9 Views in SQL: a concept for restricting column access and for simplifying application programming

Kemper / Eickler: chap.4.16-18, ; Elmasri: chap. 8.3,
Melton: chap. 4.3 – 4.4,

9.1 Views

9.1.1 General idea of db views

- tailoring the database schema for different applications
- Access protection
- Privacy
- Structuring of SQL programs

– The RDM concept for external schemas ("3-schema-architecture")

– Nothing but a named SQL-command, which becomes part of the schema

```
CREATE VIEW CustAndMovies  -- customer and their lent movies
AS
  SELECT c.name AS customer, m.title
  FROM Customer c NATURAL JOIN  Rental r
       JOIN Tape t ON (r.Tape_Id = t.t_id)
       NATURAL JOIN Movie m
  WHERE AND m.cat = 'Horror';
```

SQL Views

Views are virtual tables, i.e. not materialized

– May be defined on base tables or views (or both)

```
CREATE VIEW  GenreStatistics
AS
  SELECT   cm.cat, COUNT (cm.title) AS count
  FROM     CustAndMovies cm
  GROUP by cm.cat
```

– Access to view may be allowed even if access to defining relations (base tables, other views) is restricted (privacy!)

– May be queried like base tables

Materialized view:	snapshot of the data
View:	SQL expression

9.1.2 Query execution on views

Two steps:

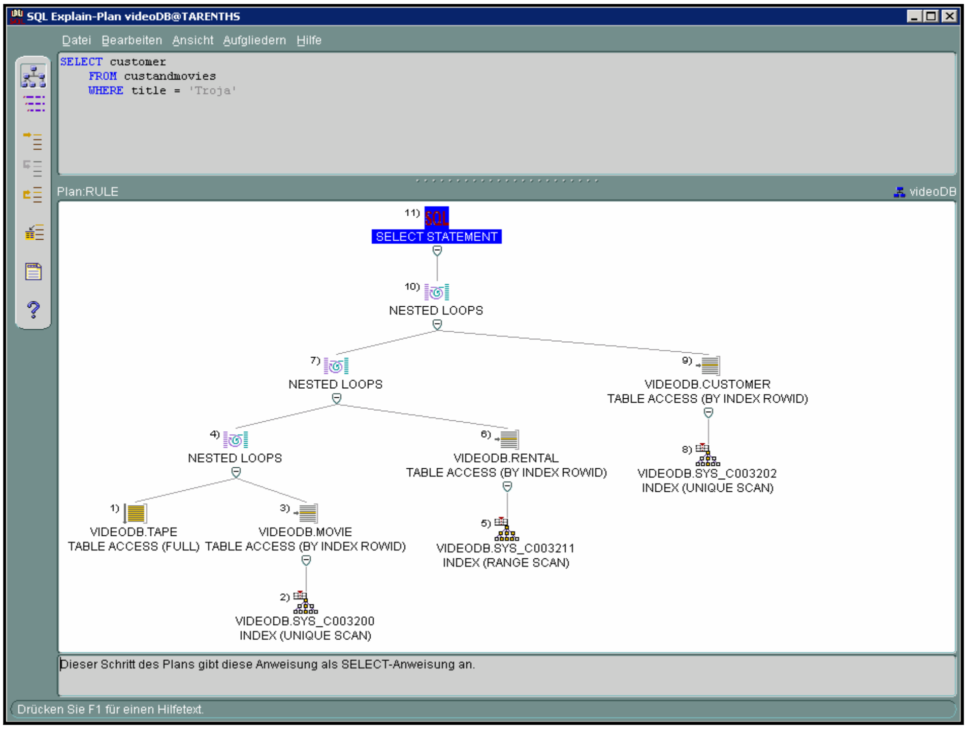
1. Transform query on V using the definiton of V
2. Execute query on base tables only

```
      SELECT  customer  FROM  CustAndMovies
      WHERE from <= '2002-01-01'
```

substitution

```
SELECT c.name
FROM Customer c NATURAL JOIN  Rental r
      JOIN Tape t ON (r.Tape_Id = t.t_id)
      NATURAL JOIN Movie m
 WHERE
     m.cat = 'Horror';
     AND r.from_date <= '2002-01-01'
```

