Database Management Systems

Vijaya Saradhi

IIT Guwahati

Mon, 24th Feb 2020

Queries on Example Database

Q6: Find the names of Sailors who have reserved a red AND a green Boat

```
(SELECT R.sid
FROM    Reserves AS R
JOIN    Boats AS B
ON      (R.bid = B.bid)
WHERE   B.color = 'red' AND R.sid

    IN

(SELECT R.sid
FROM    Reserves AS R
JOIN    Boats AS B
ON      (R.bid = B.bid)
WHERE   B.color = 'green');
```

Queries on Example Database

Q6: Find the names of Sailors who have reserved a red AND a green Boat

```
SELECT   S1.sname
FROM     Sailors AS S1
JOIN     (SELECT R.sid
          FROM     Reserves AS R
          JOIN     Boats AS B
          ON       (R.bid = B.bid)
          WHERE    B.color = 'red' AND R.sid
               IN
          (SELECT R.sid
          FROM     Reserves AS R
          JOIN     Boats AS B
          ON       (R.bid = B.bid)
               WHERE    B.color = 'green')) AS S2
ON  S1.sid = S2.sid
```

Queries on Example Database

Q7: Find the names of Sailors who have reserved at least two boats

```
CREATE TABLE Temp1 AS
     (SELECT      S.sid, S.sname, R.bid
     FROM     Sailors AS S, Reserves AS R
     WHERE    S.sid = R.sid);

SELECT T1.sname
FROM     Temp1 AS T1
JOIN     Temp1 AS T2
ON       T1.sid = T2.sid
WHERE    (T1.bid <> T2.bid)
```

Queries on Example Database

Q7: Find the names of Sailors who have reserved at least two boats

```
SELECT   T1.sname
FROM     (SELECT      S.sid, S.sname, R.bid
          FROM        Sailors AS S, Reserves AS R
          WHERE       S.sid = R.sid) AS T1
JOIN     (SELECT      S.sid, S.sname, R.bid
          FROM        Sailors AS S, Reserves AS R
          WHERE       S.sid = R.sid) AS T2
ON       T1.sid = T2.sid
WHERE    (T1.bid <> T2.bid);
```

Queries on Example Database

Q8: Find the sids of Sailors with age over 20 who have not reserved a red boat

```
SELECT S1.sid
FROM   Sailors AS S1
WHERE  S1.age >= 20 AND S1.sid
    NOT IN

(
SELECT S2.sid
FROM   Sailors AS S2
JOIN   Reserves AS R2
JOIN   Boats As B2
ON     (S2.sid = R2.sid AND R2.bid = B2.bid)
WHERE  (B.color = 'red'))
)
```

Example 09 - Expressions in SELECT

Compute increments of ratings who have sailed two different boats on same day

```
SELECT    S1.name, S1.rating + 1 as rating
FROM      Sailors AS S1
JOIN      Reserves AS R1
JOIN      Reserves AS R2
WHERE     S1.sid = R1.sid
AND       S1.sid = R2.sid
AND       R1.day = R2.day
AND       R1.bid <> R2.bid
```

Example 10 - Regular Expressions

Find the ages of sailors whose name begins and ends with B and has at least three characters

```
SELECT      S1.age
FROM        Sailors AS S1
WHERE       S1.sname LIKE 'B_%B'
```

LIKE regular expression

- % denote wild-card symbol. It matches zero or more characters
- _ denote matching for exactly one arbitrary character
- B_ denote any string starting with B followed by exactly one character
- B_%B denote any string starting with B followed by exactly one character followed by zero or more characters and ending with B

Example 11(a) - Union

Find the names of sailors who have reserved a red or green boat

```
(SELECT S1.sname
FROM    Sailors AS S1
JOIN    Reserves AS R1
JOIN    Boats AS B1
ON      S1.sid = R1.sid
AND     R1.bid = B1.bid
WHERE   B1.color = 'red')

UNION

(SELECT S1.sname
FROM    Sailors AS S1
JOIN    Reserves AS R1
JOIN    Boats AS B1
ON      S1.sid = R1.sid
AND     R1.bid = B1.bid
WHERE   B1.color = 'green')
```

