Views and Virtual Tables

**Views**

- base tables (CREATE TABLE) stored in database
- views (CREATE VIEW) dependent on base tables or other views, may or may not be stored (virtual vs materialized)
- temporary tables (subquery, etc.) limited lifetime

```
CREATE OR REPLACE VIEW CSstudents AS
SELECT *
FROM student
WHERE Program = 'COMP-SCI';

SELECT *
FROM CSstudents;
```
Point of Views

CREATE VIEW studentview AS
SELECT LastName, FirstName, SID, Career, Program FROM student;

Hide information (grant access to relevant info)
SELECT name FROM studentgroup WHERE name NOT IN (SELECT groupname FROM CSstudents, memberof WHERE StudentID = SID);

Simplify queries (improve readability)
- not necessarily a good reason to create a view in general, if temporary table is sufficient

Point of Views

CREATE VIEW enrollment(SID, LName, CID, CNR, Dpt) AS
SELECT SID, LastName, CID, CourseNr, Department FROM student, enrolled, course WHERE SID = studentID AND CourseID = CID;

SELECT count(*) FROM enrollment WHERE CNR = 440 AND Dpt = 'CSC';

speed up querying

Modifying Views

DROP VIEW Csstudents;

• What about other objects that depend on it (e.g other views)?
• How is/are the underlying base table(s) affected?

INSERT INTO CSstudents(LastName, FirstName, SID) VALUES ('Crackenden', 'Gloria', 123);

What do INSERT, DELETE, UPDATE mean for a view?

Examples: CSstudents, Enrollment
Updatable Views

"An updatable view is one you can use to insert, update, or delete base table rows."

http://download.oracle.com/docs/cd/B28359_01/server.111/b28286/statements_8004.htm

Roughly:

• FROM contains only a single relation
• no DISTINCT, aggregation, set, calculated value
• WHERE clause may not contain a sub-query involving the relation the view is based on

Statement can still fail (e.g. if primary key is missing in INSERT)

Or, you use Triggers

CREATE VIEW enrollment(SID, LName, CID, CNR, Dpt) AS
SELECT SID, LastName, CID, CourseNr, Department
FROM student, enrolled, course
WHERE SID = studentID AND CourseID = CID;

CREATE TRIGGER enrollmentinsert
INSTEAD OF INSERT ON enrollment
FOR EACH ROW
BEGIN
    INSERT INTO enrolled(StudentID, CourseID)
    VALUES (:new.SID, :new.CID);
END;

Trigger can fail for f.k violations: good

Updatable Views: Examples

• Create a trigger that implements INSERTs into studentview
• Create a trigger that implements INSERTs into Cstudents
• Create a trigger that implements DELETEs on enrollment
• Create triggers that implement UPDATEs on enrollment
WITH CHECK OPTION

CREATE OR REPLACE VIEW CSstudents AS
SELECT *
FROM student
WHERE Program = 'COMP-SCI'
WITH CHECK OPTION;

SELECT *
FROM CSstudents;

• what happens if we try inserting non-CS student?

CHECK OPTION for Assertions

CREATE OR REPLACE VIEW v_memberof AS
SELECT StudentID, GroupID, Joined
FROM memberof
WHERE joined >= (SELECT started FROM student
                   WHERE SID = StudentID)
WITH CHECK OPTION;

• if we use v_memberof in place of memberof what does this enforce?
• downside: nesting views deeply is bad, so not always good replacement for base tables

CHECK OPTION Examples

• ensure that undergraduate students do not enroll in graduate courses
• ensure that graduate students do not enroll in more than 3 courses a quarter
• limit the number of courses to at most 100
• limit the number of students each year to at most 50
VIRTUAL TABLES

Temporary Tables

create global temporary table gradstudent(
    LASTNAME VARCHAR2(40),
    SID NUMBER(5,0),
    PROGRAM VARCHAR2(10),
    primary key(sid)
) on commit delete rows;

insert into gradstudent
select lastname, sid, program
from student
where career = 'GRD';

• lifetime of temporary data is limited to session
• table exists beyond session

Common Table Expressions (CTE)

WITH GradStudents AS
(SELECT SID, LastName, SSN
FROM student
WHERE Career = 'GRD')
SELECT *
FROM enrolled
WHERE StudentID NOT IN (SELECT SID FROM GradStudents);

• temporary table, exists only for lifetime of query, cannot be used in other queries
• can create multiple such tables
### CTE Example

```sql
WITH StudentEnrollment(SID, Quarter, Year, enr_crs_ct) AS 
(SELECT StudentID, Quarter, Year, count(CourseID) 
FROM enrolled GROUP BY StudentID, Quarter, Year),
StudentMax(SID, max_enr_crs_ct) AS 
(SELECT SID, max(enr_crs_ct) 
FROM StudentEnrollment 
GROUP BY SID)
SELECT * 
FROM student S, StudentMax SM 
WHERE S.SID = SM.SID;
```

- temporary table can refer to previous temporary table
- mutual recursion not allowed (in Oracle)

### CTE Examples

- List departments in which the average enrollment in courses is below 2
- For each program compute the number of Chicago students in the program but only include programs that have at least three students.