CMPT 321 FALL 2017

SQL queries with views

Lecture 03.05

By Marina Barsky

Views

- A view is a "virtual table", a relation defined in terms of the contents of other tables and views
- Views take very little space to store the database contains only the definition of a view, not a copy of all the data that it presents
- In contrast, a relation whose value is really stored in the database is called a *base table*

Example

CREATE VIEW DMovies AS

- SELECT title, year, length
- **FROM Movie**
- WHERE studioName = 'Disney';

Querying a View

• Query a view as if it were a base table.

Examples SELECT title FROM DMovies WHERE year = 1990;

SELECT DISTINCT starName

FROM DMovies, StarsIn

WHERE DMovies.title = StarsIn.movietitle AND DMovies.year

= StarsIn.movieyear;

View on more than one relation

CREATE VIEW MoviesAndStars AS SELECT Movies.Title as a title, Movies.year as year, MovieStar.name as star MovieStar WHERE Movie.title= StarsIn.movietitle

AND MovieStar.name= StarsIn.movietitie AND MovieStar.name= StarsIn.starname;

SELECT * FROM MoviesAndStars;

movie (<u>title</u>, <u>year</u>, length, incolor, studio, producer_cert) star (<u>name</u>, address, gender, birthdate) starsIn (<u>movie_title</u>, <u>movie_year</u>, <u>star_name</u>) movieexec (<u>name</u>, address, cert, net_worth) studio (<u>name</u>, president_cert)

SQL queries with views

1. Find the stars who have worked for every studio.

CREATE VIEW MovieStarView AS

SELECT title, year, studio, star_name

FROM Movie, StarsIn

WHERE Movie.title = StarsIn.movie_title and Movie.Year = Starsin.Movie_year;

```
SELECT DISTINCT star_name
FROM MovieStarView X
WHERE NOT EXISTS ( Checks emptiness of the subquery.
SELECT studio
FROM Studio
EXCEPT
SELECT studio
FROM MovieStarView
WHERE star_name = X.star_name); Correlated subquery
```

2. Find the stars who have worked for Disney but no other studio.

CREATE VIEW MovieStarView AS

SELECT title, year, studioName, starName FROM Movies, StarsIn WHERE Movies.title = StarsIn.movieTitle AND Movies.Year = Starsin.MovieYear;

SELECT starName FROM MovieStarView X WHERE X.studioName='Disney' AND NOT EXISTS (SELECT * FROM MovieStar WHERE starName=X.starName AND studioName<>'Disney'

3. Find the stars who have worked for only one studio.

CREATE VIEW MovieStarView AS

);

SELECT title, year, studioName, starName FROM Movies, StarsIn WHERE Movies.title = StarsIn.movieTitle AND Movies.Year = Starsin.MovieYear;

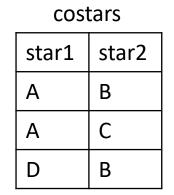
SELECT starName FROM MovieStarView X WHERE NOT EXISTS (SELECT * FROM MovieStarView WHERE starName=X.starName AND studioName<>X.studioName 4. For each star that has more than two movies with Paramount, find how many movies he/she has with Fox.

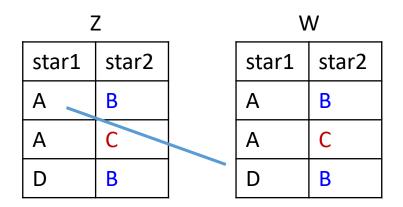
CREATE VIEW ParamountStars2 AS SELECT starName FROM MovieStarView WHERE studioName='Paramount' GROUP BY starName HAVING COUNT(title)>=2;

```
CREATE VIEW FoxStars AS
SELECT *
FROM MovieStarView
WHERE studioName='Fox';
```

SELECT starName, COUNT(title) as countFox FROM ParamountStars2 NATURAL LEFT OUTER JOIN FoxStars GROUP BY starName; 5. Find the stars who have co-starred with the same star.

CREATE VIEW costars AS SELECT X.starname AS star1, Y.starname AS star2 FROM StarsIn X JOIN StarsIn Y USING(title,year) WHERE X.starname <> Y.starname;





SELECT Z.star1, W.star1

FROM costars Z, costars W

WHERE Z.star2=W.star2 AND Z.star1<W.star1;

6. For each pair of co-stars give the number of movies each has starred in.

The result should be a set of (star1 star2 n1 n2) quadruples, where n1 and n2 are the number of movies that star1 and star2 have starred in, respectively. Observe that there might be stars with zero movies they have starred in.

CREATE VIEW starMovieCounts AS SELECT name AS star, COUNT(title) AS moviecount FROM Stars LEFT OUTER JOIN StarsIn ON name=starname GROUP BY name;

SELECT C.star1, C.star2, X.moviecount, Y.moviecount FROM costars C, starMovieCounts X, starMovieCounts Y WHERE C.star1=X.star AND C.star2=Y.star;

Summary: Views

- Provide modularization abstraction for SQL queries (like a function in programming languages)
- Limit the degree of exposure of the underlying tables to the outer world
- Allow to join and simplify multiple tables into a single virtual table
- Hide the complexity of data: provide <u>logical data</u> <u>independence</u>

In your program, retrieve data from the view: if the definition of underlining tables changes, you do not need to update your code – just re-write the view