

Views

DSCI 551

Wensheng Wu

Views

- A view is a “virtual table,” a relation that is defined in terms of the contents of other tables and views.
- Declare by:

```
CREATE VIEW <name> AS <query>;
```
- In contrast, a relation whose value is really stored in the database is called a *base table*.

Example: View Definition

- CanDrink(drinker, beer) is a view “containing” the drinker-beer pairs such that the drinker frequents at least one bar that serves the beer:
- Recall Frequents(drinker, bar), Sells(bar, beer, price)

```
CREATE VIEW CanDrink AS
SELECT distinct drinker, beer
FROM Frequents, Sells
WHERE Frequents.bar = Sells.bar;
```

Example: Accessing a View

- You may query a view as if it were a base table.
 - There is a limited ability to modify views if the modification makes sense as a modification of the underlying base table.

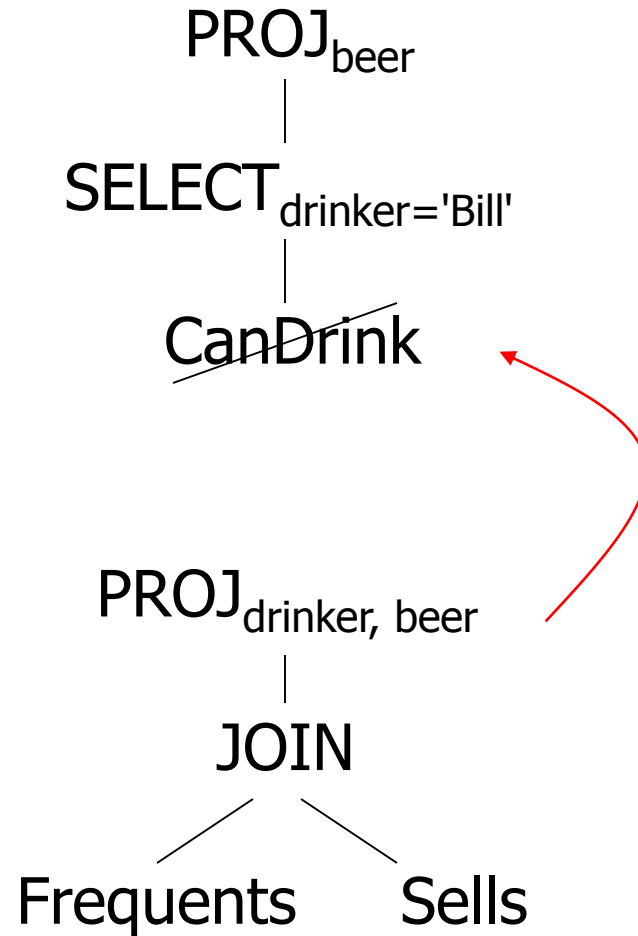
- Example:

```
select * from CanDrink
where drinker = 'Bill';
```

What Happens When a View Is Used?

- The DBMS starts by interpreting the query as if the view were a base table.
 - Typical DBMS turns the query into something like relational algebra.
- The queries defining any views used by the query are also replaced by their algebraic equivalents, and “spliced into” the expression tree for the query.