Most systems use query tree expansion instead of SQL substitution


SQL Explain-Plan videoDB@TARENTHS
Datei Bearbeiten Ansicht Aufgliedern Hilfe
Plan:RULE
videoDB

```
SELECT customer
    FROM custandmovies
    WHERE title = 'Troja'
```

```
11) SELECT STATEMENT
10) NESTED LOOPS
7) NESTED LOOPS
9) VIDEODB.CUSTOMER
   TABLE ACCESS (BY INDEX ROWID)
4) NESTED LOOPS
6) VIDEODB.RENTAL
   TABLE ACCESS (BY INDEX ROWID)
8) VIDEODB.SYS_C003202
   INDEX (UNIQUE SCAN)
1) VIDEODB.TAPE
   TABLE ACCESS (FULL)
3) VIDEODB.MOVIE
   TABLE ACCESS (BY INDEX ROWID)
5) VIDEODB.SYS_C003211
   INDEX (RANGE SCAN)
2) VIDEODB.SYS_C003200
   INDEX (UNIQUE SCAN)
```

Dieser Schritt des Plans gibt diese Anweisung als SELECT-Anweisung an.

Drücken Sie F1 für einen Hilfetext.

Views in Postgres

- More general substitution concept in Posgres
- Rules are "first class objects": CREATE RULE...

```
CREATE VIEW myview AS SELECT * FROM mytab;
```

equivalent to

```
CREATE TABLE myview (<same column list as
   mytab>);
CREATE RULE "_RETURN" AS ON SELECT TO myview
   DO INSTEAD SELECT * FROM mytab;
```

- Kind of dynamic view evaluation compared to static rewrite of query or query tree

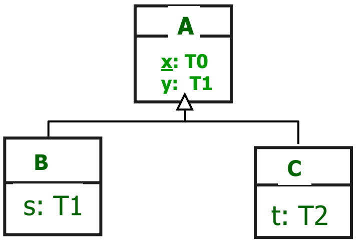
9.1.3 Generalization hierachies and views

1. "Big table" approach:
   specialize

```
    A(x, y, s, t)

Create VIEW B
AS
SELECT x,y,s FROM A
WHERE NOT IS NULL s
```

In the same way for
C and A \ B \ C

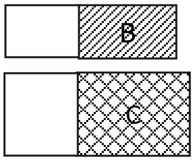


Generalization and views

2. Separate table AB, AC for
   B and C objects

```
        AB(x,y,s)
        AC(x,y,t)

Create VIEW  A
AS
(SELECT x  FROM AB)
 UNION
(SELECT x FROM AC)
```



(Third case -> assignments)

9.2 Updatable views

View updates

– Many views are not updatable, some obvious cases:

```
CREATE VIEW M_F (movie, numFormats)
 AS SELECT  m_Id, COUNT( distinct format),
    FROM    Tape
    GROUP BY m_Id HAVING COUNT(format) > 1
```

Virtual table:

7	2
8	1

```
UPDATE TABLE M_F  set
numFormats=3 WHERE ...
```

??

```
CREATE VIEW TapeOnLoan  -- tapes on loan
 AS SELECT  tape_Id, from_date
      FROM   Rental
      WHERE  until_Date IS NULL ;
```

Virtual table:

1	3.5.00
4	7.5.00

```
insert into TapeOnLoan
       values (5,10.5.00)
```

which customer??

Updatable query expressions

Updatability: an issue for

- views
- for tables defined by table expressions

```
UPDATE (Select customer
        FROM phones
        WHERE customer = 11)
 SET CUSTOMER= 7;
```

Key question: which virtual tables can be updated?

Problem statement: updatable views

For an update u of a view V find one or more updates cu on the underlying base tables D

Example:

```
  CREATE VIEW SB_Movies AS
    SELECT m_id, title, year, director
    FROM Movie
    WHERE director = 'Spielberg'
INSERT INTO SB_Movie
   VALUES (53,'ET', 1987, 'Spielberg')
```

induces the insert-statement

```
INSERT INTO SB_Movie
 VALUES (53,'ET',NULL, 1987, 'Spielberg',NULL,NULL)
```

Note: without the `m_id` or the `director` attribute `SB_Movies` should not be updatable.

9.2.1 Semantic characterization of updatable views

Update of the view (as if it were materialized) must result in the same relation as updating the base tables using one or more updates c_u and applying the view definition subsequently

$$u(V(D)) \overset{!}{=} V(c_u(D))$$

c_u denotes „translation“ of view update in update(s) on the base relations.

