1.264 Lecture 9

SQL: Joins, subqueries, views
Joins

- Relational model permits you to bring data from separate tables into new and unanticipated relationships.
- Relationships become explicit when data is manipulated: when you query the database, not when you create it.
  - This is critical; it allows extensibility in databases. The EPA never thought its data would be used in 1.264 along with DOT carrier data, and some new order tables.
  - You can join on any columns in tables, as long as data types match and the operation makes sense. They don’t need to be keys, though they usually are.
- Good joins
  - Join column is usually key column:
    - Either primary key or foreign key
  - Join columns must have compatible data types
  - Nulls will never join
Joins

- List all orders, showing order number and amount, and name and credit limit of customer
  - Orders has order number and amount, but no customer names or credit limits
  - Customers has customer names and credit limit, but no order info

- SELECT OrderNbr, Amt, Company, CreditLimit FROM Customers, Orders WHERE Cust = CustNbr;  
  (Implicit syntax)

- SELECT OrderNbr, Amt, Company, CreditLimit FROM Customers INNER JOIN Orders ON Customers.CustNbr = Orders.Cust;  
  (SQL-92)

<table>
<thead>
<tr>
<th>OrderNbr</th>
<th>Cust</th>
<th>Prod</th>
<th>Qty</th>
<th>Amt</th>
<th>Disc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>211</td>
<td>Bulldozer</td>
<td>7</td>
<td>$31,000.00</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>522</td>
<td>Riveter</td>
<td>2</td>
<td>$4,000.00</td>
<td>0.3</td>
</tr>
<tr>
<td>3</td>
<td>522</td>
<td>Crane</td>
<td>1</td>
<td>$500,000.00</td>
<td>0.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CustNbr</th>
<th>Company</th>
<th>CustRep</th>
<th>CreditLimit</th>
</tr>
</thead>
<tbody>
<tr>
<td>211</td>
<td>Connor Co</td>
<td>89</td>
<td>$50,000.00</td>
</tr>
<tr>
<td>522</td>
<td>Amaratunga Enterprises</td>
<td>89</td>
<td>$40,000.00</td>
</tr>
<tr>
<td>890</td>
<td>Feni Fabricators</td>
<td>53</td>
<td>$1,000,000.00</td>
</tr>
</tbody>
</table>
Join with 3 tables

- List orders over $25,000, including the name of the salesperson who took the order and the name of the customer who placed it.
  - SELECT OrderNbr, Amt, Company, Name FROM Orders, Customers, SalesReps WHERE Cust = CustNbr AND CustRep = RepNbr AND Amt >= 25000;

<table>
<thead>
<tr>
<th>OrderNbr</th>
<th>Cust</th>
<th>Prod</th>
<th>Qty</th>
<th>Amt</th>
<th>Disc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>211</td>
<td>Bulldozer</td>
<td>7</td>
<td>$31,000.00</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>522</td>
<td>Riveter</td>
<td>2</td>
<td>$4,000.00</td>
<td>0.3</td>
</tr>
<tr>
<td>3</td>
<td>522</td>
<td>Crane</td>
<td>1</td>
<td>$500,000.00</td>
<td>0.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CustNbr</th>
<th>Company</th>
<th>CustRep</th>
<th>CreditLimit</th>
</tr>
</thead>
<tbody>
<tr>
<td>211</td>
<td>Connor Co</td>
<td>89</td>
<td>$50,000.00</td>
</tr>
<tr>
<td>522</td>
<td>Amaratunga Enterprises</td>
<td>89</td>
<td>$40,000.00</td>
</tr>
<tr>
<td>890</td>
<td>Feni Fabricators</td>
<td>53</td>
<td>$1,000,000.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RepNbr</th>
<th>Name</th>
<th>RepOffice</th>
<th>Quota</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>Bill Smith</td>
<td>1</td>
<td>$100,000.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>89</td>
<td>Jen Jones</td>
<td>2</td>
<td>$50,000.00</td>
<td>$130,000.00</td>
</tr>
</tbody>
</table>

Result:

<table>
<thead>
<tr>
<th>OrderNbr</th>
<th>Amt</th>
<th>Company</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$31,000.00</td>
<td>Connor Co</td>
<td>Jen Jones</td>
</tr>
<tr>
<td>3</td>
<td>$500,000.00</td>
<td>Amaratunga Enterprise</td>
<td>Jen Jones</td>
</tr>
</tbody>
</table>
Join notes

- SQL-92 syntax for previous example:
  - SELECT OrderNbr, Amt, Company, Name FROM SalesReps
    INNER JOIN Customers ON SalesReps.RepNbr = Customers.CustRep
    INNER JOIN Orders ON Customers.CustNbr = Orders.Cust
    WHERE Amt >= 25000;

