SQL – Views

Carla Teixeira Lopes

Bases de Dados
Mestrado Integrado em Engenharia Informática e Computação, FEUP

Based on Jennifer Widom slides

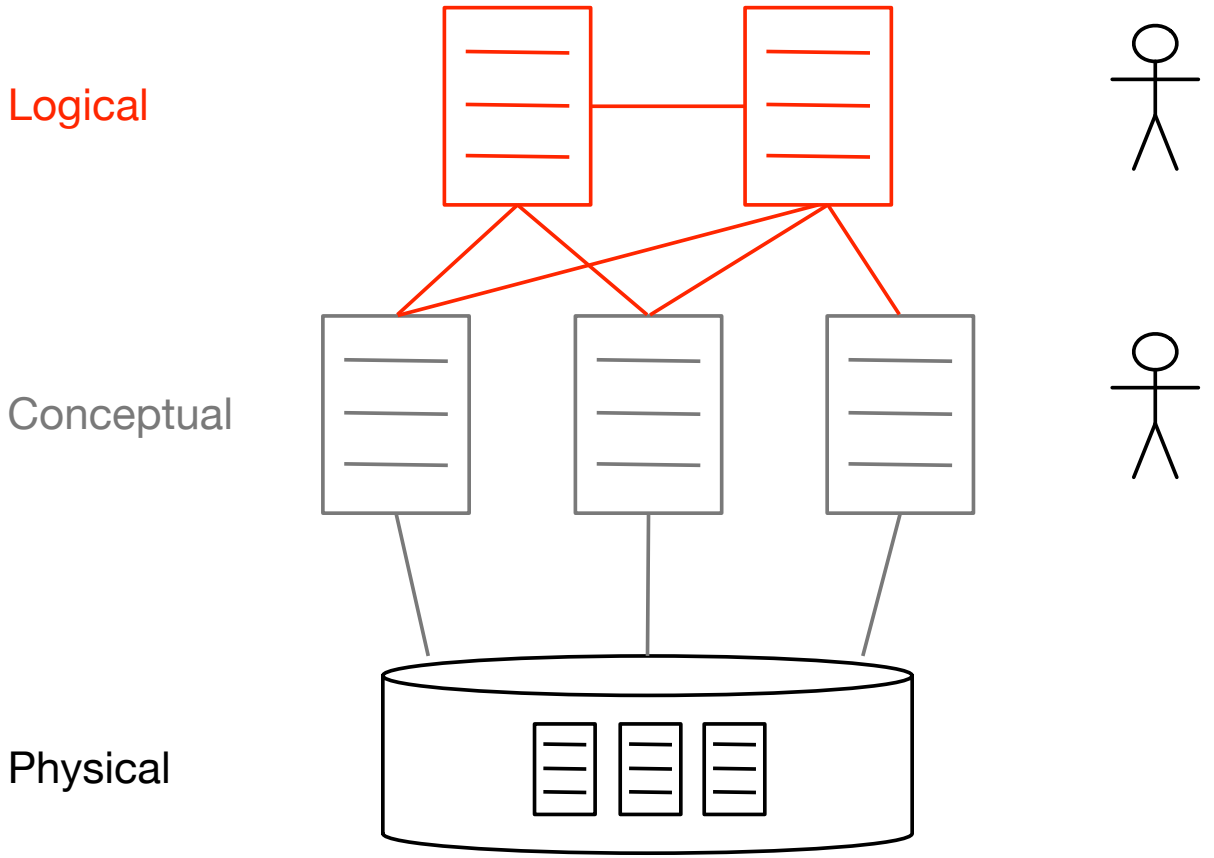
Agenda

Defining and Using Views

Views Modifications Introduction

Views Modification Using Triggers

Three-level vision of database

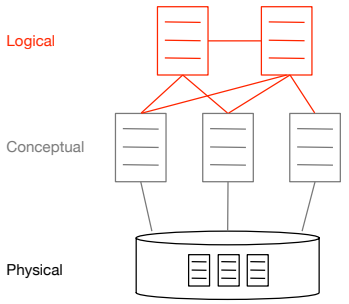


Why use views?

