Views

CSC 321/621 – 3/20/2012

Views

So far, database = collection of relations (tables)

- Relations explicitly defined with CREATE TABLE are called “base relations”

Sometimes useful to present “virtual tables” – tables that are provide an alternative *view* of the data

- Such relations may not actually exist in storage but can be dynamically generated as need be
- Technical definition from book: *The dynamic result of one or more relational operations operating on the base relations to produce another relation.*

Section 4.4 of text

Views: Example

Student

studentID	lastName	firstName	year	major	GPA
1123	Smith	Robert	4	CSC	3.5
1129	Jones	Douglas	3	MTH	2.9
1145	Brady	Susan	4	CSC	3.8

A list of all students at WFU

CSCStudentForDirectory

Student ID	Last Name	First Name	Class
1123	Smith	Robert	4
1145	Brady	Susan	4

Only those students who are majoring in CSC at WFU

Results from a selection (major='CSC') and projection (dropping 'major' and 'GPA') on Student; also renamed columns

Views: Advantages

Advantages of views:

Along with access control (*to be covered soon*), supports information security by allowing only required parts of database to be exposed to a given user

Allows customization of database interface to user needs

Can simplify to end user complex database operations (instead of having to perform a join, the user can manipulate the table resulting from a join managed by the DBMS)

Views: Example

Course

courseID	dept.	number
06902	CSC	221
06903	CSC	231
06904	CSC	241
06905	CSC	191

Enrollment

studentID	courseID
1123	06902
1129	06902
1145	06902
1123	06903
1145	06903
1123	06904
1129	06904

Course Number	Student ID
221	1123
221	1129
221	1145
231	1123
231	1145
241	1123
241	1129

A table resulting from a join on Course, Enrollment where dept=CSC and courseIDs match

Views: Advantages

When revising/refactoring a database

- Adding columns
- Splitting tables

the end-user can be made to believe the database was exact same as before by appropriate view definition.

Views: SQL Syntax For Creation

- CREATE VIEW ViewName [(newColumnName [,…])] AS subselect [WITH[CASCADED | LOCAL] CHECK OPTION]
  - We'll cover some of the optional components soon…

- Simply:
  - Assigning a name to the result of the *defining query*
  - Can re-assign column names
    - Must re-assign if subselect generates columns with the same name

Section 7.4 of text

Views: Typical View Scenarios

- *Horizontal* views: Select rows of one or more tables
- CREATE VIEW CSCStudent AS SELECT * FROM Student WHERE major='CSC';

Student

studentID	lastName	firstName	year	major	GPA
1123	Smith	Robert	4	CSC	3.5
1129	Jones	Douglas	3	MTH	2.9
1145	Brady	Susan	4	CSC	3.8

CSCStudent

studentID	lastName	firstName	year	major	GPA
1123	Smith	Robert	4	CSC	3.5
1145	Brady	Susan	4	CSC	3.8

Views: Typical View Scenarios

- *Vertical* views: Select columns of one or more tables
- CREATE VIEW CSCStudentForDirectory (StudentID,LastName,FirstName,Class) AS SELECT studentId,lastName,firstName,year FROM Student WHERE major='CSC';

Student

studentID	lastName	firstName	year	major	GPA
1123	Smith	Robert	4	CSC	3.5
1129	Jones	Douglas	3	MTH	2.9
1145	Brady	Susan	4	CSC	3.8

CSCStudentForDirectory

Student ID	Last Name	First Name	Class
1123	Smith	Robert	4
1145	Brady	Susan	4

Section 7.4 of text

Views: Views From Views

- Note, we *can* create views from views (as views are relations)
- CREATE VIEW CSCStudentForDirectory (StudentID,LastName,FirstName,Class) AS SELECT studentId,lastName,firstName,year FROM CSCStudent;

CSCStudent

studentID	lastName	firstName	year	major	GPA
1123	Smith	Robert	4	CSC	3.5
1145	Brady	Susan	4	CSC	3.8

This itself is a view…

CSCStudentForDirectory

Student ID	Last Name	First Name	Class
1123	Smith	Robert	4
1145	Brady	Susan	4

from which we extract this.

