More SQL

CPS 216
Advanced Database Systems
January 30, 2003

Thanks Dr. Jun Yang for providing the slides!

Content

- Outerjoin
- Deferred constraint checking
- Modification of constraints
- Dynamic SQL
- Modifying views

Outerjoin motivation

- Example: a master class list
  - SELECT c.CID, c.title, s.SID, s.name
    FROM Course c, Enroll e, Student s
    WHERE c.CID = e.CID AND e.SID = s.SID;
  - What if a class is empty?
    - It may be reasonable for the master class list to include
      empty classes as well
      - For these classes, SID and name columns would be NULL

Outerjoin flavors and definitions

- A full outerjoin between R and S (denoted $R \bowtie S$) includes all rows in the result of $R \bowtie S$, plus
  - “Dangling” $R$ rows (those that do not join with any $S$ rows) padded with NULL’s for $S$’s columns
  - “Dangling” $S$ rows (those that do not join with any $R$ rows) padded with NULL’s for $R$’s columns
- A left outerjoin ($R \bowtie S$) includes rows in $R \bowtie S$ plus dangling $R$ rows padded with NULL’s
- A right outerjoin ($R \bowtie S$) includes rows in $R \bowtie S$ plus dangling $S$ rows padded with NULL’s

Outerjoin examples

<table>
<thead>
<tr>
<th>Course</th>
<th>Enroll</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPS199</td>
<td>NULL</td>
</tr>
<tr>
<td>CPS196</td>
<td>142</td>
</tr>
<tr>
<td>CPS114</td>
<td>142</td>
</tr>
<tr>
<td>CPS114</td>
<td>123</td>
</tr>
<tr>
<td>CPS114</td>
<td>857</td>
</tr>
<tr>
<td>CPS130</td>
<td>857</td>
</tr>
<tr>
<td>CPS114</td>
<td>456</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enroll</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>142 CPS199</td>
<td>Independent Study</td>
</tr>
<tr>
<td>142 CPS114</td>
<td>Computer Networks</td>
</tr>
<tr>
<td>123 CPS114</td>
<td>Analysis of Algorithms</td>
</tr>
<tr>
<td>857 CPS130</td>
<td>Analysis of Algorithms</td>
</tr>
<tr>
<td>857 CPS196</td>
<td>Computer Networks</td>
</tr>
<tr>
<td>142 CPS114</td>
<td>Computer Networks</td>
</tr>
<tr>
<td>142 CPS196</td>
<td>NULL</td>
</tr>
<tr>
<td>123 CPS114</td>
<td>NULL</td>
</tr>
<tr>
<td>857 CPS130</td>
<td>NULL</td>
</tr>
</tbody>
</table>

Outerjoin syntax

- SELECT *
  FROM Course LEFT OUTER JOIN Enroll
  ON Course.CID = Enroll.CID;
- SELECT *
  FROM Course RIGHT OUTER JOIN Enroll
  ON Course.CID = Enroll.CID;
- SELECT *
  FROM Course FULL OUTER JOIN Enroll
  ON Course.CID = Enroll.CID;

These queries return all columns in Course and Enroll, so they are not exactly $Course \bowtie Enroll$, $Course \bowtie Enroll$, and $Course \bowtie Enroll$, respectively
Deferred constraint checking

- No-chicken-no-egg problem
  - CREATE TABLE Dept
    (name CHAR(20) NOT NULL PRIMARY KEY,
    chair CHAR(30) NOT NULL REFERENCES Prof(name));
  - CREATE TABLE Prof
    (name CHAR(30) NOT NULL PRIMARY KEY,
    dept CHAR(20) NOT NULL REFERENCES Dept(name));
  - The first INSERT will always violate a constraint
- Deferred constraint checking is necessary
  - Check only at the end of a transaction
  - Allowed in SQL as an option
- Wait… How can create the schema in the first place?

Modification of Constraints

- First, give names to contraints
  - In table MovieStar
    - Name CHAR(30) PRIMARY KEY
    - Name CHAR(30) CONSTRAINT NameIsKey PRIMARY KEY
- Alter constraints on tables
  - Drop a constraint
    - ALTER TABLE MovieStar DROP CONSTRAINT NameIsKey
  - Reinstate that constraint
    - ALTER TABLE MovieStar ADD CONSTRAINT NameIsKey
  - Alter assertions
    - DROP ASSERTION assertion_name

Dynamic SQL

- Why need dynamic SQL?
  - A spreadsheet that needs to access data from DBMS
  - Queries are generated by users
  - We do not know in advance about the query.
- How does SQL deal with this problem?
  - Dynamic SQL
  - Two main commands:
    - PREPARE – parse string and compile it as an SQL command
    - EXECUTE – launch the query

Dynamic SQL example

- A simple example
  - Char c_sqlstring[] = "DELETE FROM Student WHERE gpa < 2.0";
  - EXEC SQL PREPARE readytogo FROM :c_sqlstring;
  - EXEC SQL EXECUTE readytogo;
- Run-time overhead
  - Prepare a dynamic SQL at run-time requires run-time overhead
  - Many more about SQL…
    - Check SQL reference book for more detail

Modifying views

- Does not seem to make sense since views are virtual
- But does make sense if that is how users see the database
- Goal: modify the base tables such that the modification would appear to have been accomplished on the view

A simple case

CREATE VIEW StudentGPA AS
  SELECT SID, GPA FROM Student;
DELETE FROM StudentGPA WHERE SID = 123;
translates to:
DELETE FROM Student WHERE SID = 123;
An impossible case

CREATE VIEW HighGPAStudent AS
SELECT SID, GPA FROM Student
WHERE GPA > 3.7;
INSERT INTO HighGPAStudent
VALUES(987, 2.5);

No matter what you do on Student, the inserted row will not be in HighGPAStudent

A case with too many possibilities

CREATE VIEW AverageGPA(GPA) AS
SELECT AVG(GPA) FROM Student;

Note that you can rename columns in view definition

UPDATE AverageGPA SET GPA = 2.5;

Set everybody’s GPA to 2.5?
Adjust everybody’s GPA by the same amount?
Just lower Bart’s GPA?

SQL92 updateable views

- Single-table SFW
  - No aggregation
  - No subqueries

- Overly restrictive
- Still might get it wrong in some cases
  - See the slide titled “An impossible case”
  - For more, see SQL reference book…

Projects and Homeworks

- Project ideas
  - Background reference pointers have been posted on newsgroup.
    - Email me if you need it but do not have access to the newsgroup
  - You are highly welcomed to propose your own ideas!

- Homework One
  - Reminder: HW#1 due in 10 days!