SQL, Single-table, Multi-table, Joining, Aliasing

CGT 356
Web Programming, Development, & Database Integration
Lecture 7
Review Single Table SELECT

- What is the ID of the employee whose name is Bob Jenkins?

  SELECT EmployeeID
  FROM Employee
  WHERE LastName = 'Jenkins'
  AND FirstName = 'Bob';
SELECTing Multiple Fields

- List the Names and ID’s of the employees who live in Montana.

- SELECT LastName, FirstName, EmployeeID FROM Employee
  WHERE State = ‘MT’
  ORDER BY LastName ASC;
ORDER BY

- ASC
  - Ascending

- DESC
  - Descending

- Orders the data returned in either ascending or descending order.
From Last Time

- List the names of the employees whose salary is greater than $40,000

  SELECT LastName
  FROM Employee
  WHERE Salary > 40000;
From Last Time

- List all of the names of the employees whose employee ID is not 125775

  SELECT LastName
  FROM Employee
  WHERE EmployeeID <> 125775;
Multiple Table SELECT

One to Many Relationship

• An employee can have many orders
• An order can belong to only one employee
Primary & Foreign

- OrderID is the Primary Key of the Order table
  - It uniquely identifies a record in the Order table

- What are EmployeeID & CustomerID in the Order table?
  - Foreign Keys
Foreign Keys

- A Foreign Key is a field in a table that references the Primary Key of another table
  - EmployeeID in Order is a Foreign Key that references the Primary Key (EmployeeID) in Employee

- Now we know two Keys
  - Primary: Unique & Not Null
  - Foreign: References PK in another table
Relational Databases

- An arrangement of tables that store all of the data
- The tables are “related” to one another
  - Like Order has an EmployeeID
  - For this reason tables are often called Relations
- Not all databases are relational
  - All databases in this class ARE relational
Entity Integrity Constraint

- **Definition:**
  - No Primary Key value can be null

- **Why?**
  - Because a Primary Key is used to identify individual records in a relation (table). Without the entity integrity constraint, we would not be able to find records.
Referential Integrity Constraint

- Specified between two relations (tables).
- Used to maintain consistency among records of the two relations.
- Referential integrity constraints are placed on Foreign Keys (but can be used on other fields).
Entity vs. Referential

Entity Integrity Constraint placed on EmployeeID: **Must not be null.**

As a result: The Foreign Key is also Not Null.

Referential Integrity Constraint placed on EmployeeID. Order.EmployeeID must equal Employee.EmployeeID

If they are not equal, then the referential integrity constraint has been violated and the Foreign Key is invalid.
Multiple Table SELECT

- Start with a single table & build up…
- List the OrderID’s of the orders placed by the employee with employeeID of 125775

  SELECT OrderID
  FROM Order
  WHERE EmployeeID = 125775;
Multiple Table SELECT (cont.)

- Now from two tables...
- List the OrderID’s of the orders placed by the employee with the name Bob Jenkins

- SELECT OrderID
  FROM Order, Employee
  WHERE Order.EmployeeID = Employee.EmployeeID
  AND LastName = 'Jenkins'
  AND FirstName = 'Bob';
SELECT Syntax

- Several things happening here:
  - Notice the AND clause
    - Use AND whenever you need more than one condition in the WHERE clause
      - WHERE City='London' AND Salary > 50000
  - Also
    - Joining tables
      - Order.EmployeeID = Employee.EmployeeID
Joining Tables

- Types of Joins (very briefly)
  - Outer Join
    - All records from both tables
  - Inner Join
    - Only matching records from both tables
  - Left Join
    - All records from left table, only matching from right
  - Right Join
    - All records from right table, only matching from left
Joining Tables (cont.)

- Dot notation – using the dot operator (.)
  - Order.EmployeeID = Employee.EmployeeID
    - Means: the EmployeeID field in the Order table must equal the EmployeeID field in the Employee table.
  - Why is it used?
    - To reduce ambiguity
    - EmployeeID = EmployeeID would not make sense
  - This is the type of join we will use in this class
Aliasing

- The same SQL query as before, this time with aliasing
  
  ```sql
  SELECT OrderID
  FROM Order O, Employee E
  WHERE O.EmployeeID = E.EmployeeID
  AND LastName = 'Jenkins'
  AND FirstName = 'Bob'
  ORDER BY OrderID ASC;
  ```
Instead of writing out the name of the relation (table) all of the time, we shorten it by giving it an alias.

- **Alias**
  - An abbreviation for a table name that is declared in the FROM clause.
Aliasing (cont.)

- SELECT O.EmployeeID, OrderID, OrderDate FROM Order O, Employee E WHERE O.EmployeeID = E.EmployeeID AND LastName = 'Jenkins' AND FirstName = 'Bob' ORDER BY OrderID, OrderDate ASC;

- Even though the alias is declared in the FROM clause, you MUST use it wherever there is ambiguity in field names. Here it is used in the SELECT clause.

- Order by multiple fields: orders first by OrderID, then by OrderDate.