Example 11(b) - Union

Find the names of sailors who have reserved a red or green boat

```
SELECT   S1.sname
FROM     Sailors AS S1
JOIN     Reserves AS R1
JOIN     Boats AS B1
ON       S1.sid = R1.sid
AND      R1.bid = B1.bid
WHERE    B1.color = 'red' OR B1.color = 'green'
```

Example 12 - Intersection

Find the names of sailors who have reserved a red and a green boat

```
SELECT  S1.sname
FROM    Sailors AS S1
JOIN    Reserves AS R1
JOIN    Boats AS B1
ON      S1.sid = R1.sid
AND     R1.bid = B1.bid
WHERE   B1.color = 'red'
AND     (S1.sid, S1.sname, S1.rating, S1.age)

IN

(SELECT *
FROM    Sailors AS S1
JOIN    Reserves AS R1
JOIN    Boats AS B1
ON      S1.sid = R1.sid
AND     R1.bid = B1.bid
WHERE   B1.color = 'green')
```

Example 13 - Difference

Find sids of all sailors who have reserved red boat but not green boat

```
SELECT  S1.sid
FROM    Sailors AS S1
JOIN    Reserves AS R1
JOIN    Boats AS B1
ON      S1.sid = R1.sid
AND     R1.bid = B1.bid
WHERE   B1.color = 'red'
AND     (S1.sid, S1.sname, S1.rating, S1.age)

NOT IN

(SELECT *
FROM    Sailors AS S1
JOIN    Reserves AS R1
JOIN    Boats AS B1
ON      S1.sid = R1.sid
AND     R1.bid = B1.bid
WHERE   B1.color = 'green')
```

Example 14 - Union

Find all sids of sailors who have a rating of 10 or reserved boat 104

```
(SELECT S1.sid
FROM    Sailors AS S1
WHERE   S1.rating = 10)

UNION

(SELECT R1.sid
FROM    Reserves AS R1
WHERE   R1.bid = 104)
```

Example 15 - Nested Queries

Find the names of Sailors who have reserved a red boat

```
SELECT   S1.sname
FROM     Sailors AS S1
WHERE    S1.sid
IN
     (SELECT R1.sid
     FROM     Reserves AS R1
     WHERE    R1.bid
     IN
          (SELECT      B1.bid
           FROM        Boats AS B1
           WHERE       B1.color = 'red')
     )
```

Example 16 - Correlated Nested Queries

Correlated nested queries

- The inner sub-query has been completely independent of the outer query
- In general, the inner sub-query could dependent on the row currently being examined in the outer query
- Such queries are known as Correlated nested queries

Find the names of sailors who have reserved boat number 103

```
SELECT   S1.sname
FROM     Sailors AS S1
WHERE    EXISTS
     (SELECT *
     FROM     Reserves AS R1
     WHERE    R1.bid = 103
     AND      S1.sid = R1.sid
     )
```

Example 17 - Correlated Nested Queries

Correlated nested queries

- For each Sailor row S1 test whether the set of Reserves row R1 such that `R1.bid = 103 AND S1.sid = R1.sid`
- If the above test is true, sailor S1.sid has reserved boat 103
- Retrieve all such tuples
- The sub-query clearly depends on the current or of S1
- The sub-query must be evaluated for each row in S1

Example 18 - Complex Nested Queries

Find the names of sailors who have reserved both red and green boat

```
SELECT   S1.sname
FROM     Sailors AS S1
WHERE    S1.sid
IN
     (SELECT      R1.sid
     FROM     Reserves AS R1
     JOIN     Boats AS B1
     ON       R1.bid = B1.bid
     WHERE    B1.color = 'red'

     AND      R1.bid
     IN

     (SELECT      R2.sid
     FROM     Reserves AS R2
     JOIN     Boats AS B2
     ON       R2.bid = B2.bid
     WHERE    B2.color = 'green'
```

Example 19(a) - Complex Correlated Nested Queries

Find the names of sailors who have reserved all boats

```
SELECT   S1.sname
FROM     Sailors AS S1
WHERE
     NOT EXISTS
         (SELECT     B1.bid
          FROM       Boats AS B1
          WHERE      B1.bid

          NOT IN

          (SELECT        R1.bid
              FROM       Reserves AS R1
              WHERE      R1.sid = S1.sid)
         )
```

