

Views (Virtual Tables) in SQL

Database Design

Department of Computer Engineering
Sharif University of Technology

Maryam Ramezani maryam.ramezani@sharif.edu

Introduction

Concept of view



We may frequently issue queries that retrieve the employee name and the project names that the employee works on. Rather than having to specify the join of the three tables EMPLOYEE, WORKS_ON, and PROJECT every time we issue this query, we can define a view that is specified as the result of these joins. Then we can issue queries on the view, which are specified as single table retrievals rather than as retrievals involving two joins on three tables. We call the EMPLOYEE, WORKS_ON, and PROJECT tables the defining tables of the view.

EMPLOYEE Minit Ssn **B**date Address Sex Salarv Fname Super_ssn Dno Lname **PROJECT** DEPARTMENT Pname Pnumber **Plocation** Dnum Dnumber Mgr_start_date Dname Mgr_ssn WORKS ON **DEPT LOCATIONS** Essn Pno Hours Dnumber Dlocation DEPENDENT CE384: Database Design Maryam Ramezani Dependent_name Sex **B**date Relationship Essn

Concept of view



- ☐ A view in SQL terminology is a single table that is derived from other tables.
- These other tables can be base tables or previously defined views.
- A view does not necessarily exist in physical form; it is considered to be a virtual table, in contrast to base tables, whose tuples are always physically stored in the database.
 - This limits the possible update operations that can be applied to views,
 but it does not provide any limitations on querying a view.
- We can think of a view as a way of specifying a table that we need to reference frequently, even though it may not exist physically.
- The view is given a (virtual) table name (or view name), a list of attribute names, and a query to specify the contents of the view.
- ☐ If none of the view attributes results from applying functions or arithmetic operations, we do not have to specify new attribute names for the view, since they would be the same as the names of the attributes of the defining tables in the default case.

Specification of Views in SQL



CREATE VIEW

WORKS_ON1

WORKS_ON1

AS SELECT

Fname, Lname, Pname, Hours

Fname Lname Pname Hours

FROM

WHERE

EMPLOYEE, PROJECT, WORKS_ON

FROM Fname, Lname WORKS ON1

Ssn = Essn AND Pno = Pnumber;

WHERE Pname = 'ProductX';

DROP VIEW

WORKS_ON1;

CREATE VIEW

DEPT_INFO(Dept_name, No_of_emps, Total_sal)

AS SELECT

Dname, COUNT (*), SUM (Salary)

FROM

DEPARTMENT, EMPLOYEE

WHERE

Dnumber = Dno

GROUP BY

Dname;

DEPT_INFO

Dept_name | No_of_emps

Total_sal

Query on View



- SQL query on a view is in the same way we specify queries on base tables:
 - Example: retrieve the last name and first name of all employees who work on the 'ProductX' project
 - QV1:

SELECT Fname, Lname FROM WORKS_ON1 WHERE Pname = 'ProductX'; WORKS_ON1

Fname L	name	Pname	Hours
---------	------	-------	-------

View Properties



Advantages:

- Simplify the specification of certain queries.
- Used as a security and authorization mechanism (End of this slide!)
- A view is supposed to be always up-to-date; if we modify the tuples in the base tables on which the view is defined, the view must automatically reflect these changes. Hence, the view does not have to be realized or materialized at the time of view definition but rather at the time when we specify a query on the view.
- It is the responsibility of the DBMS and not the user to make sure that the view is kept up-to-date.

View Properties



- □ If we do not need a view anymore, we can use the □ DROP VIEW command to dispose of it.
 - Example: DROP VIEW WORKS_ON1

View Implementation in DBMs

View Implementation



- How a DBMS can efficiently implement a view for efficient querying?
 - . It is complex!
 - Two main approaches have been suggested:
 - Query modification
 - View materialization

Query modification (View Computation)



Involves modifying or transforming the view query (submitted by the user) into a query on the underlying base tables.