Hide some data from some users

Make some queries easier / more natural

Modularity of database access



Real applications tend to use lots and lots of views

Defining and using views

View V = ViewQuery(R_1, R_2, …, R_n)

Schema of V is schema of query result

Query Q involving V, conceptually:

V := ViewQuery(R_1,R_2,…,R_n)

Evaluate Q

In reality, Q rewritten to use R_1,…,R_n instead of V → Performed automatically by the DBMS

R_i could itself be a view

SQL Syntax

Create View Vname As

<Query>

Create View Vname(A_1,A_2,…,A_n) As

<Query>

Once the view is created, it can be used as if it is a regular table

College Admission Database

Apply

sID	cName	major	dec
123	Stanford	CS	Y
123	Stanford	EE	N
123	Berkeley	CS	Y
123	Cornell	EE	Y
234	Berkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
678	Stanford	history	Y
987	Stanford	CS	Y
987	Berkeley	CS	Y
876	Stanford	CS	Y
876	MIT	biology	Y
876	MIT	marine biology	N
765	Stanford	history	Y
765	Cornell	history	N
765	Cornell	psychology	Y
543	MIT	CS	N

College

cName	state	enr
Stanford	CA	15000
Berkeley	CA	36000
MIT	MA	10000
Cornell	NY	21000

Student

sID	sName	GPA	HS
123	Amy	3.9	1000
234	Bob	3.6	1500
345	Craig	3.5	500
456	Doris	3.9	1000
567	Edward	2.9	2000
678	Fay	3.8	200
789	Gary	3.4	800
987	Helen	3.7	800
876	Irene	3.9	400
765	Jay	2.9	1500
654	Amy	3.9	1000
543	Craig	3.4	2000

View 1

```
Create View CSaccept As
Select sID, cName
From Apply
Where major = 'CS' and dec = 'Y';
```

```
Select *
From CSaccept
```


sID	cName
123	Stanford
123	Berkeley
345	Cornell
987	Stanford
987	Berkeley

View is actually not stored, query is rewritten based on the view definition

A query using View 1

Select Student.sID, sName, GPA

From Student, CSaccept

Where Student.sID = Csaccept.sID and cName = ‘Stanford’ and
GPA < 3.8;

sID	sName	gpa
987	Helen	3.7

What happens when we run a query that refers to a view?

A query using View 1 – conceptual analysis

```
Create temporary table T as
Select sID, cName
From Apply
Where major = 'CS' and dec= 'Y';
```

sID	sName	gpa
987	Helen	3.7

```
Select Student.sID, sName, GPA
From Student, T
Where Student.sID = T.sID and cName = 'Stanford' and GPA < 3.8;

Drop table T;
```

A query using View 1 – practical analysis

```
Select Student.sID, sName, GPA
From Student,
        (Select sID, cName
        From Apply
        Where major = 'CS' and dec= 'Y') as CSaccept
Where Student.sID = CSaccept.sID and cName = 'Stanford' and
        GPA < 3.8;
```

Not how the systems tend to do it either

sID	sName	gpa
987	Helen	3.7

A query using View 1 – practical analysis

Select Student.sID, sName, GPA

From Student, Apply

Where major = ‘CS’ and dec= ‘Y’ and Student.sID = Apply.sID and
cName = ‘Stanford’ and GPA < 3.8;

sID	sName	gpa
987	Helen	3.7

View 2 – a view using other view

```
Create View CSberk As
Select Student.sID, sName, GPA
From Student, CSaccept
Where Student.sID = Csaccept.sID and cName = 'Berkeley' and HS > 500;
```

```
Select *
From CSberk
```


sID	sName	gpa
123	Amy	3.9
987	Helen	3.7

A query using View 2

```
Select *
From CSberk
Where GPA > 3.8;
```

sID	sName	gpa
123	Amy	3.9

What happens when we run a query that refers to a view?

A query using View 2 – rewrite process

```
select * from
(select Student.sID, sName, GPA
from Student, (select sID, cName from Apply
                where major = 'CS' and dec = 'Y') as CSaccept
where Student.sID = Csaccept.sID and cName = 'Berkeley' and HS
> 500) Csberk
where GPA > 3.8;
```

What would be the flattened rewrite of this query?

sID	sName	gpa
123	Amy	3.9

View 2 – dropping View 1

```
drop view CSaccept;
```

What happens?

- Postgres: Error - other objects depend on it
- SQLite/MySQL: no error on drop; error when we refer to CSberk

View 3

```
Create View Mega As

Select College.cName, state, enr, Student.sID, sName, GPA, HS,
       major, dec

From College, Student, Apply

Where College.cName = Apply.cName and Student.sID = Apply.sID;
```

```
Select *
From Mega;
```


cName	state	enr	sID	sName	gpa	HS	major	dec
Cornell	NY	21000	123	Amy	3.9	1000	EE	Y
…	…	…	…	…	…	…	…	…