- Use * carefully in joins
  - It gives all columns from all tables being joined

- If a field has the same name in the tables being joined, qualify the field name:
  - Use table1.fieldname, table2.fieldname
  - Customers.CustNbr, Orders.Amt, etc.
Self joins

<table>
<thead>
<tr>
<th>EmpNbr</th>
<th>Name</th>
<th>Title</th>
<th>Mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>105</td>
<td>Mary Smith</td>
<td>Analyst</td>
<td>104</td>
</tr>
<tr>
<td>109</td>
<td>Jill Jones</td>
<td>Sr Analyst</td>
<td>107</td>
</tr>
<tr>
<td>104</td>
<td>Sally Silver</td>
<td>Manager</td>
<td>111</td>
</tr>
<tr>
<td>107</td>
<td>Pat Brown</td>
<td>Manager</td>
<td>111</td>
</tr>
<tr>
<td>111</td>
<td>Eileen Howe</td>
<td>President</td>
<td></td>
</tr>
</tbody>
</table>

- We want to list the analysts and their managers
  - Manager could be foreign key into manager table, but it has to be a ‘foreign’ key into the employee table itself in this case

- Attempt 1:
  - SELECT Name, Name FROM Employee, Employee WHERE Mgr = EmpNbr; (Implicit syntax)
  - Fails because it references Employee table twice
  - Removing 2nd reference also fails; query looks for rows where person is her own manager, which is not what we want.
Self joins

<table>
<thead>
<tr>
<th>EmpNbr</th>
<th>Name</th>
<th>Title</th>
<th>Mgr</th>
</tr>
</thead>
<tbody>
<tr>
<td>105</td>
<td>Mary Smith</td>
<td>Analyst</td>
<td>104</td>
</tr>
<tr>
<td>109</td>
<td>Jill Jones</td>
<td>Sr Analyst</td>
<td>107</td>
</tr>
<tr>
<td>104</td>
<td>Sally Silver</td>
<td>Manager</td>
<td>111</td>
</tr>
<tr>
<td>107</td>
<td>Pat Brown</td>
<td>Manager</td>
<td>111</td>
</tr>
<tr>
<td>111</td>
<td>Eileen Howe</td>
<td>President</td>
<td></td>
</tr>
</tbody>
</table>

- Attempt 2: Pretend there are 2 copies of Employee table, once named Emp, the other named Mgr:
  - SELECT Emp.Name, Mgr.Name FROM Emp, Mgr WHERE Emp.Mgr = Mgr.EmpNbr;  
    (Implicit syntax)
- SQL essentially lets us do this by giving aliases. Valid:
  - SELECT Emp.Name, Mgr.Name FROM Employee Emp, Employee Mgr WHERE Emp.Mgr = Mgr.EmpNbr  
    (Implicit syntax)
  - SELECT Emp.Name, Mgr.Name FROM Employee AS Emp INNER JOIN Employee AS Mgr ON Emp.Mgr = Mgr.EmpNbr  
    (SQL-92)
  - We actually only need to use 1 alias (Mgr)
JOIN types

- **INNER join**: returns just rows with matching keys (join column values)
- **RIGHT join**: returns all rows from right (second) table, whether they match a row in the first table or not
- **LEFT join**: returns all rows from left (first) table, whether they match a row in the first table or not
- **OUTER join**: Returns all rows from both tables, whether they match or not
- *(We’ll do an exercise on these)*
Exercises

• List customer names whose credit limit is greater than their sales rep’s quota. Also list the credit limit and quota.
• List each rep’s name and phone number
Exercises

• List customer names whose credit limit is greater than their sales rep’s quota. Also list the credit limit and quota.
  – SELECT CreditLimit, Quota, Company FROM SalesReps INNER JOIN Customers ON SalesReps.RepNbr = Customers.CustRep WHERE CreditLimit>Quota;

• List each rep’s name and phone number
  – SELECT Name, Phone FROM Offices INNER JOIN SalesReps ON Offices.OfficeNbr = SalesReps.RepOffice;
Subqueries

- SQL subqueries let you use the results of one query as part of another query. Subqueries
  - Are often natural ways of writing a statement
  - Let you break a query into pieces and assemble it
  - Allow some queries that otherwise can’t be constructed
Subqueries

- List the offices where the sales quota [target] for the office exceeds the sum of individual salespersons’ quotas
  - SELECT City FROM Offices WHERE Target > ???
  - ??? is “the sum of the quotas of the salespeople”, or
    - SELECT SUM(Quota) FROM SalesReps WHERE RepOffice = OfficeNbr
- We combine these to get
  - SELECT City FROM Offices WHERE Target > (SELECT SUM(Quota) FROM SalesReps WHERE RepOffice = OfficeNbr);
Subqueries

