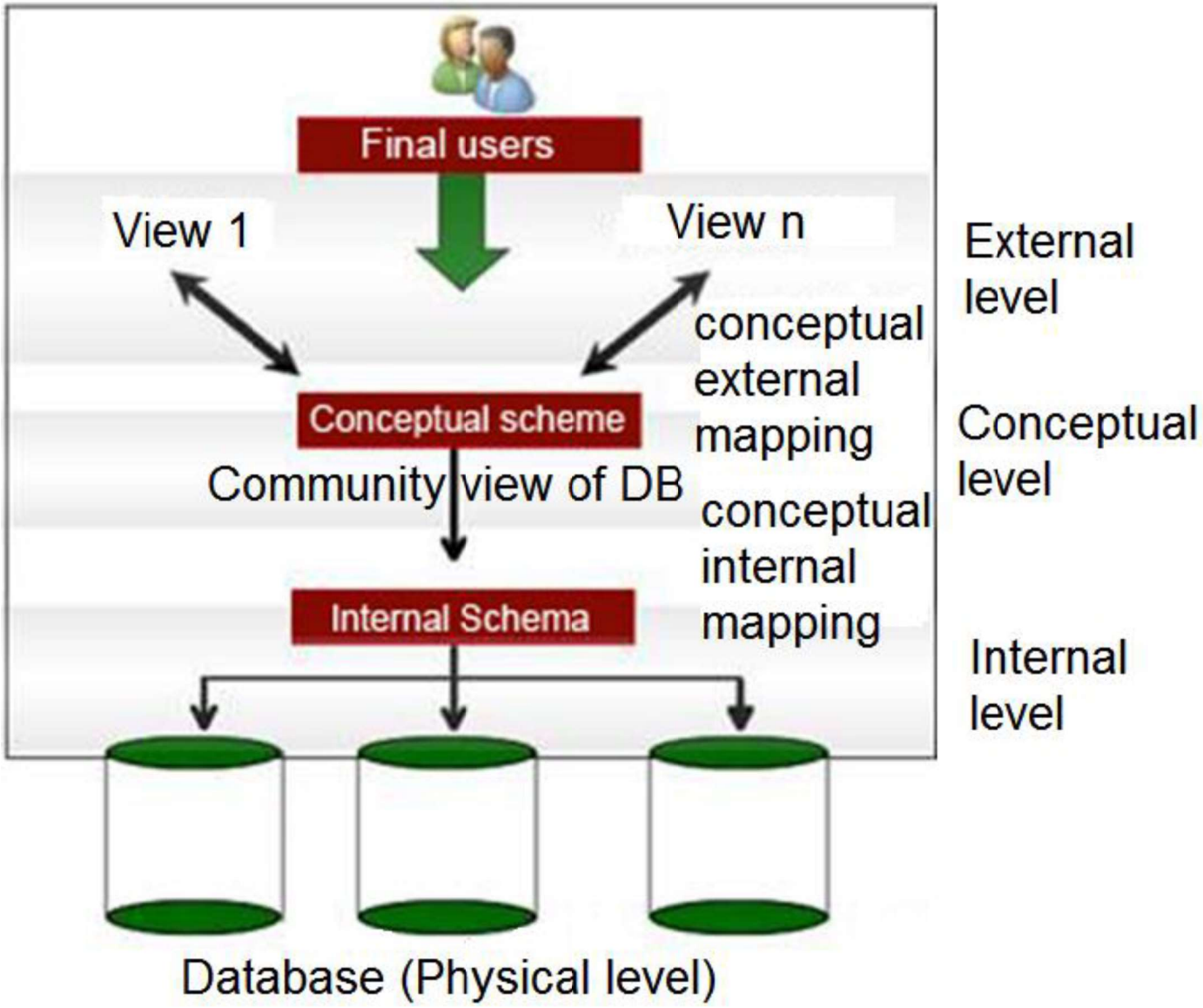
Week 5

Views

ANSI/X3 SPARC Architecture for databases



The ANSI/X3 SPARC DBMS Framework: Report of the Study Group on Database Management Systems (1977)

Views

- Physical, conceptual, logical levels

- Why Views?
  - Hide some data from some users
  - Make some queries easier
  - Modularity of database access (customized access = access to parts of the database)
  - Powerful and flexible security mechanism

  The bigger the application, more views are used

Views

- A view is a ‘virtual relation’ that does not actually exist in the database but is produced upon request, at the time of the request.