Example 19(b) - Complex Correlated Nested Queries

Find the names of sailors who have reserved all boats

```
SELECT   S1.sname
FROM     Sailors AS S1
WHERE

    NOT EXISTS
        (SELECT     B1.bid
         FROM       Boats AS B1
         WHERE

         NOT EXISTS
             (SELECT     R1.bid
             FROM        Reserves AS R1
             WHERE       R1.sid = S1.sid
             AND         R1.bid = B1.bid
             )
        )
```

Example 20(a) - Insertion

Insertion into table

```
INSERT INTO Sailors VALUES(75, 'New Sailor 01', 9, 27);
INSERT INTO Reserves VALUES(75, 101, '1998-09-03');
INSERT INTO Boats VALUES(105, 'New Boat', 'green');
```

Example 20(a) - Update

Insertion into table

```
%Update one row
UPDATE Sailors set sname='New Sailor 07' WHERE sid=75;
%Updates several rows
UPDATE Sailors set sname='New Sailor 08' WHERE rating=7;
```

Example 20(a) - Delete

Insertion into table

```
%Delete one row
DELETE FROM Sailors where sid=75;
%Deletes several rows
DELETE FROM Sailors WHERE rating=7;
```

Set Comparison Operators

Operators

IN	a given attribute is a member of a specified set
EXISTS	given set is nonempty or not
op ANY	with example
op ALL	with example

ANY operator

Example

- Find sid's whose rating is better than some sailor named Horatio
- Inner query results a set of ratings of Horatios' say $\{3, 5, 7, 9\}$
- Every row of outer query is examined with the obtained set.
- If the rating is $>$ ANY from the set $\{3, 5, 7, 9\}$ then the row will be part of the output relation
- That is the one row from S1 and its corresponding rating is, say, 4 and the sailor name is, say, Art.
- $4 >$ ANY $\{3, 5, 7, 9\}$ condition satisfies and Art is included in the result table

```
SELECT   S1.sname
FROM     Sailors AS S1
WHERE    S1.rating > ANY (
         SELECT   S2.rating
         FROM     Sailors AS S2
         WHERE    S2.name = 'Horatio')
```

ALL operator

Example

- Find the sid's with highest rating
- Inner query results a set of ratings from Sailors say $\{7, 1, 8, 8, 10, 7, 10, 9, 3, 3\}$
- Every row of outer query is examined with the obtained set.
- If the rating is $\geq$ ALL of the ratings in $\{7, 1, 8, 8, 10, 7, 10, 9, 3, 3\}$ then the row will be part of the output relation

```
SELECT   S.sid
FROM     Sailors AS S
WHERE    S.rating >= ALL (
         SELECT   S2.rating
         FROM     Sailors AS S2)
```

Pit falls

- Find name and age of oldest sailor
- Intent is to find maximum age of a sailor and his/her name
- Use of MAX along with other attributes in a relation may not make sense
- The following query is illegal

```
SELECT   S.sname, MAX(S.age)
FROM     Sailors AS S;
```

Altenate query formulation

Alternate query

- Aggregation operation result in a relation
- = in where clause is being compared with a relation
- SQL converts it to a single value
- This query is still legal and yields the correct output

```
SELECT  S.sname, S.age
FROM    Sailors AS S
WHERE   S.age =
        (SELECT MAX(S2.age)
            FROM Sailors AS S2);
```

One more example

Q30 (notation from text book)

- Find names of sailors who are older than oldest sailor with a rating of 10

```
SELECT  S.sname
FROM    Sailors AS S
WHERE   S.age > (
        SELECT MAX(S2.age)
        FROM    Sailors AS S2
        WHERE S2.rating = 10);
```

One more example: Alternate formulation

Q30 (notation from text book)

- Find names of sailors who are older than oldest sailor with a rating of 10

```
SELECT  S.sname
FROM    Sailors AS S
WHERE   S.age > ALL (
        SELECT S2.age
        FROM    Sailors AS S2
        WHERE S2.rating = 10);
```