Section 7.4 of text

Views: Typical View Scenarios

- *Grouped* views: Subselect uses GROUP BY clause
- CREATE VIEW MajorCount (major,cnt) AS SELECT major, COUNT(*) FROM Student GROUP BY major;

Student

studentID	lastName	firstName	year	major	GPA
1123	Smith	Robert	4	CSC	3.5
1129	Jones	Douglas	3	MTH	2.9
1145	Brady	Susan	4	CSC	3.8

MajorCount

major	cnt
CSC	2
MTH	1

Section 7.4 of text

Views: Example

Course

courseID	dept.	number
06902	CSC	221
06903	CSC	231
06904	CSC	241
06905	CSC	191

Enrollment

studentID	courseID
1123	06902
1129	06902
1145	06902
1123	06903
1145	06903
1123	06904
1129	06904

Course Number	Student ID
221	1123
221	1129
221	1145
231	1123
231	1145
241	1123
241	1129

A table resulting from a join on Course, Enrollment where dept=CSC and courseIDs match

Views: Typical View Scenarios

- *Joined* views: Subselect uses a JOIN clause
- CREATE VIEW CSCEnroll(CourseNumber,StudentID) AS SELECT number,studentID from Course c, Enrollment e where c.dept='CSC' AND c.courseID = e.courseID ;
- See next page for example relations (tables)

Views: Example

Course

courseID	dept.	number
06902	CSC	221
06903	CSC	231
06904	CSC	241
06905	CSC	191

Enrollment

studentID	courseID
1123	06902
1129	06902
1145	06902
1123	06903
1145	06903
1123	06904
1129	06904

CSCEnroll

CourseNumber	StudentID
221	1123
221	1129
221	1145
231	1123
231	1145
241	1123
241	1129

A table resulting from a join on Course, Enrollment where dept=CSC and courseIDs match

Views: SQL Syntax For Deletion

- DROP VIEW ViewName [RESTRICT | CASCADE]
  - CASCADE says: Also delete all objects that reference the view ViewName (that is, other views built on this view)
  - RESTRICT says: Reject delete of ViewName if other views exist that reference view ViewName

Views: DBMS Implementation

- Remember, views are dynamic tables – they are never “permanently stored” but instead are managed at runtime
  - The definition of the view is stored itself, but not the data
- Two common techniques used by DBMS to support dynamic generation
  - View Resolution
  - View Materialization

Views: View Resolution

- *View Resolution*: At the time a query is made on a view, the query is translated into a query that is applied as a variant of the view's defining query
  - Remember, we can look up the definition
- Translation steps:
  - Convert column names from view names used in user query with column names used in defining query
  - Replace FROM clause from user query with defining query
  - Merge WHERE clauses from user query and defining query
  - Copy GROUPBY/HAVING clauses from defining query
  - Copy ORDER BY from user query

Views: View Resolution

- SELECT StudentID FROM CSCEnroll WHERE CourseNumber='221';

- Translation steps:
  - Convert column names from view names used in user query with column names used in defining query
    - StudentID ➔ studentID, CourseNumber➔ number
  - Replace FROM clause from user query with defining query
    - FROM CSCEnroll ➔ FROM Course c, Enrollment e
  - Merge WHERE clauses from user query and defining query using AND
    - WHERE CourseNumber='221' ➔ WHERE c.number='221' AND c.dept='CSC' AND c.courseID = e.courseID
  - Copy GROUPBY/HAVING clauses from defining query
    - Not relevant for given example
  - Copy ORDER BY from user query
    - Not relevant for given example (could easily imagine ordering by StudentID)

- SELECT studentID FROM Course c, Enrollment e WHERE c.number='221' AND c.dept='CSC' and c.courseID = e.courseID;

Views: Example

Course

courseID	dept.	number
06902	CSC	221
06903	CSC	231
06904	CSC	241
06905	CSC	191

Enrollment

studentID	courseID
1123	06902
1129	06902
1145	06902
1123	06903
1145	06903
1123	06904
1129	06904

CSCEnroll

CourseNumber	StudentID
221	1123
221	1129
221	1145
231	1123
231	1145
241	1123
241	1129

A table resulting from a join on Course, Enrollment where dept=CSC and courseIDs match

Views: View Materialization

- *View Materialization:*
  - Actually store the view after queried for the first time
  - Make queries directly against the materialized relation
  - Employ v*iew maintenance* to keep the view up-to-date as changes are made to the underlying relations that the defining query is based on

Views: Resolution vs. Materialization

A simple way to think about the differences:

Let V be a view defined by a query X on database D.

$V = X(D)$

A query Y on view V is then:

$Y(V) = Y(X(D))$

Materialization is actually generating X(D) and then applying the query Y to the result

Resolution is determining the composition function ZY and then applying that composition function to D

Views: Resolution vs. Materialization

- At query time, materialization is much faster
  - No translation
- View maintenance, however, can be expensive
  - Have to check and see
    - A) whether a given statement can affect the view
    - B) if so, how the view should be updated