- Base relation = a named relation corresponding to an entity in the conceptual schema, whose tuples are physically stored in the database.

- View = The dynamic result of one ore more relational operations operating on the base relations to produce another relation.

Defining and using views

- View V = Query(R_1, R_2, …, R_n) where R_i is a table or another view.
- Schema of V = schema of query result
- “Temporary table”
- In reality, the DBMS re-writes the query Q to use R_1, R_2, …, R_n instead of V

Views in SQL

- CREATE VIEW ViewName AS Query
- CREATE VIEW ViewName(A_1, A_2, …, A_n) AS Query
  - Query – SQL SELECT query
  - Creates the view as an object in the database catalogue; query is run every time the view is opened / used

- ALTER VIEW – modifies an existing view

- DROP VIEW ViewName;
  - Some DBMS returns an error if View is used in other queries; others returns the error only when the query involving the view is run.

- SELECT * FROM ViewName - queries an existing view

Examples

```
Courses(CourseTitle:CHAR(50), Department:CHAR(20),
Credits:INTEGER)

Students(StudID:INTEGER, StudName:CHAR(50), DoB:DATE,
PoB:CHAR(50), Major:CHAR(40), TotalCredits:INTEGER)

Enrollments(StudID:INTEGER, CourseTitle:CHAR(50),
EnrollmentDate:DATE, Decision:BOOLEAN)

CREATE VIEW DBAccepted AS
     SELECT StudID, EnrollmentDate FROM Enrollments
     WHERE CourseTitle='Database' AND Decision='Y'

CREATE VIEW DBAccepted2 AS
     SELECT Students.StudID, StudName, Major, TotalCredits
     FROM Students, DBAccepted
     WHERE Students.StudID = DBAccepted.StudID
```

Examples

```
Courses(CourseTitle:CHAR(50), Department:CHAR(20),
Credits:INTEGER)

Students(StudID:INTEGER, StudName:CHAR(50), DoB:DATE,
PoB:CHAR(50), Major:CHAR(40), TotalCredits:INTEGER)

Enrollments(StudID:INTEGER, CourseTitle:CHAR(50),
EnrollmentDate:DATE, Decision:BOOLEAN)

CREATE VIEW DBAccepted AS
     SELECT StudID, EnrollmentDate FROM Enrollments
     WHERE CourseTitle='Database' AND Decision='Y'

// With Query re-write (v1)
CREATE VIEW DBAccepted2 AS
     SELECT Students.StudID, StudName, Major, TotalCredits
     FROM Students,
          (SELECT StudID, EnrollmentDate FROM Enrollments
           WHERE CourseTitle='Database' AND Decision='Y')
               AS DBAccepted
     WHERE Students.StudID = DBAccepted.StudID
```

Examples

```
Courses(CourseTitle:CHAR(50), Department:CHAR(20),
Credits:INTEGER)

Students(StudID:INTEGER, StudName:CHAR(50), DoB:DATE,
PoB:CHAR(50), Major:CHAR(40), TotalCredits:INTEGER)

Enrollments(StudID:INTEGER, CourseTitle:CHAR(50),
EnrollmentDate:DATE, Decision:BOOLEAN)

CREATE VIEW DBAccepted AS
     SELECT StudID, EnrollmentDate FROM Enrollments
     WHERE CourseTitle='Database' AND Decision='Y'

// With Query re-write (v2)
CREATE VIEW DBAccepted2 AS
     SELECT Students.StudID, StudName, Major, TotalCredits
     FROM Students, Enrollments
     WHERE CourseTitle='Database' AND Decision='Y' AND
           Students.StudID = Enrollments.StudID
```