NULL Values

Example

sid	sname	rating	age
98	Dan	$\perp$	39

Result of logical expressions

$\perp$ `rating = 8? rating > 8? rating < 8?`

true `rating = 8 OR age < 40?`

$\perp$ `rating = 8 AND age > 40?`

Rules

$\perp$ NOT $\perp$

true OR: one of the argument is true

false OR: one of the argument is false

Truth tables

Operators - I

AND	True	False	Null
True	True	False	Null
False	False	False	Null
Null	Null	Null	Null

OR	True	False	Null
True	True	True	Null
False	True	False	Null
Null	Null	Null	Null

Impact on SQL constructs

- WHERE clause eliminates rows for which qualification does not evaluate to `true`
- The above includes `false` or $\perp$
- Row elimination has consequences on nested queries involving `EXISTS` or `UNIQUE`
- SQL duplicate definition
  - Two rows are considered duplicate if corresponding columns are equal
  - or both contain $\perp$
- Contrast to this definition $\perp$ == $\perp$ evaluates to $\perp$
- In the context of duplicates the above results in `true`

Impact on SQL constructs

- `COUNT(*)` evaluates $\perp$ to true
- Arthimetic operators involving $\perp$ results in $\perp$
- All aggregate operations COUNT, SUM, AVG, MIN, MAX simply discards $\perp$ values

Constraints in table creation

rating between 1 and 10

```
CREATE TABLE Sailors(
        sid INT,
        sname CHAR(10),
        rating INT,
        age FLOAT,
        CHECK( rating >= 1 AND rating <= 10)
    );
```

Constraints in table creation

Constraint: Interlake boat cannot be reserved

```
CREATE TABLE Reserves(
        sid INT,
        bid INT,
        day DATE,
        FOREIGN KEY (sid) REFERENCES Sailors,
        FOREIGN KEY (bid) REFERENCES Boats,
        CONSTRAINT noInterlake
        CHECK( 'Interlake' <>
            (SELECT B.bname
             FROM   Boats as B
             WHERE  B.bid = Reserves.bid
             )
        )
);
```

View Definitions - 01

Virtual Tables

- Relations defined using `CREATE TABLE` statement
- They actually exist in the database
- They are persistent
- Relations defined using `CREATE TEMPORARY TABLE` statemet
- They exist till certain period
- That is SQL system stores tables in some physical organization
- There is another class of SQL relations called views

View Definitions - 02

Virtual Tables

- Views do not exist physically
- They are defined by an expression much like a query
- View in turn be queried as if they exist physically
- In some cases they can be modified
- That is perform `INSERT`, `UPDATE`, `DELETE` operations on views

Declaring Views

Syntax Elements

Simple form of view definition is:

- The keyword CREATE VIEW
- The name of the view
- They keyword AS
- A query Q

About Q

Q is the definition of the view

Declaring Views

Syntax Elements

Simple form of view definition is:

- The keyword CREATE VIEW
- The name of the view
- They keyword AS
- A query Q

Complete Syntax

```
CREATE VIEW [view-name] AS [Q];
```

Creating Views

Example - 01

`Movie(title, year, length, inColor, studioName, producerC)`

```
CREATE VIEW ParamountMovies AS
    SELECT title, year
    FROM Movie
    WHERE studioName = 'Paramount';
```

Querying Veiws

Example - 02

List titles of movies released in 1979 by Paramount studio from the view ParamountMovies

```
SELECT   title
FROM     ParamountMovies
WHERE    year = 1979;
```

Querying Veiws

Example - 03 internal conversion

List titles of movies released in 1979 by Paramount studio from the view ParamountMovies

```
SELECT   title
FROM     Movie
WHERE    studioName='Paramount' and year = 1979;
```

Querying Views AND tables

Example - 04

Query both view and table

```
SELECT   DISTINCT starName
FROM     ParamountMovies, StarsIn
WHERE    title='Top Gun' and year = 1986;
```

Creating Views

Example - 05 - Renaming attributes

`Movie(title, year, length, inColor, studioName, producerC)`

```
CREATE VIEW ParamountMovies(movieTitle, yr) AS
    SELECT  title, year
    FROM    Movie
    WHERE   studioName = 'Paramount';
```

Modifying Views - 01

Example

- Two types of views are created
- Read only view
- Updatable view

Modifying Views - 02

Example

- Updatable view should include the primary key
- For example, the primary key for `Movie` table is: (title, year, startName)
- Created view has all the three attributes then modification is:

```
1    INSERT INTO ParamountMovies('Top Gun 02', 2020, 'Mr.
     ABCD');
2
```