CREATE VIEW AS SELECT FROM WHERE WORKS_ON1
Fname, Lname, Pname, Hours
EMPLOYEE, PROJECT, WORKS_ON
Ssn = Essn AND Pno = Pnumber;

QV1: SELECT FROM WHERE

Fname, Lname
WORKS_ON1
Pname = 'ProductX';

SELECT

Fname, Lname

FROM

EMPLOYEE, PROJECT, WORKS_ON

WHERE

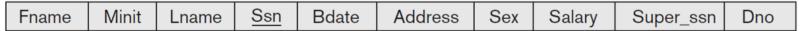
Ssn = Essn AND Pno = Pnumber

AND Pname = 'ProductX';

Query modification (View Computation)



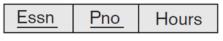
EMPLOYEE



PROJECT







SELECT

FROM

WHERE

Fname, Lname

EMPLOYEE, PROJECT, WORKS_ON

Ssn = Essn AND Pno = Pnumber

AND Pname = 'ProductX';

Disadvantage:

Inefficient for views defined via complex queries that are time-consuming to execute, especially if multiple view queries are going to be applied to the same view within a short period of time

View Implementation



- How a DBMS can efficiently implement a view for efficient querying?
 - . It is complex!
 - Two main approaches have been suggested:
 - Query modification
 - View materialization

View Materialization



Involves physically creating a temporary or permanent view table when the view is first queried or created and keeping that table on the assumption that other queries on the view will follow.

Efficient strategy for automatically updating the view table when the base tables are updated? incremental update have been developed for this purpose, where the DBMS can determine what new tuples must be inserted, deleted, or modified in a materialized view table when a database update is applied to one of the defining base tables.

View Materialization



- The view is generally kept as a materialized (physically stored) table as long as it is being queried.
- ☐ If the view is not queried for a certain period of time, the system may then automatically remove the physical table and recompute it from scratch when future queries reference the view.
- Application: For Data mining goals!

Materialized View Example



```
CREATE MATERIALIZED VIEW user_purchase_summary AS SELECT
    u.id as user_id,
    COUNT(*) as total_purchases,
    SUM(CASE when p.status = 'cancelled' THEN 1 ELSE 0 END) as cancelled_purchases
FROM users u
JOIN purchases p ON p.user_id = u.id;
```

When executed, this statement instructs the database to:

- Execute the SELECT query within the materialized view definition.
- Cache the results in a new "virtual" table named user purchase summary
- Save the original query so it knows how to update the materialized view in the future.

View Materialization DBMS Update Strategies



- Different strategies as to when a materialized view is updated are possible:
 - Immediate update strategy updates a view as soon as the base tables are changed
 - Lazy update strategy updates the view when needed by a view query
 - Periodic update strategy updates the view periodically (in the latter strategy, a view query may get a result that is not up-to-date).

How do materialized views work in specific databases?



Database	Materialized Views?	View Maintenance	Notes
PostgreSQL	Yes, in v9.3+	Manual	Materialized views are populated at time of creation and must be manually refreshed via REFRESH MATERIALIZED VIEW statements that recompute the entire view.
MySQL	No	N/A	
Microsoft SQL Server	Yes	Automatic	SQL Server calls them "Indexed Views" because the materialization step is a matter of creating an index on a regular view. SQL Server limits indexed views to basic SQL queries.
Oracle	Yes	Multiple Options	Materialized views in Oracle databases can be set to manually refresh, refresh on a schedule, or, if the SQL query meets these requirements, automatically refreshed.

User Commands on View

User Command on View



- A user can always issue a retrieval query against any view.
- Issuing an INSERT, DELETE, or UPDATE command on a view table is in many cases not possible.
- An update on a view defined on a single table without any aggregate functions can be mapped to an update on the underlying base table under certain conditions.
- For a view involving joins, an update operation may be mapped to update operations on the underlying base relations in multiple ways. Hence, it is often not possible for the DBMS to determine which of the updates is intended.

