Part 12: Views (Virtual Tables)

References:

- Elmasri/Navathe:Fundamentals of Database Systems, 3rd Edition, 1999.
  Section. 8.5, "Views (Virtual Tables) in SQL"
- Silberschatz/Korth/Sudarshan: Database System Concepts, 3rd Edition, McGraw-Hill, 1999.
  Section 4.8, "Views" (plus 4.9.4, "Update of a View")
- Kemper/Eickler: Datenbanksysteme (in German), Ch. 4, Oldenbourg, 1997.
- Lipeck: Skript zur Vorlesung Datenbanksysteme (in German), Univ. Hannover, 1996.
- Date/Darwen: A Guide to the SQL Standard, Fourth Edition, Addison-Wesley, 1997.
- Oracle8 SQL Reference, Oracle Corporation, 1997, Part No. A58225-01.
- Oracle8 Concepts, Release 8.0, Oracle Corporation, 1997, Part No. A58227-01.
- Don Chamberlin: A Complete Guide to DB2 Universal Database. Morgan Kaufmann, 1998.
- Microsoft SQL Server Books Online: Accessing and Changing Data, Administering SQL Server.

Objectives

After completing this chapter, you should be able to:

- explain the concept of a view.

  E.g. what is the difference between declaring a view and storing the corresponding query result as a new table?
  Why are views called "virtual tables"?

- enumerate some possible applications for views.
- declare views in SQL.

  You should also know when to use "`WITH CHECK OPTION`".

- explain how view updates are done, why not all views are updatable, and which views are.

Overview

1. Concept, Using Views in Queries
2. View Updates
3. Views and Security

Example Database

COURSES

CRN	TOPIC	TITLE	INSTRUCTOR
22332	2710	Databases	Brass
31864	2550	Client-Server	Spring
41590	2711	Data Structures	Brass

INSTRUCTORS

NAME	OFFICE	PHONE	EMAIL
Brass	724	9404	sbrass@sis.pitt.edu
Spring	727	9429	spring@...

Views (1)

- Views make it possible to store a query in the database and give it a name (optionally, the result columns can be renamed):

```
CREATE VIEW MY_COURSES(NO, TITLE) AS
SELECT CRN, TITLE
FROM   COURSES
WHERE  NAME = 'Brass'
```

- The result of an SQL query is a table.
- Views ("virtual tables") can be used in queries like tables that are actually stored ("base tables").

Views (2)

- The view can be used e.g. as follows:

```
SELECT *
FROM   MY_COURSES
WHERE  TITLE LIKE 'Database%'
```

- The DBMS treats the view name simply as an abbreviation (alias) for the corresponding query:

```
SELECT *
FROM   (SELECT CRN NO, TITLE
        FROM   COURSES
        WHERE  NAME = 'Brass') MY_COURSES
WHERE  TITLE LIKE 'Database%'
```


Views (3)

- Views were already contained in the SQL-86 standard, but with certain restrictions.
- The problem was that SQL-86 did not permit subqueries under FROM.
- Therefore it was not easy to translate the given query into one that refers only to the base tables:

```
SELECT CRN NO, TITLE
FROM   COURSES
WHERE  TITLE LIKE 'Database%'
AND    NAME = 'Brass'
```


Views (4)

- E.g. in some systems, it was not possible to use an aggregation in a query to a view that itself is computed by an aggregation.

  This would need a nested aggregation which SQL-86 does not permit (it needs a subquery under FROM in SQL-92).

- In SQL-92 and in the three DBMS (Oracle, DB2, SQL Server) there are no restrictions on the use of views in queries.

  Except fulltext queries in SQL Server.

Views (5)

- Views are derived, virtual tables which are computed from the (actually stored) base tables.
- Views can never contain information not present in the base tables.

  One can regard the view definition itself as additional information.
- Views can only present the information in the base tables in more convenient ways.
- The DBMS translates queries that refer to views (and possibly base tables) into equivalent queries that refer only to base tables.

Views (6)

- Note the difference of the `CREATE VIEW` to

```
CREATE TABLE MY_COURSES2(...);
INSERT INTO MY_COURSES2(NO, TITLE)
       SELECT CRN, TITLE FROM COURSES
       WHERE NAME = 'Brass'
```

- This evaluates the defining query only once and stores the result permanently in a new relation.
- In contrast, in a real view, the defining query is always evaluated when the view is used.

  The query optimizer will try to compute only the part of the view that is needed for the current query.