- The record is inserted into the base table that is Movie
- The attributes length, inColor, producer assumes default value or NULL

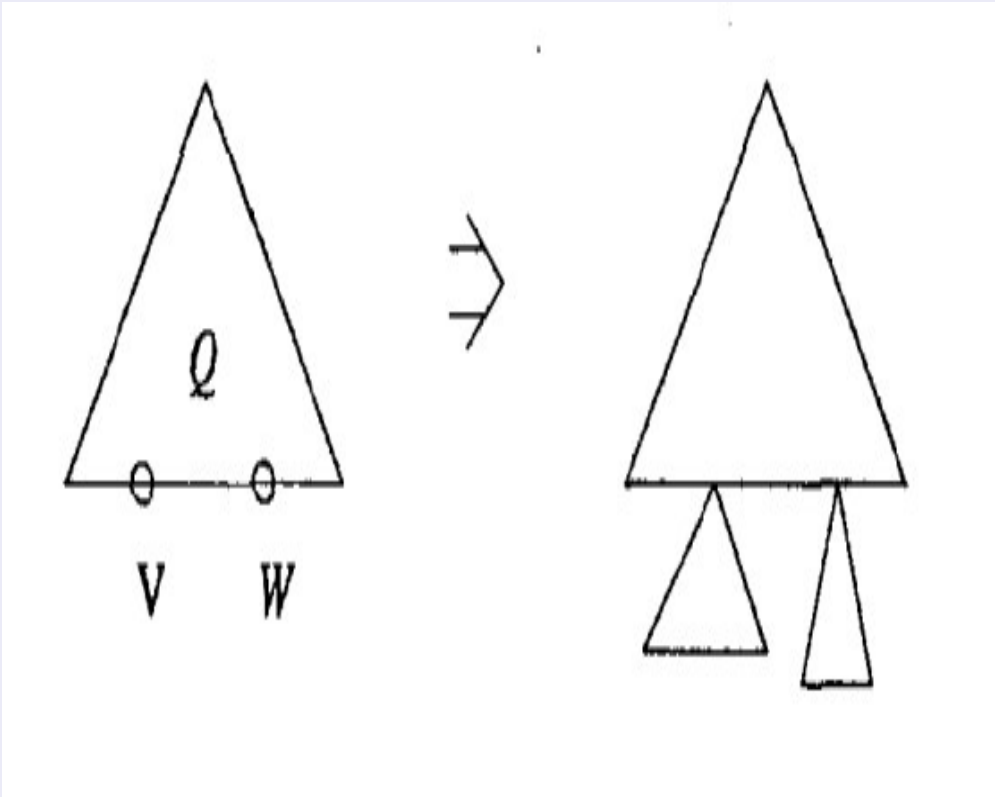
Modifying Views - 03

Example

```
DELETE
FROM      ParamountMovies
WHERE     title LIKE '%Trek%';
```

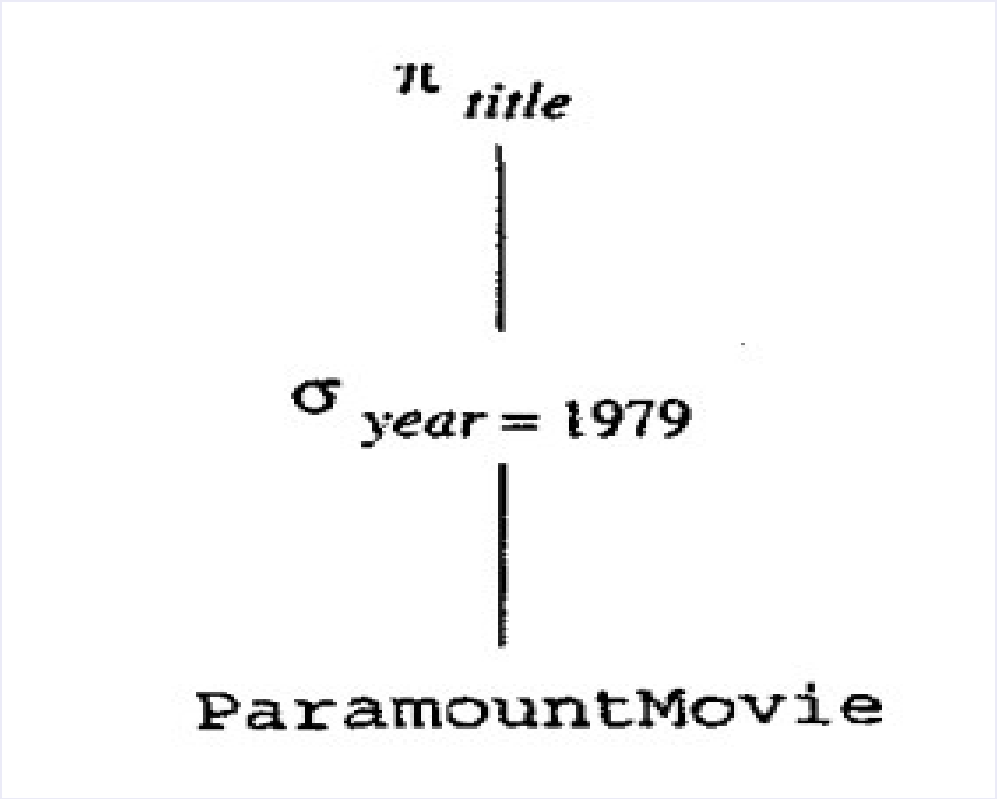
Interpreting Queries Involving Views

Interpretation - 01



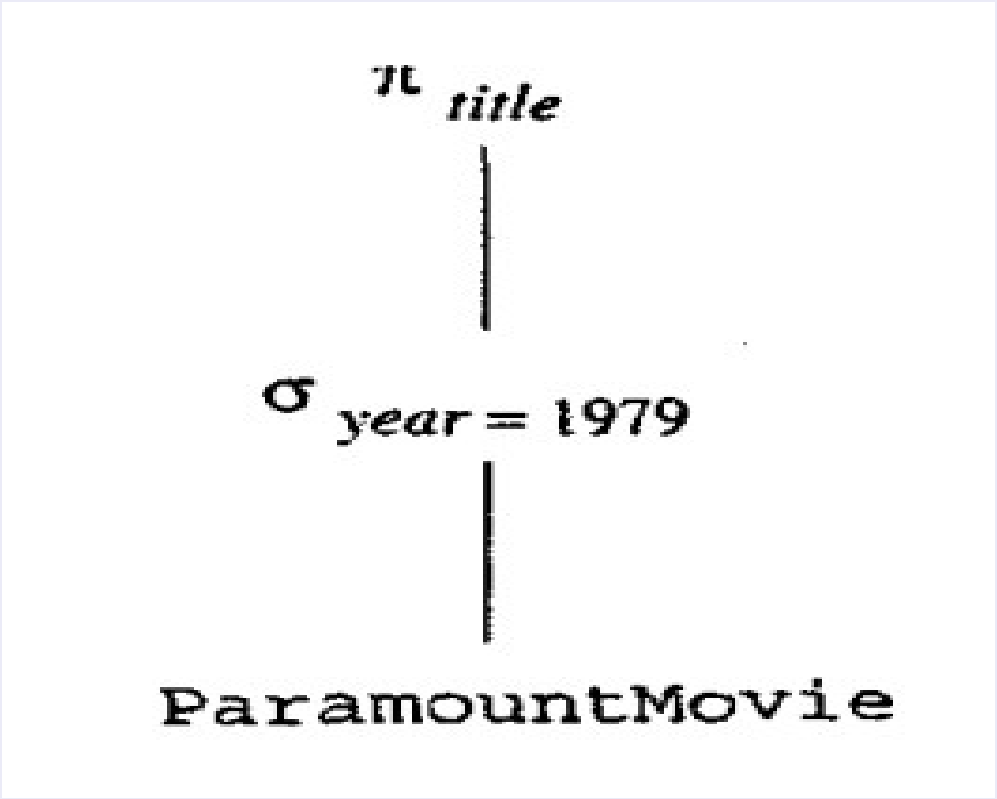
Interpreting Queries Involving Views

Interpretation - 02



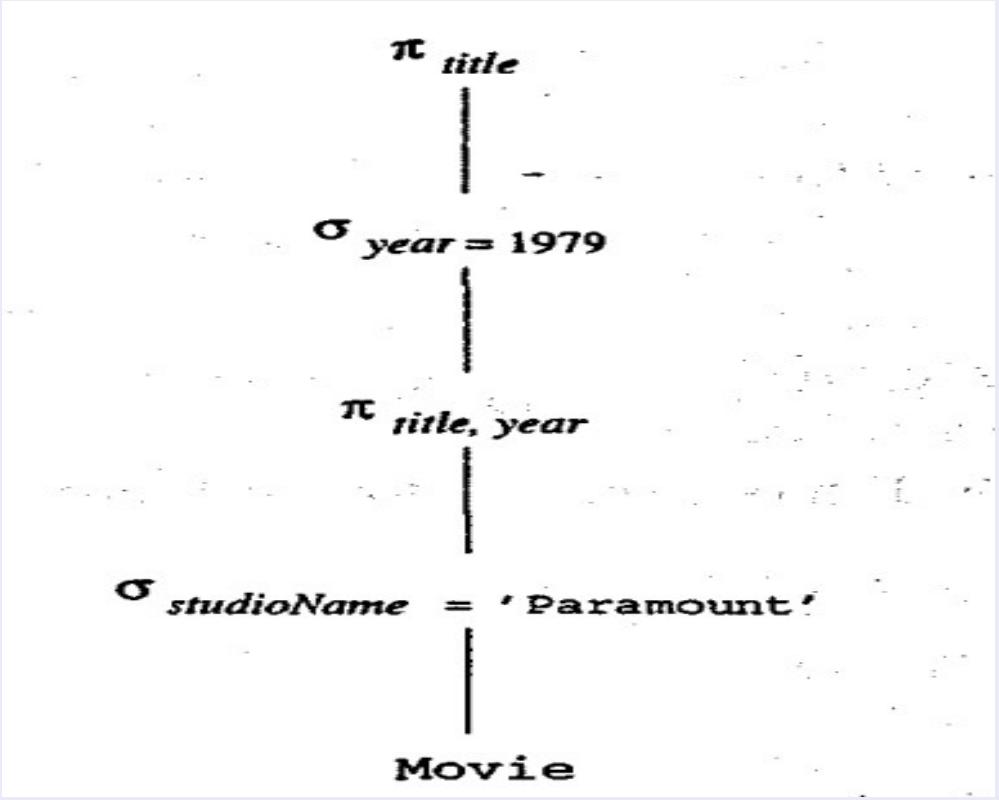
Interpreting Queries Involving Views

Interpretation - 03