- Subqueries always appear as part of the WHERE (or HAVING) clause
- Subquery can only produce a single column of data as its result
  - Only one field can be in the subquery SELECT
- ORDER BY is not allowed; it would not make sense
- Usually refer to name of a main table column in the subquery
  - This defines the current row of the main table for which the subquery is being run. This is called an outer reference.
  - In our example, it’s RepOffice= OfficeNbr from Offices table
Views

- Virtual tables that present data in denormalized form to users
- They are NOT separate copies of the data; they reference the data in the underlying tables
- Database stores definition of view; the data is updated when the underlying tables are updated
- Advantages:
  - Designed to meet specific needs of specific users
  - Much simpler queries for users on views constructed for them
  - Security: give access only for data in views
  - Independence: layers user or program away from change in underlying tables
Views

- **CREATE VIEW CustomerOrders AS**
  
  ```
  SELECT CustNbr, Company, Name, OrderNbr, Prod, Qty, Amt
  FROM Customers, SalesReps, Orders
  WHERE CustRep = RepNbr AND CustNbr = Cust
  (Implicit syntax)
  ```

- **Orders**

<table>
<thead>
<tr>
<th>OrderNbr</th>
<th>Cust</th>
<th>Prod</th>
<th>Qty</th>
<th>Amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>88</td>
<td>2</td>
<td>C</td>
<td>7</td>
<td>$31,000</td>
</tr>
<tr>
<td>99</td>
<td>522</td>
<td>CDE</td>
<td></td>
<td>$4,000</td>
</tr>
</tbody>
</table>

- **Customers**

<table>
<thead>
<tr>
<th>CustNbr</th>
<th>Company</th>
<th>CustRep</th>
<th>CreditLimit</th>
</tr>
</thead>
<tbody>
<tr>
<td>211</td>
<td>QGG Co</td>
<td>89</td>
<td>$50,000</td>
</tr>
<tr>
<td>322</td>
<td>DBO C</td>
<td>89</td>
<td>$40,000</td>
</tr>
</tbody>
</table>

- **CustomerOrders**

<table>
<thead>
<tr>
<th>CustNbr</th>
<th>Company</th>
<th>Name</th>
<th>OrderNbr</th>
<th>Prod</th>
<th>Qty</th>
<th>Amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>211</td>
<td>Co</td>
<td>Jen Smith</td>
<td>88</td>
<td>ABAC</td>
<td>7</td>
<td>$31,000</td>
</tr>
<tr>
<td>322</td>
<td>DBO Co</td>
<td>Jen Smith</td>
<td>99</td>
<td>CDE</td>
<td>2</td>
<td>$4,000</td>
</tr>
</tbody>
</table>

- **SalesR**

<table>
<thead>
<tr>
<th>RepNbr</th>
<th>Name</th>
<th>RepOffice</th>
<th>Quota</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>Bill Smith</td>
<td>22</td>
<td>$100,000</td>
<td>$0</td>
</tr>
<tr>
<td>89</td>
<td>Jen Smith</td>
<td>44</td>
<td>$50,000</td>
<td>$130,000</td>
</tr>
</tbody>
</table>
View subtleties

• Possible to change views to invalidate them
  – E.g. View of books under $5
  – What happens if you update the price of a book to $5.99 through the view. It disappears!
  – Prevent this by adding: WITH CHECK OPTION

• Not all views can be updated. View is read-only if:
  – DISTINCT is in the SELECT statement
  – Expressions (averages, totals, etc.)
  – References to views that are not updatable
  – GROUP BY or HAVING clauses
  – In bad databases: References to more than one table (defeats purpose)

• You will use views in Dreamweaver
  – It’s easier for a Web or XML page to have a single source of its data
Exercises

• Display all customers with orders > $50,000 or credit limits > $50,000.
  – Hint: You need to use a RIGHT or LEFT JOIN since you want all the customers, whether they have an order or not, to be the ‘raw material’ for the WHERE clause

• Delete reps in sales offices in New York (NY) with quotas over $40,000
  – Hint: Remember you have to delete FROM a single table. Use a subquery.
  – Delete reps who are IN the result of the subquery
  – When you have this right, it will not delete the reps due to referential integrity, but you’ll know it’s working from that message.
Exercises

- Display all customers with orders or credit limits > $50,000.
  - `SELECT DISTINCT CustNbr
    FROM Customers LEFT JOIN Orders ON CustNbr = Cust
    WHERE (CreditLimit > 50000 OR Amt > 50000)`

- Delete reps in sales offices in New York (NY) with quotas over $40,000
  - `DELETE FROM SalesReps
    WHERE RepNbr IN
      (SELECT RepNbr
       FROM SalesReps, Offices
       WHERE OfficeNbr = RepOffice AND
         Quota>40000 AND State='NY');`

- (Syntax is correct, but won’t execute due to referential integrity in our sample database)