VirginiaTech

CS 4604: Introduction to Database Management Systems

B. Aditya Prakash

Lecture #7: Views

Views

- A view is a relation that does not exist physically.
- A view is defined by a query over other relations (tables and/or views).
- Just like a table, a view can be
  - queried: the query processor replaces the view by its definition.
  - used in other queries.
- Unlike a table, a view cannot be updated unless it satisfies certain conditions.

Example: View Definition

- CREATE VIEW ViewName AS Query;

- Suppose we want to perform a set of queries on those students who have taken courses both in the computer science and the mathematics departments.
- Let us create a view to store the PIDs of these students and the CS-Math course pairs they took.

Example: View Definition

- Suppose we want to perform a set of queries on those students who have taken courses both in the computer science and the mathematics departments.
- Let us create a view to store the PIDs of these students and the CS-Math course pairs they took.

```
CREATE VIEW CSMathStudents AS
 SELECT T1.StudentPID, T1.Number AS CSNum, T2.Number AS MathNum
 FROM Take AS T1, Take AS T2
 WHERE (T1.StudentPID = T2.StudentPID)
      AND (T1.DeptName = 'CS')
      AND (T2.DeptName = 'Math');
```

Querying Views

- Query a view as if it were a base table.
- How many students took both CS and Math courses?

```
SELECT COUNT(StudentPID)
FROM CSMathStudents
```

Querying Views

- Just replace view by its definition

```
SELECT COUNT(StudentPID)
FROM CSMathStudents

SELECT COUNT(StudentPID)
FROM
   (SELECT T1.StudentPID, T1.Number AS CSNum,
 T2.Number AS MathNum
  FROM Take AS T1, Take AS T2
  WHERE (T1.StudentPID = T2.StudentPID)
        AND (T1.DeptName = 'CS')
        AND (T2.DeptName = 'Math'));
```

Modifying Views

- What does it mean to modify a view?
- How is tuple deletion from a view executed?
- Can we insert a tuple into a view? Where will it be inserted, since a view does not physically exist?
- Can we insert tuples into any view? SQL includes rules that specify which views are updatable.

Deleting Views

- DROP VIEW CSMathStudents;

- Like a Symbolic Link: only the view definition is deleted

Deleting Tuples from Views

- Delete tuples for students taking 'CS 4604'.

  DELETE FROM CSMathStudents

  WHERE (CSNum = 4604);

- Deletion is executed as if were executing

  DELETE FROM Take

  WHERE (Number = 4604);

- Incorrect: non-CS tuples where (Number = 4604) will be deleted.

Deleting Tuples from Views

- Tuples only seen in the view should be deleted!
- Add conditions to the WHERE clause

```
DELETE FROM CSMathStudents
WHERE (CSNum = 4604) AND (DeptName = 'CS');
```

Inserting tuples into Views

- Again, passed through to the underlying relation

```
INSERT INTO CSMathStudents
VALUES ('123-45-6789', 4604, 8811);
```

- But Take schema is (PID, Number, Dept)
  - what should dept values be?
  - NULL?

    Then it is not part of CSMathStudents!

Inserting tuples into Views

- CREATE VIEW CSStudents AS
  SELECT StudentPID, Number
  FROM Take
  WHERE (DeptName = 'CS');

Works?

- INSERT INTO CSStudents
  VALUES ('123-45-6789', 4604);