Interpreting Queries Involving Views

Interpretation - 04



Types of VIEWS

Types

- Single-table projection and restrictions
- Calculated columns
- Translated columns
- Grouped views
- Union-ed views
- Joins in views
- Nested views

Types of VIEWS

Calculated columns

Personnel(emp_id, salary, commision, $\cdots$)

```
CREATE VIEW Payroll AS
    SELECT emp_id, (salary + COALESCE(commission), 0.00)
    FROM    Personnel;
```

COALESCE returns a non-null value in the given list

Types of VIEWS

Translated columns

T1(a11, a12); T2(a21, a22);

```
CREATE VIEW temp_view AS
    SELECT T1.a21, T2.a22
    FROM   T1, T2
    WHERE  T1.a11 = T2.a21;
```

Types of VIEWS

Grouped Views

```
CREATE VIEW BigSales AS
    SELECT state_code, MAX(sales_amount)
    FROM   Sales
    GROUP BY state_code;
```

Types of VIEWS

UNION-ed Views

```
CREATE VIEW UnionView AS
(SELECT  *
FROM     T1
WHERE    a11 = 1)
     UNION
(SELECT  *
FROM     T2
WHERE    a21 = 2)
```

Types of VIEWS

Nested Views

```
CREATE VIEW all_boats AS SELECT * FROM boats;
CREATE VIEW red_boats AS SELECT * from all_boats where bcolor
    ='red';
```

Dropping VIEWS

Droping

```
DROP VIEW red_boats;
DROP VIEW all_boats;
```