what is the movie count?

I) Count the movies
II) Return a single number

APP Level Processing



How many movies do we have?

I) Get all movies
II) Count the movies

Return all movies

DB vs. APP Level Processing

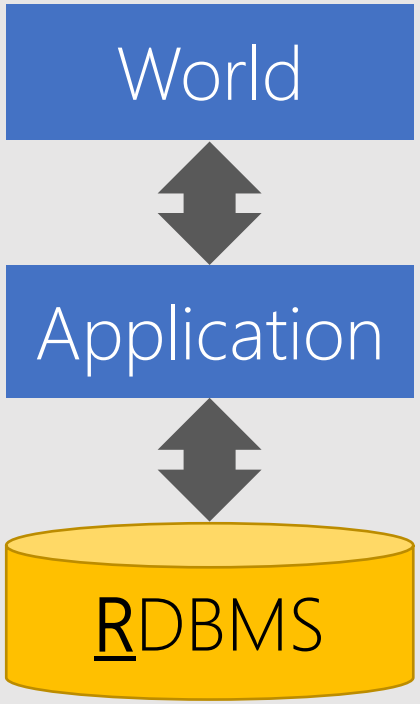
APP Level

+:

Only simple SQL
Able to do very complex tasks

–:

Slow, moving all data to app. level
Waste of network bandwidth



DB Level

+:

Fast, no need to move data
Fast, DBMS is a powerful machine

–:

Master SQL language
Not able to do very complex tasks

Ad hoc SQL Query

Ad hoc query is created to obtain info as need arises,
e.g., which director has made the most movies?

Contrast with a query that is predefined & routinely processed,
e.g., INSERT, UPDATE, DELETE, SELECT by Id