Examples

```
Courses(CourseTitle:CHAR(50), Department:CHAR(20),
Credits:INTEGER)

Students(StudID:INTEGER, StudName:CHAR(50), DoB:DATE,
PoB:CHAR(50), Major:CHAR(40), TotalCredits:INTEGER)

Enrollments(StudID:INTEGER, CourseTitle:CHAR(50),
EnrollmentDate:DATE, Decision:BOOLEAN)

CREATE VIEW DBAccepted AS
     SELECT StudID, EnrollmentDate FROM Enrollments
     WHERE CourseTitle='Database' AND Decision='Y'

// Using views
SELECT * FROM DBAccepted WHERE EnrollmentDate < '01-nov-2015'
```

View modification

- Can a view V be modified (insert/delete/update) as any other table?
  - Remember, V is not stored => doesn't make much sense
  - Some users only see the views => it should be possible
- SOLUTION: Modification of V is rewritten (automatically by the system) to modify the base tables.
- Ambiguities: there may be multiple rewrites; which one the one user wanted? Example.

View modification approaches

- Restrict modifications so that the translation to base table modifications is meaningful and unambiguous
  - (+) No user intervention
  - (-) Restrictions may be significant
  - Imposed by SQL standard

- View creator specifies the rewriting process, i.e. what happens in case of delete/update/insert
  - (+) Can handle all modifications
  - (-) No guarantee of correctness
  - Enabled by INSTEAD OF triggers

View modification using automatic view modification

- Restrictions in SQL standard for ‘updatable views’:
  - SELECT (no DISTINCT) on a single table T
  - Attributes not in view can be NULL (or default value)
  - Sub-queries must not refer to T
  - No GROUP BY or HAVING

- Not supported by all DBMS (e.g. MySQL supports it)

View modification using automatic view modification

Example: (an updateable view)

```
CREATE VIEW DBAccepted(ID, Curs, Date) AS
    SELECT StudID, CourseTitle, EnrollmentDate
        FROM Enrollments
        WHERE CourseTitle='Database' AND Decision='Y'
```

- `DELETE FROM DBAccepted WHERE ID=1234`
- `UPDATE DBAccepted SET Date = '12-dec-2014' WHERE ID=1234`
- `INSERT INTO DBAccepted (ID, Curs, Date)`
  `VALUES (567, 'Networks', '15-dec-2015')`

View modification using automatic view modification

Example: (an updateable view)

```
CREATE VIEW DBAccepted(ID, Curs, Date) AS
    SELECT StudID, CourseTitle, EnrollmentDate
        FROM Enrollments
        WHERE CourseTitle='Database' AND Decision='Y'
```

- `DELETE FROM DBAccepted WHERE ID=1234`
- `UPDATE DBAccepted SET Date = '12-dec-2014' WHERE ID=1234`
- `INSERT INTO DBAccepted (ID, Curs, Date)`
  `VALUES (567, 'Database', '15-dec-2015')`
  - Value for Decision is not set according to view definition (it is given the default value / NULL)
  - To avoid insertion of undesired tuples, use WITH CHECK OPTION in view definition and then the above INSERT is flagged as erroneous.
  - WITH CHECK OPTION also prevents rows from migrating out of the view
  - WITH LOCAL/CASCADE CHECK OPTION – apply check on underlying views

View modification using automatic view modification – WITH CHECK OPTION

Example: (an updateable view)

```
CREATE VIEW DBAccepted(ID, Curs, Date) AS
    SELECT StudID, CourseTitle, EnrollmentDate
    FROM Enrollments
    WHERE CourseTitle='Database' AND Decision='Y'
```

Will the following update be accepted / rejected?

```
UPDATE DBAccepted
    SET CourseTitle = 'Networks'
    WHERE ID = 1234;
```