Updatable and Non Updatable View

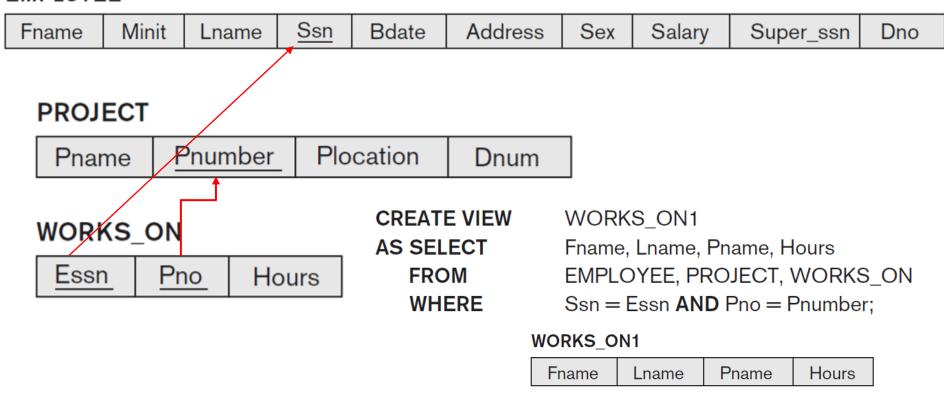


- Updatable: Can be updated but has problems in some cases.
- Non Updatable: Can not convert external to conceptual.
- □ View can be defined on:
 - One base table:
 - Key Preserving: view has primary key of base table: OK with some problems.
 - Non Key Preserving: NO
 - View has aggregated columns: NO
 - More than one base table: In practice non updatable but updatable in theory.

Example



EMPLOYEE





■ Challenge 1: Consider the WORKS_ON1 view, and suppose that we issue the command to update the PNAME attribute of 'John Smith' from 'ProductX' to 'ProductY'.

UV1: UPDATE WORKS_ON1

SET Pname = 'ProductY'

WHERE Lname = 'Smith' AND Fname = 'John'

AND Pname = 'ProductX';



■ Two possible updates on the base relations corresponding to the view update operation in UV1:

```
(a):
      UPDATE WORKS_ON
      SET
                           ( SELECT Pnumber
                 Pno =
                            FROM
                                      PROJECT
                            WHERE
                                      Pname = 'ProductY')
      WHERE
                 Essn IN
                           ( SELECT
                                      Ssn
                            FROM
                                      EMPLOYEE
                                      Lname = 'Smith' AND Fname = 'John')
                            WHERE
                                                                relates 'John Smith' to the
                 AND
                 Pno =
                           ( SELECT
                                      Pnumber
                                                                'ProductY' PROJECT tuple
                            FROM
                                      PROJECT
                                                                instead of the
                            WHERE
                                      Pname = 'ProductX');
                                                                'ProductX' PROJECT tuple
                                                                and is the most likely desired
                                                                update.
                                                                                       24
     CE384: Database Design
                                         Maryam Ramezani
```



- Two possible updates on the base relations corresponding to the view update operation in UV1:
- (b): UPDATE PROJECT SET Pname = 'ProductY'
 WHERE Pname = 'ProductX';

(b) Would also give the desired update effect on the view, but it accomplishes this by changing the name of the 'ProductX' tuple in the PROJECT relation to 'ProductY'. It is quite unlikely that the user who specified the view update UV1 wants the update to be interpreted as in (b), since it also has the side effect of changing all the view tuples with Pname = 'ProductX'.



□ Challenge 2: Some view updates may not make much sense; for example, modifying the Total_sal attribute of the DEPT_INFO view does not make sense because Total_sal is defined to be the sum of the individual employee salaries. This incorrect request is shown as UV2:

V2: CREATE VIEW DEPT_INFO(Dept_name, No_of_emps, Total_sal)

AS SELECT Dname, COUNT (*), SUM (Salary)

FROM DEPARTMENT, EMPLOYEE

WHERE Dnumber = Dno

GROUP BY Dname;

UV2: UPDATE DEPT_INFO

SET Total sal = 100000

WHERE Dname = 'Research';

Notes



■ In SQL, the clause WITH CHECK OPTION should be added at the end of the view definition if a view is to be updated by INSERT, DELETE, or UPDATE statements.

CREATE VIEW V2 [(SN, SJ, SL)]

AS SELECT STID, STJ, STL

FROM STT

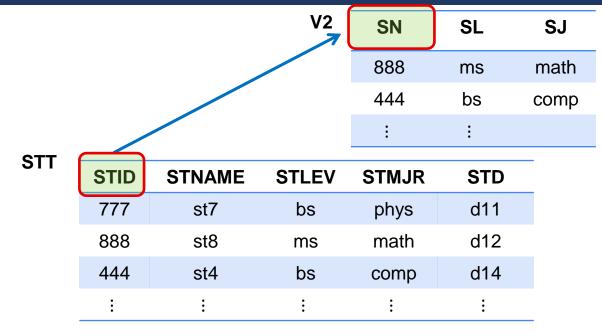
WHERE STJ != 'phys'

[WITH CHECK OPTION]

☐ This allows the system to reject operations that violate the SQL rules for view updates. The full set of SQL rules for when a view may be modified by the user are more complex than the rules stated earlier.

