CS245: Databases

SQL

Vijaya saradhi

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

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

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:

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

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

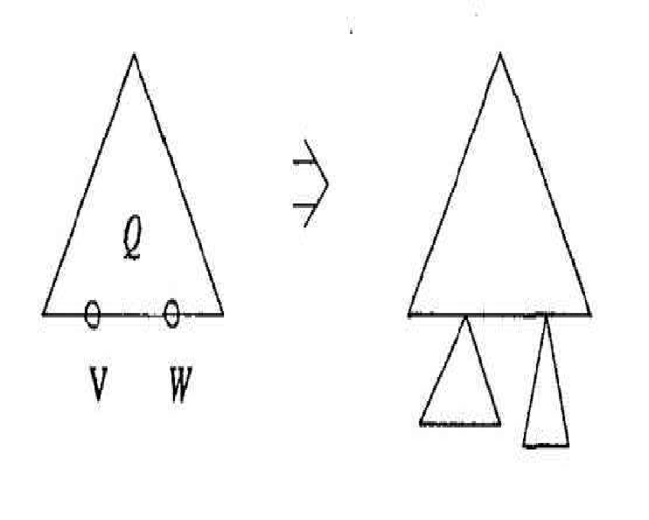
Modifying Views - 03

Example

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

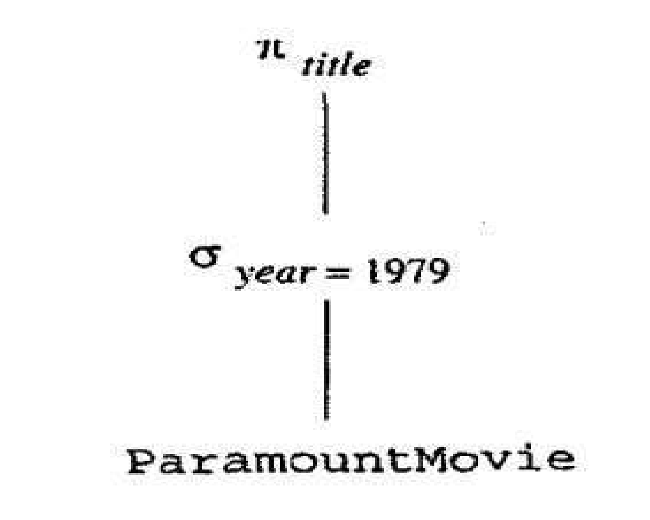
Interpreting Queries Involving Views

Interpretation - 01



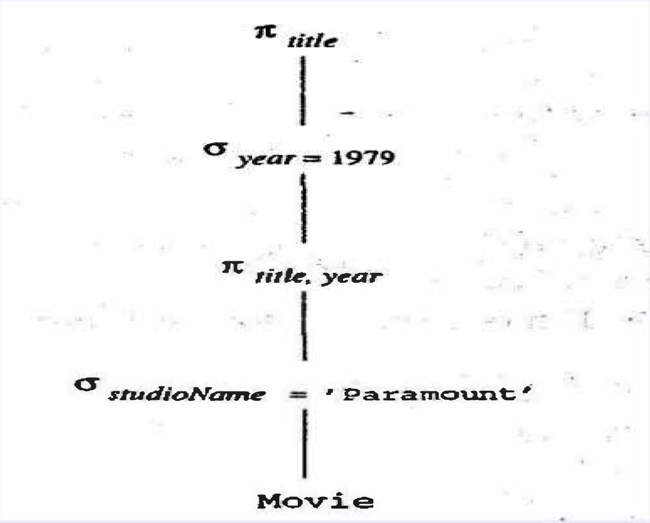
Interpreting Queries Involving Views

Interpretation - 02



Interpreting Queries Involving Views

Interpretation - 03



Types of VIEWS

Types

- Single-table projection and restrictions
- Calculated 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

Another example

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='re
```

Dropping VIEWS

Droping

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