Views (7)

- If the base table (`COURSES`) changes,
  - ◇ the view will automatically reflect the change,

    It is anyway recomputed whenever it is accessed. Only the defining query is permanently stored in the system.
  - ◇ whereas the above table (`MY_COURSES2`) needs to be updated manually (or by means of a trigger).

    A trigger is a procedure that is stored in the database and executed when certain events happen. In this case, e.g. when a row is inserted into `COURSES`, it must be checked whether `NAME = 'Brass'`, and if yes, the course data must also be inserted into `MY_COURSES2`.
- Some systems (experimental deductive DBs) can automatically manage "materialized views".

Views (8)

- It is possible to define views which give different results depending on the user who queries them:

```
CREATE VIEW MY_COURSES(NO, TITLE) AS
SELECT CRN, TITLE FROM COURSES
WHERE  NAME = USER
```

- "`USER`" returns the login name of the current user.

  For the view to work, `NAME` in `COURSES` must be DB login names.
- E.g. the Oracle Data Dictionary uses this.

  E.g. the system "table" (view) `CAT` shows every user his/her tables.
- Views can depend also on the current date.

Views (9)

- Views can be used in the definitions of other views.
- In this way, complex queries can be built step by step.
- However, recursive views are excluded.

  I.e. it is impossible to use a view directly or indirectly in its own definition.
- Deductive databases support also recursive views.

  In deductive databases, a Prolog-like language ("Datalog") is used for queries and programs. Derived predicates (i.e. views, procedures) can be defined by sets of logical rules ("Horn clauses", if-then-rules).

Outlook: Recursion (1)

- Recursion is useful for tree-structured data (hierarchies), and e.g. "transitive closure" queries:
  - ◇ E.g. in the row for employee A, the table EMP contains the direct supervisor B of A (MGR).
  - ◇ In the same way, B might have the direct supervisor C, and C's boss might be D.
  - ◇ C and D are indirect supervisors of A.
  - ◇ In order to compute all indirect supervisors, recursion is needed.

    Unless one knows the maximal number n of hierarchy levels and n is small.

Outlook: Recursion (2)

- Another example is the "bill of materials" query:
  - ◇ The database contains information about parts that the company assembles e.g. to bicycles.
  - ◇ Elementary parts are bought from other companies. The prices of elementary parts is known.
  - ◇ However, parts can also be built from smaller parts by the company itself. In this case, it is known which smaller parts are needed and how much time is needed for the assembly.
  - ◇ Now, what is the cost of a bicycle?

Outlook: Recursion (3)

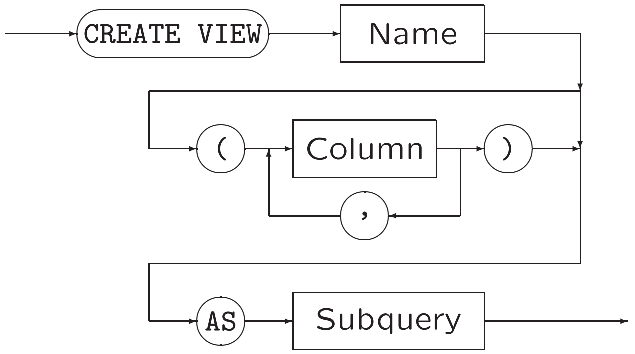
- SQL-99 will support limited forms of recursion:

```
WITH
    RECURSIVE INDIRECT_MGR(EMPNO, BOSS) AS
        SELECT EMPNO, MGR FROM EMP
        UNION
        SELECT A.EMPNO, B.BOSS
        FROM   EMP A, INDIRECT_MGR B
        WHERE  A.MGR = B.EMPNO
SELECT BOSS FROM INDIRECT_MGR
WHERE  EMPNO = 7369
```

- Actually, DB2 has already recursive views.

  But prints a warning that the view can result in an infinite loop.

Syntax (1)



Syntax (2)

- `ORDER BY` is not permitted in view definitions.

  It is permitted only at the very end of a query, but views define subqueries that will be part of a larger query. However, Oracle permits `ORDER BY` in view definitions.

- View definitions can be deleted with:

  `DROP VIEW ⟨NAME⟩`

- In Oracle, one can write

  `CREATE OR REPLACE VIEW ⟨NAME⟩ ...`

  This will redefine the view if it already exists.

Advantages of Views (1)

- Convenience / Reuseability: Repeating patterns in queries are already predefined.
- The base relations should be in BCNF, but it might be more convenient to use non-normalized relations in queries.

  Redundant data in views is no problem because view data are not stored (they are computed whenever the view is needed).