Exercise

Given the following views
CREATE VIEW **LowSalary** AS SELECT * FROM Staff WHERE salary > 9000

CREATE VIEW **HighSalary** AS SELECT * FROM LowSalary
WHERE salary > 10000
WITH LOCAL CHECK OPTION;

CREATE VIEW **Manager3Staff** AS SELECT * FROM HighSalary WHERE branch=10;

which of the following updates will be rejected/accepted by the DBMS?

a) UPDATE Manager3Staff SET salary = 9500 WHERE EmpId = 1234

b) UPDATE Manager3Staff SET salary = 8000 WHERE EmpId = 1234

c) UPDATE Manager3Staff SET salary = 11000 WHERE EmpId=1234

Exercise

- Which of the following views are updateable?

a)
```
CREATE VIEW View2 AS
        SELECT StudName, Major, AVG(TotalCredits)
        FROM Students
        GROUP BY Major;
```

b)
```
CREATE VIEW View3 AS
        SELECT DISTINCT Major FROM Students;
```

View modification using triggers

```
CREATE VIEW DBAccepted AS
      SELECT StudID, EnrollmentDate FROM Enrollments
      WHERE CourseTitle='Database' AND Decision='Y'


DELETE FROM DBAccepted WHERE StudID=12345
        => error
```

View modification using triggers

```
CREATE VIEW DBAccepted AS
      SELECT StudID, EnrollmentDate FROM Enrollments
      WHERE CourseTitle='Database' AND Decision='Y'


DELETE FROM DBAccepted WHERE StudID=12345
       => error

CREATE TRIGGER DBAcceptedDelete
INSTEAD OF DELETE ON DBAccepted
REFERENCING OLD ROW AS OldRow
FOR EACH ROW
DELETE FROM Enrollments
      WHERE StudID = OldRow.StudID AND
            EnrollmentDate = OldRow.EnrollmentDate AND
            CourseTitle='Database' AND
            Decision='Y'
```

Remark: tuples of DBAccepted don't physically exist, but the `Old` variable **is** bind to those that need to be logically deleted!

View modification using triggers

```
CREATE VIEW DBAccepted AS
      SELECT StudID, EnrollmentDate FROM Enrollments
      WHERE CourseTitle='Database' AND Decision='Y'


UPDATE DBAccepted SET EnrollmentDate='01-nov-2015' WHERE
StudID=12345
        => error
```

View modification using triggers

```
CREATE VIEW DBAccepted AS
      SELECT StudID, EnrollmentDate FROM Enrollments
      WHERE CourseTitle='Database' AND Decision='Y'


UPDATE DBAccepted SET EnrollmentDate='01-nov-2015' WHERE
StudID=12345


CREATE TRIGGER DBAcceptedUpdate
INSTEAD OF UPDATE OF EnrollmentDate ON DBAccepted
REFERENCING NEW ROW AS NewRow, OLD ROW AS OldRow
FOR EACH ROW
UPDATE Enrollments SET EnrollmentDate = NewRow.
EnrollmentDate
      WHERE StudID = OldRow.StudID AND
            EnrollmentDate = OldRow.EnrollmentDate AND
            CourseTitle='Database' AND
            Decision='Y';
```

CAUTION: Writing incorrect trigger will modify unexpected tuples that may even not be part of the view!

Materialized views

- View V = Query(R_1, R_2, …, R_n) where R_i is a table or another view.
- Create a physical table V with schema of query result
- **Execute Query and put results in V**
- Queries refer to V as if it is a table
- (+) Advantage of materialized view: improved query performance
- (-) V can be very large
- (-) Modifications to R_1, R_2, …, R_n => recompute/modify V

Advantages of materialized views

- Hide some data from some users
- Make some queries more natural to express
- Modularity of database access
- Improve query performance