Semantic characterization,
Wanted: syntactic criteria for updatability

Updatability conditions

Some plausible conditions:

1. if u does not have an effect, then c_u should not

```
CREATE VIEW T_F (movie, format)
 AS SELECT DISTINCT  mId, format
    FROM Tape WHERE mID > 100;
```

```
INSERT INTO T_F VALUES (m_id=47, format = 'VHS')
```

c_u is:

```
INSERT INTO Tape (m_Id, format) VALUES ....
```

Expression has no effect on view
but on base table: bad

Updatability conditions

2. c_u should only effect tuples in D which are represented in V(D) ("no side effects")

Virtual table:

7	2
10	3
11	1,5

```
CREATE VIEW M_P (movie, price)
 AS SELECT  mId, price_Day  FROM Movie
    WHERE price_Day > 1
```

```
UPDATE TABLE M_P  SET  price =
(price + 1) WHERE  price_Day < 2
```

u has an effect on tuples of view, (expl: "1 < x < 2")
but more may be affected in base table
(expl: " 0 < price_Day <= " for some rows)

Updatability conditions

3. For a view update u there should be an inverse update w such that

   w (u (V(D))) = V (D)

   example?

4. No constraint on base tables must be violated by u (e.g. a NOT NULL restriction)

```
CREATE VIEW M_PTitle (movieTitle, price)
 AS SELECT  title, price_Day  FROM Movie;

INSERT INTO M_Ptitle VALUES('To be or not to be', 2.0);
```

Causes a NULL value in primary key column

1. – 4. are not independent.

9.2.2 Syntactic criteria

- Views for reading only may be arbitrarily defined
- Update is rejected, if view not updatable

- Syntactic criteria

  Not updatable (SQL 92)
    - if grouped (GROUP BY), HAVING or aggregated
    - DISTINCT in SELECT clause
    - set operators (INTERSECT, EXCEPT, UNION)
    - more than one table in from clause

    - No updates on join views (restrictive!)

Syntactic criteria (2)

- SQL 1999

Columns (of views) are potentially updatable if ...

- no DISTINCT operator
- no GROUP BY, HAVING clause
- no derived columns (e.g. arithmetic expressions)

(1) Column is updatable if potentially updatable and one table in from clause (!)

```
CREATE VIEW Dir (d,mplus) AS
SELECT director, m_id+1
FROM Movie
WHERE director >= 'L'
```

Find updatable columns

Find updatable columns by querying the catalogue

```
SELECT column_name, updatable
FROM user_updatable_columns
WHERE table_name = 'DIR'   -- Oracle
```

must be upper case

```
COLUMN_NAME   UPD
------------- ---
D             YES
MPLUS         NO
```

9.2.3 Key preserved tables

… SQL 1999: more than one table in FROM clause

(2) Column c is updatable if potentially updatable and

- if c belongs to exactly one table
- the key of the table is preserved, i.e. the update of c may be traced back to exactly one row.

```
CREATE view T_M
AS SELECT m.m_Id AS mid, t_id, title
FROM movie m, tape t
WHERE m.m_Id = t.m_Id;
```

```
COLUMN_NAME    UPD
-------------  ---
MID            NO
T_ID           YES
TITLE          NO
```

Key preserved tables

Table is key preserved if every key of the table can also be a key of the join result table.
A key-preserved table has its keys preserved through a join.

```
CREATE view T_M
AS SELECT m.m_Id AS mid, t_id, title
     FROM movie m, tape t
     WHERE m.m_Id = t.m_Id;
```

t_id is key preserved, m_Id is not, since mId is not a key in t_m

Views updatable…. sometimes:

a) Only update on ONE base table:
`INSERT INTO T_M (t_id, mid) values (106, 1)`

b) Restrictions on values must not be violated, like NOT   NULL, foreign key etc.

9.2.4 Views WITH CHECK OPTION

Issue: side effects on base table rows, no effect on view

```
CREATE VIEW M_P (movie,title, price)
 AS SELECT  m_Id,title, price_Day
     FROM  Movie
     WHERE price_Day >= 1
     and price_Day <=2
   WITH CHECK OPTION
```

3	"To be or not to Be"	2.00
4	"Marnie"	1.00
7	"The Kid"	1.00

```
UPDATE TABLE M_P  SET  price =
(price + 1) WHERE  price <= 2
```

- Update may result in insertion andd deletion (!) of rows
- CHECK OPTION: update and insert must result in rows the view can select , otherwise exception raised

9.3 View update by triggers

Triggers: Event – Condition – Action rules

Event: **`Update, insert, delete`** (basically)

Condition: **`WHEN`** < some conditon on table>

Action: some operation (expressed as DML, DB-Script language expression, even Java)

INSTEAD OF Triggers

- defined on views
- specify what to do in case of an update of the view

Summary

- Views: important mechanism for
  - access protection / privacy
  - making application SQL programming on DB simpler
- The mechanism for defining external schemas in the RDM
- Useful for modeling generalization hierarchies
- Disadvantage: updates (inserts, deletes) not always possible
- Criteria for updatable views complex
- INSTEAD OF triggers are a convenient work around