- Adaption of the database schema to the wishes of different users / user groups.

Advantages of Views (2)

- Security: Some users should see only part of a table, or only aggregated/anonymous information.

  Without views, the granularity for access rights is the table.

- Logical Data Independence: New attributes can be added to a table, but the old version can still be supported as a view.

Overview

View Update Problem (1)

- Since the views are not explicitly stored, update requests on the view must be translated in updates on the base tables.
- This is not always possible:

```
CREATE VIEW MY_COURSES3(NAME, TITLE) AS
SELECT NAME, TITLE
FROM   COURSES
WHERE  NAME = 'Brass'
```

- Now consider the command:

```
INSERT INTO MY_COURSES3
VALUES ('Spring', 'Document Processing');
```


View Update Problem (2)

- Even if the row were inserted into the base table, it would never appear in the view.
- A minimal requirement for a correct translation on the base tables is that the view is changed as if it were a stored table.
- Furthermore, in this example the `INSERT` does not specify a CRN. But the attribute `CRN` is the key of the table `COURSES`, so it is not null, and we would have to invent a value.

View Update Problem (3)

- Not all update requests can be translated on the base tables.
- In addition, even if there is such a translation, it can be ambigious, i.e. there might be more than one possible update on the base tables that would cause the requested update on the view.
- An example for an ambigious update request is given on the next slide.

View Update Problem (4)

- Here the view is defined by a join, but the join attribute is projected out:

```
CREATE VIEW COURSE_CONTACT(CRN, TITLE, PHONE) AS
SELECT CRN, TITLE, PHONE
FROM   COURSES C, INSTRUCTORS I
WHERE  C.NAME = I.NAME
```

COURSES		
CRN	TITLE	NAME
22268	Databases	Brass
31822	Client-Server	Spring

INSTRUCTORS	
NAME	PHONE
Brass	9404
Spring	9429

View Update Problem (5)

- For the given base tables, the view looks like this:

COURSE_CONTACT		
CRN	TITLE	PHONE
22268	Databases	9404
31822	Client-Server	9429

- Now consider the following update request:

```
UPDATE COURSE_CONTACT
SET    PHONE = 9429
WHERE  TITLE = 'Databases'
```


View Update Problem (6)

- Did Michael Spring take over the database course?

```
UPDATE COURSES
SET    NAME = 'Spring'
WHERE  TITLE = 'Databases'
```

- Or was Stefan Brass temporarily moved into Michael Spring's office, so that they now share the phone?

```
UPDATE INSTRUCTOR
SET    PHONE = 9429
WHERE  NAME = 'Brass'
```

- The system has no way to decide this.

View Update Problem (7)

- Therefore, most views are read-only and cannot be updated.
- However, there are exceptions, the most important being views that only define a subset of the rows and columns of a single table.

  The columns that are missing in the view must permit null values.

- Limited forms of joins might also be possible, but it must be clear which base table should be updated.

View Update Problem (8)

- The updatability is very important for most applications of views, e.g. for security or logical data independence.
- In these cases, users have only access to the views, and should be able to use them like normal tables, including updates.
- There has been a lot of research on the view update problem.

Translation of Updates (1)

- Consider a deletion from the view MY_COURSES:

```
CREATE VIEW MY_COURSES(NO, TITLE) AS
SELECT CRN, TITLE FROM COURSES
WHERE  NAME = 'Brass'
```

```
DELETE FROM MY_COURSES
WHERE  TITLE LIKE '%Database%'
```

- The deletion is translated into:

```
DELETE FROM COURSES
WHERE  TITLE LIKE '%Database%'
AND    NAME = 'Brass'
```


Translation of Updates (2)

- In the same way, an UPDATE-request would be restricted to those rows in the base table which satisfy the selection condition in the view definition.

  Of course, only the columns CRN and TITLE of COURSES can be updated via the view (NAME does not appear in it).

- The translation of insertions is a bit problematic:

```
INSERT INTO MY_COURSES(NO, TITLE)
VALUES (42232, 'DB Analysis and Design')
```

  This is translated into:

```
INSERT INTO COURSES(CRN, TITLE)
VALUES (42232, 'DB Analysis and Design')
```


Translation of Updates (3)

- The column NAME, which is not contained in the view, will be set to the default value of the column (e.g. null).
- SQL does not analyze the WHERE-clause of the view definition, which would show that NAME must be 'Brass'.
- However, it is possible to declare USER as a default value for the column NAME in the table COURSES.

Translation of Updates (4)

- This would put the current database user into the column `NAME` if no other value is specified in the `INSERT` command.

