



Views

Materialized Views

Why use views?

- Hide some data from some users
- Make some queries easier / more natural
- Modularity of database access

Real applications tend to use lots and lots (and lots and lots!) of views

Why use (virtual) views?

- Hide some data from some users
- Make some queries easier / more natural
- Modularity of database access

Why use materialized views?

- Hide some data from some users
- Make some queries easier / more natural
- Modularity of database access
- *Improve query performance*

Virtual views

- View V = ViewQuery(R_1, R_2, \dots, R_n)
- Schema of V is schema of query result
- Query Q involving V , conceptually:

→ V := ViewQuery(R_1, R_2, \dots, R_n);
Evaluate Q

- In reality, Q rewritten to use R_1, \dots, R_n instead of V

Materialized views

- View V = ViewQuery(R_1, R_2, \dots, R_n)
- Create table V with schema of query result
- Execute ViewQuery and put results in V
- Queries refer to V as if it's a table

But...

- V could be very large
- Modifications to R_1, R_2, \dots, R_n \Rightarrow
recompute or modify V

Create Materialized View **CA-CS** As
 Select **C.cName**, **S.sName**
 From **College C**, **Student S**, **Apply A**
 Where **C.cName = A.cName** And **S.sID = A.sID**
 And **C.state = 'CA'** And **A.major = 'CS'**

+ Can use **CA-CS** as if it's a table (it is!)

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec

```

Create Materialized View CA-CS As
Select C.cName, S.sName
From College C, Student S, Apply A
Where C.cName = A.cName And S.sID = A.sID
And C.state = 'CA' And A.major = 'CS'

```

– Modifications to base data invalidate view

College

cName	state	enr

Student

sID	sName	GPA	HS

Apply

sID	cName	major	dec


```
Create Materialized View CA-CS As
Select C.cName, S.sName
From College C, Student S, Apply A
Where C.cName = A.cName And S.sID = A.sID
And C.state = 'CA' And A.major = 'CS'
```

– Modifications to base data invalidate view

| College : ~~inserts~~, ~~deletes~~, updates (cName, state)

| Student : ~~inserts~~, ~~deletes~~, updates (sName, sID)

| Apply : ~~inserts~~, ~~deletes~~, updates (cName, sID, major)

General Assertions ★

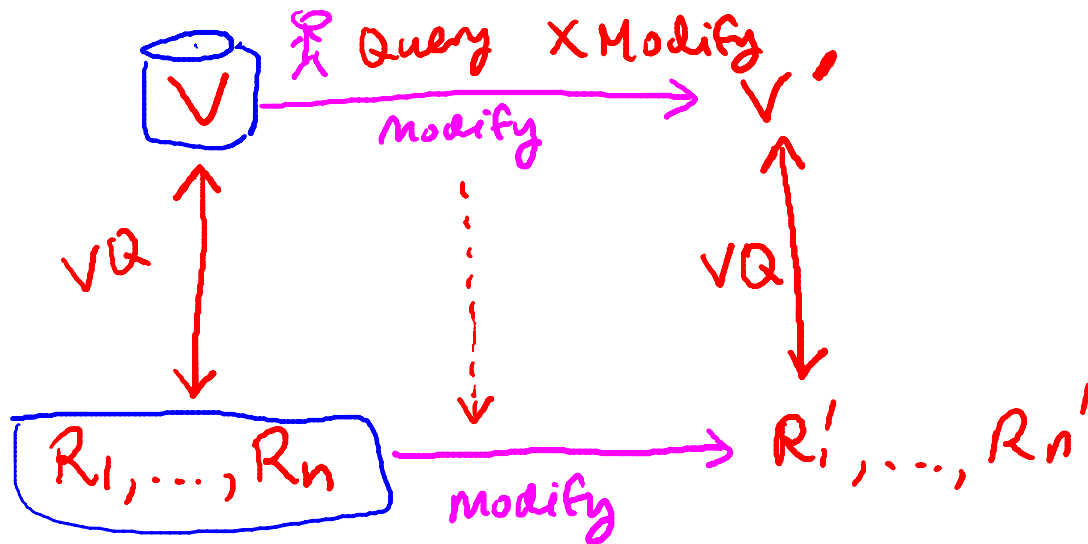
Queries over materialized views

- View $V = \text{ViewQuery}(R_1, R_2, \dots, R_n)$
- Create table V with schema of query result
- Execute ViewQuery and put results in V
- Queries refer to V as if it's a table

Modifications on materialized views?


- Good news: just update the stored table
- Bad news: base tables must stay in synch
 - ❖ Same issues as with virtual views

❖ Modifications to V must also modify base tables



Picking which materialized views to create

(Efficiency) benefits of a materialized view depend on:

- Size of data ←
 - Complexity of view ←
 - Number of queries using view
 - Number of modifications affecting view 
- * Query ←*
** Update ←*
trade off

→ Also “incremental maintenance” versus full recomputation

Indexes

Automatic query rewriting to use materialized views

```
Create Materialized View CA-Apply AS
Select sID, cName, major
From Apply A
where cName In
  (Select cName From College where state = 'CA')
```

↓

```
Select Distinct S.sID, S.GPA
From College C, Student S, CAApply A
where cName = A.cName And S.sID = A.sID
And S.GPA > 3.5 And C.state = 'CA' And A.Major = 'CS'
```

$[V_1 \ V_2 \ \dots \ V_{1000}] \leftarrow \textcircled{Q}$

Why use materialized views?

- Hide some data from some users
- Make some queries easier / more natural
- Modularity of database access
- *Improve query performance*