Example

- ```
  CREATE MATERIALIZED VIEW MV1
    SELECT StudName, Enrollments.CourseTitle, Credits
    FROM Courses, Students, Enrollments
    WHERE Enrollments.StudID = Students.StudID AND
          Enrollments.CourseTitle = Courses.CourseTitle AND
          Students.Major = 'CS'
  ```
  ```

Materialized views and modifications

- Modifications to base relations invalidate the view (example)
- DBMS need to maintain the view status
- Modifications on materialized view
  - Just update the stored table
  - Base tables need to be synchronized => same issue with virtual views
- Materialized views are often used to improve performance, hence users will not be allowed to modify them

Design of materialized views

- Which materialized views to create?

- **Efficiency benefits** of materialized views depend on:
  - Size of data
  - Complexity of the view
  - Number of queries using view
  - Number of modifications affecting the view
  - Incremental maintenance vs. Full re-computation

- Analyse the workload using above criteria

- Materialized views generalize the concept of index

- Automatic query rewriting to use materialized views
  - DBMS transparently use existing materialized views to rewrite users' queries, without users even knowning

Example

```
CREATE MATERIALIZED VIEW CSEnrollments
 SELECT StudID, CourseTitle, Decision FROM Enrollments
 WHERE StudID IN (SELECT StudID FROM Students WHERE
Major='CS')
```

Try to re-write the following query using CSEnrollments :

```
SELECT DISTINCT StudID, TotalCredits
FROM Courses, Students, Enrollments
WHERE Courses.CourseTitle=Enrollments.CourseTitle AND
      Students.StudID=Enrollments.StudID AND
      Courses.Department='INFO' AND Students.Major='CS'
```

Example

```
CREATE MATERIALIZED VIEW CSEnrollments
 SELECT StudID, CourseTitle, Decision FROM Enrollments
 WHERE StudID IN (SELECT StudID FROM Students WHERE
Major='CS')
```

Try to re-write the following query using CSEnrollments :

```
SELECT DISTINCT StudID, TotalCredits
FROM Courses, Students, Enrollments
WHERE Course.CourseTitle=Enrollments.CourseTitle AND
      Students.StudID=Enrollments.StudID AND
      Courses.Department='INFO' AND Students.Major='CS'
```

```
SELECT DISTINCT StudID, TotalCredits
FROM Courses, CSEnrollments
WHERE Courses.CourseTitle=CSEnrollments.CourseTitle AND
      Courses.Department='INFO'
```

‘Parameterized’ views

- Standard-wise, does not exist
- In general, not available
- MS SQL Server Functions
  - scalar functions: return only scalar/single value; used in SELECT clauses
  - table valued functions (TVF): return a table (set of rows) -> alternative to views (parameterized views)
  - Inline table valued functions (iTVF): contains only one (return) statement that defines the rows/columns to be returned

Inline table valued functions

- Unlike Stored Procedures / multi-statement TVF, the database engine handles this inline TVF as a VIEW
- It computes the execution plan using the statistics on the tables used by this function (similar to views)
- There is no extra load of creating a table variable
- Better in performance than SP / multi-statement TVF

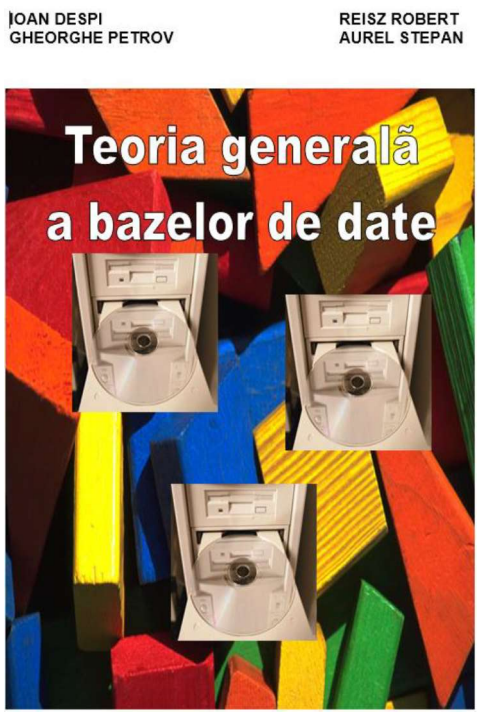
Inline table valued function - Example