  And the view user is unable to specify a value.

- This would fit well with the condition "`NAME = USER`" in the view definition (see above).

Updatable Views (1)

Restrictions in SQL-92:

- The `FROM`-clause contains only a single table.
- The `SELECT`-clause contains no `DISTINCT` and no datatype operations (like `+`).
- No aggregations are done.
- Subqueries in the `WHERE`-clause may not refer to the main base table of this view in `FROM`-clauses.
- No `UNION` etc. is allowed.

  The query must be a a single `SELECT`-expression.

Updatable Views (2)

SQL-92, continued:

- Even if the view is updatable in the above sense, the translation of an update might violate integrity constrains such as `NOT NULL`.

Updatable Views (3)

DB2:

- DB2 distinguishes between deletable views, updatable columns in views, and insertable views.
- DB2 basically has the same requirements for deletable views as SQL-92 for updatable views.

  `UNION ALL` and data type operations are allowed.

- A column is updatable if the view is deletable and that column does not contain data type operations.
- A view is insertable if all its columns are updatable and no `UNION ALL` is used.

Updatable Views (4)

Oracle:

- Oracle also allows certain updates on views defined by joins.
- In addition, the result of view updates can be defined procedurally with "INSTEAD OF" triggers.
- If there is one base table, the key of which is also the key of the view, then updates on the view are translated into updates on this "key preserved table".

Updatable Views (5)

Oracle, continued:

- Example for a view that is updatable in Oracle:

```
CREATE VIEW COURSE_INF(CRN, TITLE, NAME, PHONE)
AS SELECT CRN, TITLE, C.NAME, PHONE
   FROM   COURSES C, INSTRUCTORS I
   WHERE  C.NAME = I.NAME
```

- Columns CRN and TITLE are updatable. Deletions from COURSE_INF are translated into deletions from COURSES. Insertions cannot specify a value for PHONE.
- The view is not updatable in SQL-92 or DB2.

Updatable Views (6)

SQL Server:

- SQL Server seems to be quite generous with updatability, but the manual says very little about this problem.

Overview

Check Option (1)

- Suppose that a lower level personnel manager is only allowed to work with employees up to $4000 monthly income.
- The following view is created for him:

```
CREATE VIEW RESTRICTED_EMP AS
SELECT EMPNO, ENAME, SAL
FROM   EMP
WHERE  SAL <= 4000
```

- But this does not prevent the following insertion:

```
INSERT INTO RESTRICTED_EMP
VALUES (1234, 'Brass', 1000000)
```


Check Option (2)

- Of course, the inserted tuple will not be visible in the view, and it cannot be deleted or modified afterwards via the view.
- But it will be stored in the base table. The translation is:

```
INSERT INTO EMP(EMPNO, ENAME, SAL)
VALUES (1234, 'Brass', 1000000)
```

- In the same way, UPDATEs can lead to tuples which violate the view condition.

  The update then becomes a deletion from the view, and a security violation for the base table.

Check Option (3)

- This can be avoided by adding "WITH CHECK OPTION" to the view definition:

```
CREATE VIEW RESTRICTED_EMP AS
SELECT EMPNO, ENAME, SAL FROM EMP
WHERE  SAL <= 4000
WITH   CHECK OPTION
```

- Then insertions/updates will be rejected if lead to a violation of the WHERE-condition in the view.

  The "CHECK OPTION" can only be omitted when the attributes used in the WHERE condition of the view are projected out (and when the column default is set up properly).

Views and Security (1)

- Views can be used like tables in the GRANT command.
- In order to use a view, one needs the usual privileges for it.
- However, one does not need rights on the base tables. View accesses are evaluated with the rights of the view owner, not those of the current database user.

  Because of this, views allow to "encapsulate" the data in the base tables: If a user has only access rights for the view, but not for the base table, he/she can access the base table only via the view.

Views and Security (2)

- The user who defines a view needs to have at least `SELECT` rights on the base tables. If he/she later loses these rights due to a `REVOKE`, the view becomes inaccessible.
- If the owner of the view wants to pass rights on the view to other users, he/she must have the corresponding rights on the base tables with grant option (in Oracle and DB2).

Views and Security (3)

- It seems that in SQL Server, one can only grant rights on the view, if one also owns the underlying base tables.

  Otherwise the grant is still successful, but base table accesses via the view are evaluated with the rights of the current user, not with the rights of the view owner.

- In DB2, one gets the control privilege on the view only if one also has the control privilege on the base tables.