A query using View 3

```
Select sID, sName, GPA, cName
From Mega
Where GPA > 3.5 and major = 'CS' and enr > 15000;
```

sID	sName	gpa	cName
123	Amy	3.9	Berkeley
987	Helen	3.7	Berkeley

View 3 rewritten

Select Student.sID, sName, GPA, College.cName

From College, Student, Apply

Where College.cName = Apply.cName and Student.sID = Apply.sID
and GPA > 3.5 and major = ‘CS’ and enr > 15000;

sID	sName	gpa	cName
123	Amy	3.9	Berkeley
987	Helen	3.7	Berkeley

Agenda

~~Defining and Using Views~~

Views Modifications Introduction

Views Modification Using Triggers

Modifying views

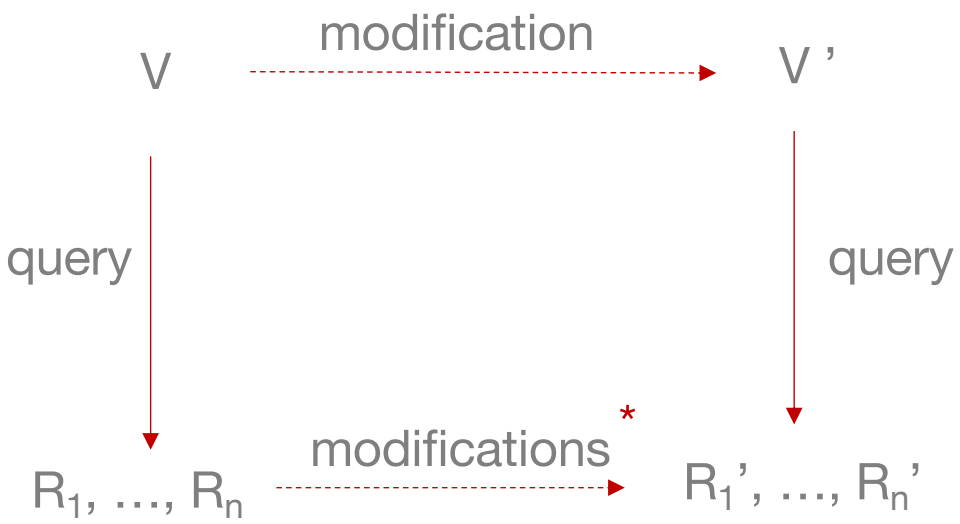
Once view V defined, can we modify V like any table ?

Doesn’t make sense: V is not stored

Has to make sense: views are some users’ entire “view” of the database

Solution: Modifications to V rewritten to modify base tables

Modifying base tables

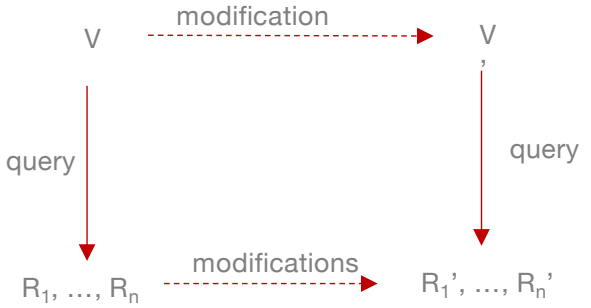


Can we always perform the modifications * so the diagram holds?

Usually, yes

Problems with modifying base tables

Often, there are many possible modifications



R(A,B)

(1,2)

V = $\pi_A(R)$

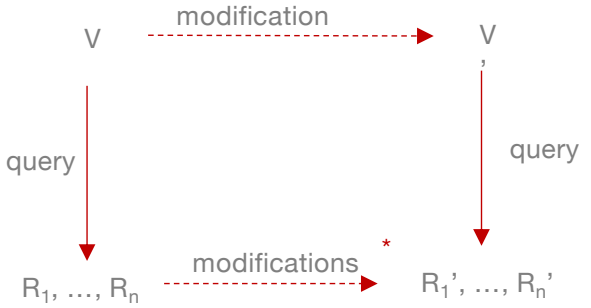
(1)

(3) ← insert

What modifications will be done on R? What do we insert on B?

Problems with modifying base tables

Often, there are many possible modifications



R(N) V = $avg(N)$

(1)

(3) (3) ← update to 7

(5)

What modifications will be done on R? There are many options.

↓

Unlike queries, modifications cannot be automated in general

Modifying views

Rewriting process specified explicitly by view creator

- \+ Can handle all modifications
- – No guarantee of correctness (or meaningful)

Instead-of triggers

Restrict views + modifications so that translation to base table modifications is meaningful and unambiguous

- \+ No user intervention
- – Restrictions are significant

SQL standard

Modifying views according to the SQL standard

To be updatable according to the SQL standard, a view must:

- Have only one table T in its top-level FROM clause
- Not use SELECT DISTINCT in its top-level FROM clause
- Include all attributes from T that do not permit NULLs
- Not refer to T in subqueries
- Not use GROUP BY or aggregation

Kahoot time!

Any doubts?

Readings

Jeffrey Ullman, Jennifer Widom, A first course in Database Systems 3rd Edition

- Section 8.1 – Virtual Views
- Section 8.2 – Modifying Views