View Example





CREATE VIEW V2 [(SN, SJ, SL)] AS SELECT STID, STMJR, STLEV FROM STT WHERE STMJR != 'phys' [WITH CHECK OPTION] Maryam Ramezani

View - Delete by User



DELETE FROM V2
WHERE SN='444'



DELETE FROM STT

WHERE STID='444' AND STMJR != 'phys'

- Any single row operation is ok! This is a single row because we have STID the primary key in the condition of query!
- The row is deleted, and any other user will not have this row in his view!

View - Delete by User



DELETE FROM V2

WHERE SL='ms'



DELETE FROM STT

WHERE STLEV='ms' AND STMJR != 'phys'

- ☐ This is not a single row, because we don't have primary key in query condition. Many rows in STT may be deleted.
- The request is rejected by the DBMS!

View - Delete by User



CREATE VIEW V3

AS SELECT DISTINCT STNAME, STMJR
FROM STT



DELETE FROM V3

WHERE STNAME='a'



Rejected by the DBMS!

View - Update by User



```
UPDATE √2
SET SJ='IT'
WHERE SN='444'
```

UPDATE STT

SET STMJR='IT' **WHERE** STID='444' ' **AND** STMJR != 'phys'

View - Update by User



```
SET SJ='phys'

WHERE SN='888'
```



UPDATE STT

SET STMJR='phys'

WHERE STID='888' ' AND STMJR != 'phys'



Rejected because of WITH CHECK OPTION

If there was not WITH CHECK OPTION, what happened?

SELECT V2.* FROM V2

No row with 888 key in view! :D

CE384: Database Design

Summarized updating a view by user



- Generally, a view update is feasible when only one possible update on the base relations can accomplish the desired update operation on the view.
- Whenever an update on the view can be mapped to more than one update on the underlying base relations, it is usually not permitted.

Note: Some researchers have suggested that the DBMS have a certain procedure for choosing one of the possible updates as the most likely one. Some researchers have developed methods for choosing the most likely update, whereas other researchers prefer to have the user choose the desired update mapping during view definition. But these options are generally not available in most commercial DBMSs

View – Insert by User



VALUES ('555', 'chem', 'bs')

INSERT INTO STT VALUES ('555', ?, 'chem', 'bs', ?)

- The request is rejected, if the hidden columns has NOT NULL constraint.
- If STID is a primary key and '555' was inserted in STT table before but is hidden in view, then the request is rejected.
- Inserting value VALUES ('555', 'phys', 'bs') is rejected because of [WITH CHECK OPTION].

View with Aggregated Columns



☐ It is non updatable in practice and theory

V 4	PN	SQ
	P1	100
	P2	210
	Р3	80

CREATE VIEW V4 (PN, SQ)

AS SELECT P#, SUM(QTY)

FROM SP

GROUP BY P#

SP

S#	P#	QTY
S1	P1	100
S1	P2	140
S2	P3	80
S2	P2	70



Summarized inserting into view by user



A user can not insert data into view with a single defining table if:

- Having [WITH CHECK OPTION] and conflict with the constraints.
- Violation of the UNIQUENESS constraint of primary key.
- Violation of the NOT NULL constraint

Conclusion in Practical



- □ A view with a single defining table is updatable if:
 - o (1) the view attributes contain the primary key of the base relation
 - (2) All the other attributes of base table do not have the NOT NULL constraint or has the default values
 - (3) View query does not have Distinct in the select.
 - (4) View query does not have group by or having.
- □ Views defined on multiple tables using joins are generally not updatable.
- □ Views defined using grouping and aggregate functions are not updatable.

Theory: Update View of Multiple Table

Multiple Table



- As practically view of multiple base tables is not updatable, but when is it theoretically acceptable?
 - Cartesian Product ×
 - Join on two primary keys PK-PK
 - Join on primary and foreign keys PK-FK
 - Join on two foreign keys which are referenced to primary key of another table FK-FK
 - Join on two non-key columns of two tables without any relationship NK-NK
 - Union
 - Intersect
 - Except

Multiple Table Example



V5

STID	STNAME	STLEV	STMJR	STDEID
777	st7	bs	phys	d11
888	st8	ms	math	d12
444	st4	bs	comp	d14
:	:	:	:	:

CREATE VIEW V5

AS SELECT *
FROM ST1 NATURAL JOIN ST2

ST1

STID	STNAME	STLEV
777	st7	bs
888	st8	ms
444	st4	bs
:	:	:

 STID
 STMJR
 STDEID

 777
 phys
 d11

 888
 math
 d12

 444
 comp
 d14

 :
 :
 :

ST2

Multiple Table PK-PK



```
INSERT INTO V5 VALUES ('999', 'St9', 'chem', 'bs', 'D15')
```

INSERT INTO ST1 **VALUES** ('999', 'St9', 'bs')

INSERT INTO ST2 **VALUES** ('999', 'chem', 'D15')

Multiple Table PK-FK



STT

STID	STNAME	STLEV	STMJR	STD
777	st7	bs	phys	d11
888	st8	ms	math	d12
444	st4	bs	comp	d14
:	:	:	:	:

STCOT

STID	COURSE	GRADE
777	40280	19
888	40567	20
444	40232	15
:	:	:

Multiple Table PK-FK



CREATE VIEW V6

AS SELECT STT.STID, STT.NAME, STCOT.COURSE,STCOT.GRADE FROM STT JOIN STCOT

INSERT INTO V6

VALUES ('9212345', 'Amir', '40638', 15)

INSERT INTO STT

VALUES ('9212345', 'Amir', ?, ?, ?)

INSERT INTO STCOT **VALUES** ('9212345', '40638', 15)

*Insert one row in STCOT (table having foreign key!)

*If the student is not in table STT then it will be inserted.

Multiple Table PK-FK



CREATE VIEW V6

AS SELECT STT.STID, STT.NAME, STCOT.COURSE, STCOT.GRADE FROM STT JOIN STCOT

Delete from V6

where STID='9212345' and NAME= 'Amir' and STCOT= '40638' and GRADE=15)

Delete from STCOT
What about STT? Affect on other tables and view?

Union



Create view v7 as select * from T1 union select * from T2

- Insert row: into both or one of T1 or T2.
- Delete row: delete from both T1 and T2.
- □ Update row: update both in T1 and T2.

Intersect



Create view v8 as select * from T1 intersect select * from T2

- Insert row: insert into both T1 and T2.
- Delete row: delete from T1 or T2.
- ☐ Update row: update both in T1 and T2.

Except



Create view v9 as select * from T1 except select * from T2

- Insert row: insert into T1 and check not be in T2.
- Delete row: delete from T1
- □ Update row: update in T1 and check not be in T2.

In-line View

Inline View



- It is also possible to define a view table in the FROM clause of an SQL query. This is known as an in-line view. In this case, the view is defined within the query itself.
- □ Inline views refer to a SELECT statement located in the FROM clause of secondary SELECT statement. Inline views can help make complex queries simpler by removing compound calculations or eliminating join operations while condensing several separate queries into a single simplified query.

Inline View



- PostgreSQL semantics may refer to inline views as Subselect or as Subquery
- ERROR: subquery in FROM must have an alias. This is because in PostgreSQL the use of aliases is mandatory. The following example uses B as an alias.

```
SELECT A.LAST_NAME, A.SALARY, A.DEPARTMENT_ID, B.SAL_AVG
FROM EMPLOYEES A,

(SELECT DEPARTMENT_ID, ROUND(AVG(SALARY)) AS SAL_AVG
FROM EMPLOYEES GROUP BY DEPARTMENT_ID) B

WHERE A.DEPARTMENT_ID = B.DEPARTMENT_ID;
```

Authorization Mechanisms with View

Views as Authorization Mechanisms



- Creating an appropriate view and granting certain users access to the view and not the base tables, they would be restricted to retrieving only the data specified in the view.
 - Access to specific rows

CREATE VIEW DEPT5EMP AS

SELECT *

FROM EMPLOYEE

WHERE Dno = 5;

Access to specific columns

CREATE VIEW BASIC_EMP_DATA AS

SELECT Fname, Lname, Address

FROM EMPLOYEE;

Conclusion



Advantages:

- Simplify the specification of certain queries.
- Used as a security and authorization mechanism (for columns & rows)
 - Data sharing
 - Hidden Data
- Data Independence (Next Topic!)
- Good for data mining and reporting

Conclusion



Disadvantages:

- Not good when system is single user!
- Not good when user needs to run updatable commands.
- Overhead for converting query to base table queries