Same Problem

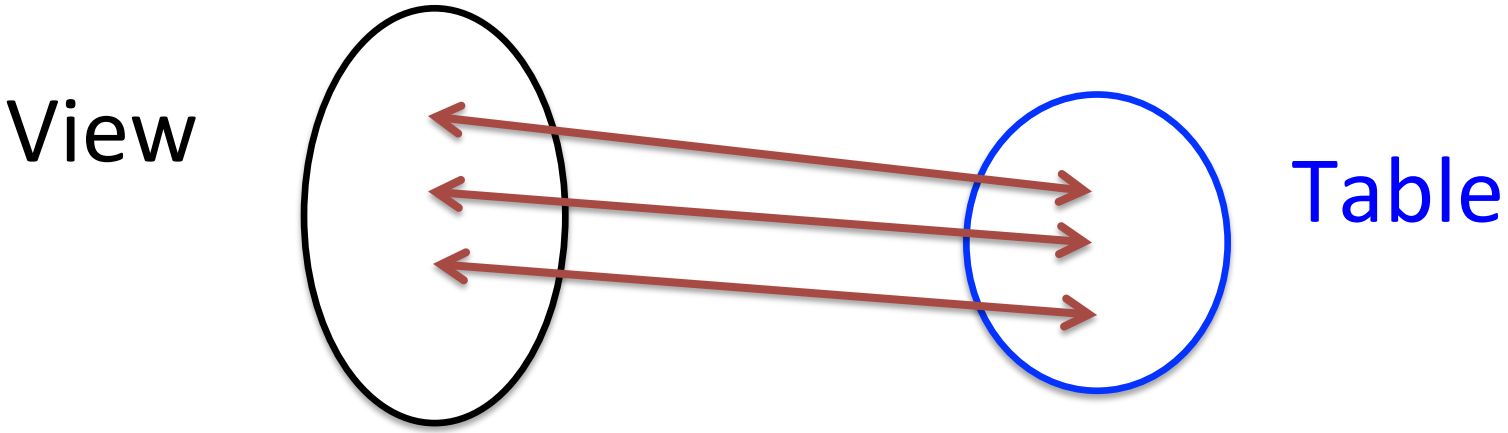
Inserting tuples into Views

- Include DeptName in the view's schema
- CREATE VIEW CSStudents AS
  SELECT StudentPID, DeptName, Number
  FROM Take
  WHERE (DeptName = 'CS');

- INSERT INTO CSStudents
  VALUES ('123-45-6789', 'CS', 4604)

Updatable Views

- The idea is that there must be a one-one relationship between rows in the view and the rows in the underlying table



Updatable Views

- SQL has complex rules
- Defined by selecting some attributes from one relation R
- R may itself be an updatable view.
- Use SELECT and not SELECT DISTINCT.
- WHERE clause must not involve R in a sub-query.
- FROM clause can contain only one occurrence of R and must not contain any other relation.
- SELECT clause must contain enough attributes so that for every tuple inserted into the view, other attributes can get NULL values or default values.
  - An attribute that is declared NOT NULL and has no default must be mentioned in the SELECT clause.

Materialized Views

- Two kinds:
  1. ***Virtual*** = not stored in the database; just a query for constructing the relation.
  2. *Materialized* = actually constructed and stored.

  WHY?
  - Some views may be frequently used in queries.
  - It may be efficient to materialize such a view, i.e., maintain its value at all times as a physical table

Declaring Views

- Declare by:

  CREATE [MATERIALIZED] VIEW <name> AS <query>;

- Default is virtual.

Maintaining Materializing Views

- Cost?
  - Re-computing it when the underlying tables change
  - Materialized view may be much larger than original relations, e.g., in the case of joins

Maintaining Materialized Views

- CREATE MATERIALIZED VIEW CSStudents AS
  SELECT StudentPID, DeptName, Number
  FROM Take
  WHERE (DeptName = 'CS');
- When?
  - Insertion/deletion/update of Take
- Cost?
  - Insertion of tuple: Insert tuple into CSStudents only if new tuple has DeptName = 'CS'
  - Same for Deletion
  - Update? Delete followed by an Insert…

Maintaining Materialized Views

- Key idea is that many materialized views can be updated incrementally.

- Incremental maintenance of a view that involves a join: read Chapter 8.5.1 of the textbook.

