SQL Joins, Queries and Views

Chapter 4 – Manga Guide to Databases

Today

- Outer Joins
- Sub-queries (Correlated and Non-correlated)
- SQL Views

JOIN ON Syntax

List the students and the courses they are enrolled in

<table>
<thead>
<tr>
<th>Students</th>
<th>Enrolled</th>
</tr>
</thead>
<tbody>
<tr>
<td>S.Nb</td>
<td>SName</td>
</tr>
<tr>
<td>100</td>
<td>Doe</td>
</tr>
<tr>
<td>373</td>
<td>Doe</td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
</tr>
</tbody>
</table>

SELECT S.Nb, SName, E.Cid FROM Students S, Enrolled E WHERE S.Nb = E.SNb

Only enrolled students listed
Outer Joins

List all students and the courses they are enrolled in:

```sql
SELECT S.SNb, SName, E.Cid
FROM Students S
LEFT JOIN Enrolled E
ON S.Snb=E.Snb;
```

### Students

<table>
<thead>
<tr>
<th>SNb</th>
<th>SName</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Doe</td>
<td><a href="mailto:jdoe2@usna.edu">jdoe2@usna.edu</a></td>
</tr>
<tr>
<td>373</td>
<td>Doe</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
</tr>
<tr>
<td>312</td>
<td>Doe</td>
<td><a href="mailto:jdoe@usna.edu">jdoe@usna.edu</a></td>
</tr>
</tbody>
</table>

### Enrolled

<table>
<thead>
<tr>
<th>SNb</th>
<th>Cid</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>IT430</td>
<td>Fall2008</td>
</tr>
<tr>
<td>312</td>
<td>IT430</td>
<td>Fall2008</td>
</tr>
</tbody>
</table>

---

Sub-Queries (with No Correlation)

Find names of sailors who have reserved boat #103:

```sql
SELECT S.sname
FROM Sailors S
WHERE S.sid IN (SELECT R.sid
FROM Reserves R
WHERE R.bid=103);
```

---

Sub-Queries (with Correlation)

Find names of sailors who have reserved boat #103:

```sql
SELECT S.sname
FROM Sailors S
WHERE EXISTS (SELECT *
FROM Reserves R
WHERE R.bid=103 AND S.sid=R.sid);
```
Sub-Queries (with Correlation)

Find names of sailors who have not reserved boat #103:

```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS (SELECT * FROM Reserves R
WHERE R.bid=103 AND S.sid=R.sid)
```

Double NOT EXISTS

- The following code determines the name of any ARTIST that is of interest to every CUSTOMER:

```
SELECT A.Name
FROM ARTIST AS A
WHERE NOT EXISTS
(SELECT C.CustomerID
FROM CUSTOMER C
WHERE NOT EXISTS
(SELECT CI.CustomerID
FROM CUSTOMER_artist_int CI
WHERE C.CustomerID = CI.CustomerID
AND A.ArtistID = CI.ArtistID));
```

SQL Views

- SQL view is a virtual table that is constructed from other tables or views
- It has no data of its own, but obtains data from tables or other views
- It only has a definition

- SELECT statements are used to define views
  - A view definition may not include an ORDER BY clause
  - Views can be used as regular tables in SELECT statements
CREATE VIEW Command

- CREATE VIEW command:
  ```sql
  CREATE VIEW view_name
  AS
  select_statement
  ```

- Use the view:
  - In SELECT statements
  - Sometimes in INSERT statements
  - Sometimes in UPDATE statements
  - Sometimes in DELETE statements

CREATE VIEW Command

- CREATE VIEW command:
  ```sql
  CREATE VIEW CustomerNameView
  AS
  SELECT CustName AS CustomerName
  FROM CUSTOMER;
  ```

- To use the view:
  ```sql
  SELECT *
  FROM CustomerNameView
  ORDER BY CustomerName;
  ```

Uses for SQL Views

- Security: hide columns and rows
- Display results of computations
- Hide complicated SQL syntax
- Provide a level of isolation between actual data and the user’s view of data
  - three-tier architecture
- Assign different processing permissions to different views on same table
Security: hide columns and rows

- MIDS database, Midshipmen table
  - View for faculty – all mids with IT major
  - View for students – all mids, no grades
- Midshipmen (Alpha, Name, DateOfBirth, GPA, Major)
- Exercise: Write the SQL to create the views
- SELECT, INSERT, UPDATE, DELETE?

Display results of computations

- Faculty (EmpID, LName, FName, Department, AreaCode, LocalPhone)
- Create a view to display 2 columns:
  - Name = FName LName
  - Phone = (AreaCode) LocalPhone
- SELECT, INSERT, UPDATE, DELETE?

Hide complicated SQL syntax

- Mid(Alpha, LName, FName, Class, Age)
- Course(CourseID, Description, Textbook)
- Enroll(Alpha, CourseID, Semester, Grade)
- Create a view to display the student alpha, name, CourseID and description of courses they are/were enrolled
- SELECT, INSERT, UPDATE, DELETE?
Provide a level of isolation between actual data and application

- CREATE VIEW CustomerV AS
  SELECT *
  FROM Customers
- Applications use CustomerV
- Can change the underlying table without changing the application

ALTER VIEW CustomerV AS
SELECT *
FROM New_Customers

Updating Views

- CREATE VIEW CustomerV AS
  SELECT *
  FROM Customers
- SELECT, INSERT, DELETE, UPDATE?
- CREATE VIEW FacultyPhone AS
  SELECT FName + ' ' + LName AS Name,
  '( ' + AreaCode + ' )' + LocalPhone AS Phone
  FROM Faculty
  UPDATE FacultyPhone
  SET Phone = '(410)-293-6822'
  WHERE Name='Adina Crainiceanu'

Updateable Views

- Views based on a single table
  - No computed columns
  - All non-null columns present in view

- Views based on a single table, primary key in view, some non-null columns missing from view
  - Updates for non-computed columns ok
  - Deletes ok
  - Inserts not ok
### Summary – SQL Views

```sql
CREATE VIEW view_name
AS
select_statement
```

- **Virtual table**
  - It only has a definition
  - Data is computed at run-time from base tables
- All views can be used in SELECT
- Some views can be used in INSERT, DELETE, UPDATE