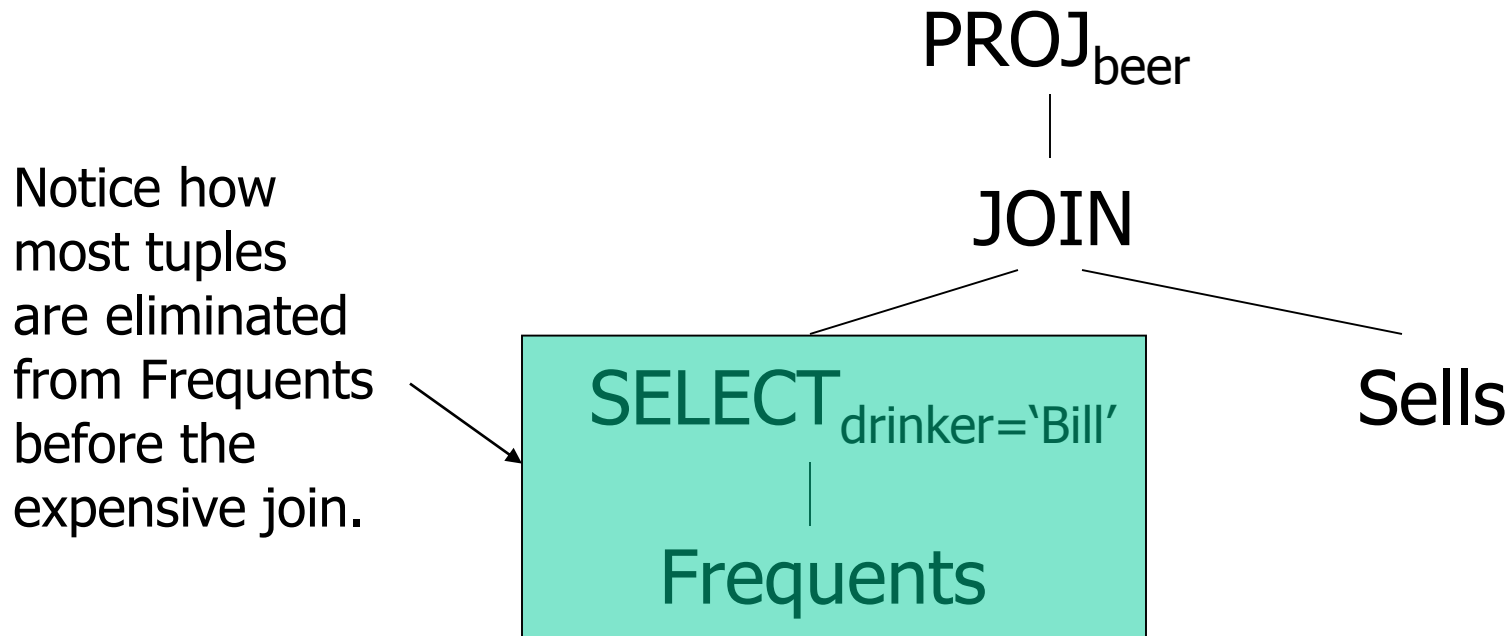
Example: View Expansion



DMBS Optimization

- It is interesting to observe that the typical DBMS will then “optimize” the query by transforming the algebraic expression to one that can be executed faster.
- Key optimizations:
 1. Push selections down the tree.
 2. Eliminate unnecessary projections.

Example: Optimization



More Examples: Defining Views

Views are relations, except that they are not physically stored.

Can be used for presenting different information to different users

Employee(ssn, name, department, project, salary)

```
CREATE VIEW Developers AS  
  SELECT name, project  
  FROM Employee  
  WHERE department = 'Development'
```

Payroll has access to all **Employees**, others only to **Developers**

A Different View

Purchase(buyer, seller, product, store, price)

Product(name, maker, category)

Person(name, city, phone)

```
CREATE VIEW LA-view AS
```

```
SELECT buyer, seller, product, store
```

```
FROM Person, Purchase
```

```
WHERE Person.city = 'LA' AND  
       Person.name = Purchase.buyer
```

We have a new virtual table:

LA-view(buyer, seller, product, store)

A Different View

We can later use the view:

```
SELECT name, store
FROM LA-view, Product
WHERE LA-view.product = Product.name AND
      Product.category = 'shoes'
```

What Happens When We Query a View ?

Recall: LA-view(buyer, seller, product, store)

```
SELECT name, LA-view.store
FROM   LA-view, Product
WHERE  LA-view.product = Product.name AND
       Product.category = 'shoes'
```



```
SELECT name, Purchase.store
FROM   Person, Purchase, Product
WHERE  Person.city = 'LA' AND
       Person.name = Purchase.buyer AND
       Purchase.product = Product.name AND
       Product.category = 'shoes'
```

Types of Views

- Virtual views:
 - Computed only on-demand – slow at runtime
 - Always up to date
- Materialized views
 - Precomputed offline – fast at runtime
 - Common in data warehouses
 - May have stale data

Reusing a Materialized View

- Suppose I have **only** the result of LAView:

```
SELECT buyer, seller, product, store
FROM   Person, Purchase
WHERE  Person.city = 'LA'   AND
       Person.name = Purchase.buyer
```

- and I want to answer the query

```
SELECT buyer, seller
FROM   Person, Purchase
WHERE  Person.city = 'LA'   AND
       Person.name = Purchase.buyer AND
       Purchase.product='gizmo'.
```

Can I answer the query using only the view?

Query Rewriting Using Views

Rewritten query:

```
SELECT buyer, seller  
FROM    LAView  
WHERE   product= 'gizmo'
```

Original query:

```
SELECT buyer, seller  
FROM    Person, Purchase  
WHERE   Person.city = 'LA'   AND  
        Person.name = Purchase.buyer AND  
        Purchase.product='gizmo'.
```

Another Example

- I still have **only** the result of LAView:

```
SELECT buyer, seller, product, store
FROM   Person, Purchase
WHERE  Person.city = 'LA'   AND
       Person.name = Purchase.buyer
```

- but I want to answer the query

```
SELECT buyer, seller
FROM   Person, Purchase
WHERE  Person.city = 'LA'   AND
       Person.name = Purchase.buyer AND
       Person.phone LIKE '206 543 %'.
```


And Now?

- I still have **only** the result of (slightly different) LAView:

```
SELECT buyer, seller, product, store
FROM Person, Purchase, Product
WHERE Person.city = 'LA' AND
      Person.name = Purchase.buyer AND
      Purchase.product = Product.name
```

- but I want to answer the query

```
SELECT buyer, seller
FROM Person, Purchase
WHERE Person.city = 'LA' AND
      Person.name = Purchase.buyer
```

And Now?

- I still have **only** the result of view SBS:

```
SELECT seller, buyer, Sum(Price)
FROM Purchase
WHERE Purchase.store = 'The Bon'
Group By seller, buyer
```

- but I want to answer the query

```
SELECT seller, Sum(Price)
FROM Purchase
WHERE Purchase.store = 'The Bon'
Group By seller
```

And what if it's the other way around?

Example

View SBS(seller, buyer, sp)

Seller	Buyer	Sum(price) sp
David	Bill	10
David	Jennifer	20
David	Steve	10
Bill	David	20
Bill	Mary	10

Query:

Seller	Sum(price)
David	?
Bill	?

Finally...

- I still have **only** the result of:

```
SELECT seller, buyer, Count(*)  
FROM Purchase  
WHERE Purchase.store = 'The Bon'  
Group By seller, buyer
```

- but I want to answer the query

```
SELECT seller, Count(*)  
FROM Purchase  
WHERE Purchase.store = 'The Bon'  
Group By seller
```

Example

View SBC(seller, buyer, cnt)

Seller	Buyer	count(*) cnt
David	Bill	2
David	Jennifer	4
David	Steve	2
Bill	David	5
Bill	Mary	2

Query:

Seller	count(*)
David	?
Bill	?