Maintaining Materialized Views with Joins

- CREATE MATERIALIZED VIEW CSMathProfs(PID, Pname, CNum, CName) AS
  SELECT PID, P.Name, T.Number, T.Name
  FROM Teach AS T, Professors AS P
  WHERE (P.DeptName = 'CS') AND (T.DeptName = 'Math') AND (T.ProfessorPID = P.PID);

- Insert a tuple t into Teach:

- Delete a tuple t into Teach:

Maintaining Materialized Views with Joins

- CREATE MATERIALIZED VIEW CSMathProfs(PID, Pname, CNum, CName) AS
  SELECT PID, P.Name, T.Number, T.Name
  FROM Teach AS T, Professors AS P
  WHERE (P.DeptName = 'CS') AND (T.DeptName = 'Math') AND (T.ProfessorPID = P.PID);

- Insert a tuple t into Teach (assume t.DeptName = Math):
  Find the tuple p in Professors such that (t.ProfessorPID = p.PID) AND (p.DeptName = 'CS').
  Insert (p.PID, p.Name, t.Number, t.Name) into CSMathProfs

Maintaining Materialized Views with Joins

- CREATE MATERIALIZED VIEW CSMathProfs(PID, Pname, CNum, CName) AS
  SELECT PID, P.Name, T.Number, T.Name
  FROM Teach AS T, Professors AS P
  WHERE (P.DeptName = 'CS') AND (T.DeptName = 'Math') AND (T.ProfessorPID = P.PID);

- Delete a tuple t from Teach (assume t.DeptName = Math):
  DELETE FROM CSMathProfs WHERE CNum = t.Number;

Maintaining Materialized Views with Joins

- CREATE MATERIALIZED VIEW CSMathProfs(PID, Pname, CNum, CName) AS
  SELECT PID, P.Name, T.Number, T.Name
  FROM Teach AS T, Professors AS P
  WHERE (P.DeptName = 'CS') AND (T.DeptName = 'Math') AND (T.ProfessorPID = P.PID);

- Insert a tuple t into Professors:

- Delete a tuple t into Professors:

Maintaining Materialized Views with Joins

- CREATE MATERIALIZED VIEW CSMathProfs(PID, Pname, CNum, CName) AS
  SELECT PID, P.Name, T.Number, T.Name
  FROM Teach AS T, Professors AS P
  WHERE (P.DeptName = 'CS') AND (T.DeptName = 'Math') AND (T.ProfessorPID = P.PID);

- Insert a tuple t into Professors (assume p.DeptName = CS):
  INSERT INTO CSMathProfs
  SELECT p.PID, p.Name, T.Number, T.Name
  WHERE (p.PID = T.ProfessorPID) AND (T.DeptName = 'Math');

Maintaining Materialized Views with Joins

- CREATE MATERIALIZED VIEW CSMathProfs(PID, Pname, CNum, CName) AS
  SELECT PID, P.Name, T.Number, T.Name
  FROM Teach AS T, Professors AS P
  WHERE (P.DeptName = 'CS') AND (T.DeptName = 'Math') AND (T.ProfessorPID = P.PID);

- Delete a tuple t from Professors (assume p.DeptName = CS):
  DELETE FROM CSMathProfs WHERE (PID = p.PID);

Periodic Maintenance

- DB for inventory of a department store.
- Aggregate buyer patterns for further analysis → can be a (materialized) view
- Analysis is only periodic, so update the materialized view at only regular intervals
- Automatic creation of materialized views: Read Chapter 8.5.4 of the textbook.

Rewriting Queries Using Materialized Views

- In practice, views are materialized because they are helpful to answer common queries.

- Can we rewrite a query to use a materialized view rather than the original relations?

Rewriting Queries Using Materialized Views

- Find names and addresses of students taking CS courses

```
SELECT Name, Address
FROM Students, Take
WHERE (Students.PID = Take.StudentPID) AND
(DeptName = 'CS');
```

Rewrite it using CSStudents?

```
SELECT Name, Address
FROM Students, CSStudents
WHERE (Students.PID = CSStudents.StudentPID);
```

Rules for Rewriting Queries

- Complete sets of rules is very complex!
- A simple rule

View V:	Query Q:	(New) Query Q':
SELECT *LV*	SELECT *LQ*	SELECT *LQ*
FROM *RV*	FROM *RQ*	FROM V, *RQ - RV*
WHERE *CV*	WHERE *CQ*	WHERE *C*

- We can replace Q by the new query Q' if
  - RV $\subseteq$ RQ
  - CQ == CV AND C, for some condition C, which may be empty
  - If C is not empty, then attributes of relations in RV that C mentions are also in LV
  - Attributes in LQ that come from relations in RV are also in the list of attributes LV