# Part 13: Views

Elmasri/Navathe:

- Section. 8.5, "Views (Virtual Tables) in SQL"

I have also used the following sources:

- 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

- You should be able to explain the concept of a view.
  E.g. you should know the difference to storing the result of a query as a new table.
  You should know possible applications for views.
- You should know how to declare views in SQL (including "WITH CHECK OPTION").
- You should be able to explain how view updates are done, why not all views are updatable, and which views are.





## Overview







## Views (1)

- The result of an SQL query is a table, so you should be able to use it in queries like a stored table.
- Views make this possible. You define a view by specifying an SQL query and a table name (optionally column names):

```
CREATE VIEW My_Courses(No, Title) AS
SELECT CRN, Title
FROM   Course
WHERE  Name = 'Brass, S.'
```

- So views are derived, virtual tables which are computed from the (actually stored) base tables.

## Views (2)

- Then you can use the view in queries:

```
SELECT *
FROM   My_Courses
WHERE  Title LIKE 'Database%'
```

- From the definition of the view “My_Courses” and this query, the DBMS computes an equivalent query which refers only to the base tables. That query is then actually executed:

```
SELECT CRN No, Title
FROM   Course
WHERE  Title LIKE 'Database%'
AND    Name = 'Brass, S.'
```





## Views (3)

- This translation was quite complicated in SQL-86, and resulted in a number of restrictions on the use of views.
- SQL-92 allows subqueries in the FROM clause. Then the DBMS only has to replace the view name by its definition:

```
SELECT *
FROM   (SELECT CRN No, Title
        FROM   Course
        WHERE  Name = 'Brass, S.' ) My_Courses
WHERE  Title LIKE 'Database%'
```

- 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 (4)

- Note the difference of the CREATE VIEW to

```
CREATE TABLE My_Courses(...);
INSERT INTO My_Courses(No, Title) SELECT ...
```

- 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.
- So if the base table (Course) changes, the view will automatically reflect the change, whereas the above table needs to be updated manually (or by means of a trigger).





## Views (5)

- Views can be used in the definitions of other views.
- However, recursive views are excluded.
  DB2 supports recursive view definitions (but you might get a warning that the view can result in an infinite loop).
- Deductive databases allow also recursive views.
  Thus you can work with tree-structured data (hierarchies). and compute “transitive closure” / “bill of materials” queries.
- SQL-3 will also allow limited forms of recursion.
- A view can be deleted in the following form:

```
DROP VIEW My_Courses
```

- Updates on views are a difficult problem, see below.

# Advantages of Views

- 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.
- Adaption of the database schema to the wishes of different users / user groups.
- 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_Courses(Name, Title) AS
SELECT Name, Title
FROM   Course
WHERE  Name = 'Brass, S.'
```

- Now consider the command:

```
INSERT INTO My_Courses
VALUES ('Spring, M.', 'Client-Server');
```





# View Update Problem (2)

- Even if we would enter this tuple 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 `Course`, so it is not null, and we would have to invent a value.
- So 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 is more than one translation.

## View Update Problem (3)

- Example for an ambigious update request: 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   Course C, Instructor I
WHERE  C.Name = I.Name
```

**Course**

| CRN | Title | Name |
|---|---|---|
| 22268 | Databases | Brass |
| 31822 | Client-Server | Spring |

**Instructor**

| Name | Phone |
|---|---|
| Brass | 9404 |
| Spring | 9429 |





## View Update Problem (4)

- So 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 (5)

- Did Michael Spring take over the database course?

```
UPDATE Course
SET    Name = 'Spring'
WHERE  Title = 'Databases'
```

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

```
UPDATE Instructor
SET    Phone = 9429
WHERE  Name = 'Brass'
```

- The system has no way to decide this.





## View Update Problem (6)

- So views are in general read-only and cannot be updated.
- However, there are exceptions, the most important being views which are defined by a selection and a projection on a single table, where the "projected out" columns can be null.
- 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 View Updates (1)

- Consider a deletion from the view My_Courses:

```
CREATE VIEW My_Courses(No, Title) AS
SELECT CRN, Title FROM Course
WHERE  Name = 'Brass, S.'

DELETE FROM My_Courses
WHERE  Title LIKE '%Database%'
```

- The deletion is translated into:

```
DELETE FROM Course
WHERE  Title LIKE '%Database%'
AND    Name = 'Brass, S.'
```





## Translation of View 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 Course 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, 'Data Structures')
```

  This is translated into:

```
INSERT INTO Course(CRN, Title)
VALUES (42232, 'Data Structures')
```





## Translation of View Updates (3)

- So the attribute Name, which was projected out in the view, will be set to the declared default value (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 Course.
- 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 has no possibility to specify a value.

- You can use the condition "Name = USER" in the view definition. This would show every user his/her own courses.





## Updatable Views (1)

Restrictions in SQL-92:

- The FROM-clause contains only a single table.
- The SELECT-clause contains no DISTINCT.
- The SELECT-clause contains 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, only a single SELECT-expression.

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

# Updatable Views (2)

- Oracle allows also certain updates on views defined by joins.

  You can also define the the result of view updates 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”.

```
CREATE VIEW Course_Inf(CRN, Title, Name, Phone)
AS SELECT CRN, Title, C.Name, Phone
   FROM   Course C, Instructor I
   WHERE  C.Name = I.Name
```

  Then columns CRN and Title would be updatable. Deletions from `Course_Inf` are translated into deletions from `Course`. Insertions may not specify a value for `Phone`.





# Updatable Views (3)

- 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, but allows `UNION ALL` under certain conditions, and does not forbid data type operations.
- A column of a view is updatable, if the view is deletable and that column does not contain data type operations.
- A view is insertable if all columns of it are updatable and no `UNION ALL` is used.
- SQL Server seems to be quite generous with updatability, but the manual says very little about this problem.





# Overview

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





# Check Option (1)

- Suppose that a lower level personnel manager is only allowed to work with employees up to $4000 monthly income.
- So we create a view

```
CREATE VIEW Restricted_Emp AS
SELECT EmpNo, EName, Sal
FROM   Emp
WHERE  Sal <= 4000
```

- 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 (so that the update becomes a deletion from the view, and a security violation for the base table).

  The translated command checks that the input tuples satisfy the condition, but it doesn't check the output tuples — the SQL UPDATE command has no corresponding clause.





## 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 the DBMS will refuse insertions/updates which 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 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).

  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.