```
-- Definition
CREATE FUNCTION EnrolledStudents (@CourseTitle VARCHAR(50))
RETURNS TABLE
AS
RETURN
SELECT S.*, E.EnrollmentDate, E.Decision
FROM Students S
INNER JOIN Enrollments E ON S.StudID = E.StudID
WHERE E.CourseTitle = @CourseTitle

-- Usage
SELECT * FROM EnrolledStudents('Database') WHERE Decision='Y'

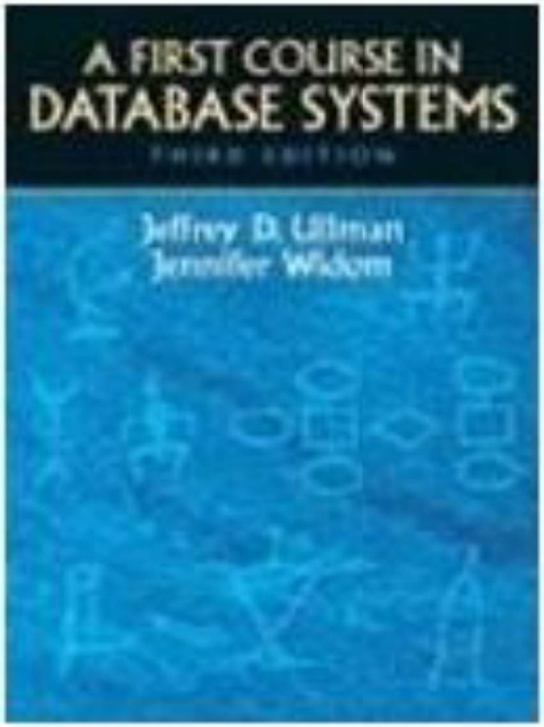
-- Removal
DROP FUNCTION EnrolledStudents
```

Bibliography (recommended)



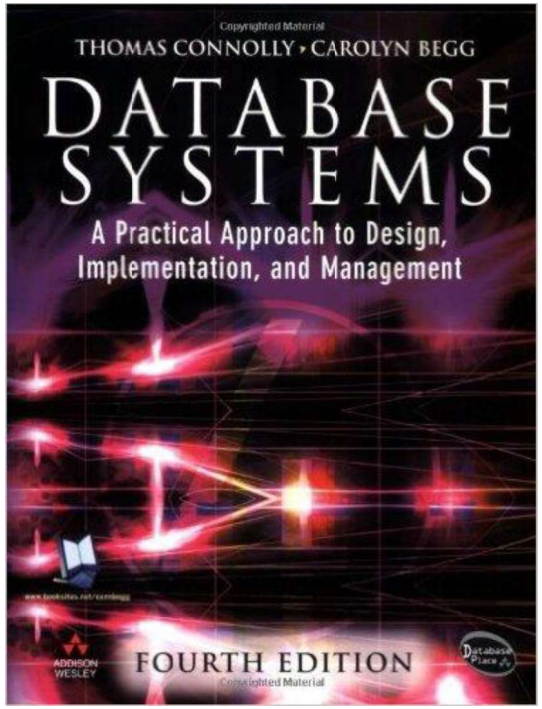
Teoria generala a bazelor de date, I. Despi, G. Petrov, R. Reisz, A. Stepan, Mirton, 2000

Cap 11.2.5



A First Course in Database Systems (3rd edition) by Jeffrey Ullman and Jennifer Widom, Prentice Hall, 2007

Chapter 8.1, 8.2, 8.5



Database Systems - A Practical Approach to Design, Implementation, and Management (4th edition) by Thomas Connolly and Carolyn Begg, Addison-Wesley, 2004

Chapter 3.4, 6.4