SQL

• Structured query language (SQL) used for
  – Data definition (DDL): tables and views (virtual tables). These are the basic operations to convert a data model to a database
  – Data manipulation (DML): user or program can INSERT, DELETE, UPDATE or retrieve (SELECT) data.
  – Data integrity: referential integrity and transactions. Enforces keys.
  – Access control: security
  – Data sharing: by concurrent users
• **Not a complete language like Java**
  – SQL is sub-language of about 30 statements
• **Nonprocedural language**
  – No branching or iteration
  – Declare the desired result, and SQL provides it
**SQL SELECT**

- SELECT constructed of clauses to get columns and rows from one or more tables or views. Clauses must be in order:
  - SELECT columns/attributes
  - INTO new table
  - FROM table or view
  - WHERE specific rows or a join is created
  - GROUP BY grouping conditions (columns)
  - HAVING group-property (specific rows)
  - ORDER BY ordering criterion  ASC | DESC

---

### Example tables

<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>AmaratungaEnterprise</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>

<table>
<thead>
<tr>
<th>OfficeNbr</th>
<th>City</th>
<th>State</th>
<th>Region</th>
<th>Target</th>
<th>Sales</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Denver</td>
<td>CO</td>
<td>West</td>
<td>$3,000,000.00</td>
<td>$130,000.00</td>
<td>970.586.3341</td>
</tr>
<tr>
<td>2</td>
<td>New York</td>
<td>NY</td>
<td>East</td>
<td>$200,000.00</td>
<td>$300,000.00</td>
<td>212.942.5574</td>
</tr>
<tr>
<td>57</td>
<td>Dallas</td>
<td>TX</td>
<td>West</td>
<td>$0.00</td>
<td>$0.00</td>
<td>214.781.5342</td>
</tr>
</tbody>
</table>
Using SQL Server and Management Studio Express

- Your SQL Server database engine should start by default when your system starts
  - Ask TA for help if needed at office hours
- Start Management Studio Express (MSE) from Start->Programs->SQL Server 2008
- Open Lecture3CreateDB.sql with MSE in Windows Explorer
  - Download the .sql file from Stellar and double-click on it
- Select ‘Execute’ from toolbar
  - Database MIT1204 will be created and data inserted for examples during this class
- Open Lecture3Examples.sql for all SQL code in this lecture
  - Experiment with it
**SQL queries: SELECT**

- Click ‘New Query’ in MSE; type these statements:
  - List the sales reps
    - SELECT Name, Sales, Quota FROM SalesReps
  - Find the amount each rep is over or under quota
    - SELECT Name, Sales, Quota, (Sales-Quota) FROM SalesReps
  - Find the slackers
    - SELECT Name, Sales, Quota, (Sales-Quota) FROM SalesReps WHERE Sales < Quota

<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>

**SQL queries: calculation, insert, delete, update**

- Find the average sale
  - SELECT AVG(Amt) FROM Orders;
- Find the average sale for a customer
  - SELECT AVG(Amt) FROM Orders WHERE Cust = 211;
- Add an office
  - INSERT INTO Offices (OfficeNbr, City, State, Region, Target, Sales, Phone) VALUES ('55', 'Dallas','TX','West', 200000, 0, '214.333.2222');
- Delete a customer
  - DELETE FROM Customers WHERE Company = ‘Connor Co’;
  - (Syntax is valid but command will fail due to referential integrity)
- Raise a credit limit
  - UPDATE Customers
    - SET CreditLimit = 75000 WHERE Company = ‘Amaratunga Enterprises’;
SELECT: * and duplicates

• Select all columns (fields)
  – SELECT * FROM Offices;
• Duplicate rows: query will get two instances of ‘West’
  – SELECT Region FROM Offices;
• Eliminate duplicates:
  – SELECT DISTINCT Region FROM Offices;

NULLs

• NULL values evaluate to NOT TRUE in all cases.
  – Insert ‘NewRep’ with NULL (blank or empty) Quota
• The following two queries will not give all sales reps:
  – SELECT Name FROM SalesReps WHERE Sales > Quota;
  – SELECT Name FROM SalesReps WHERE Sales <= Quota;
  – A new rep with a NULL quota will not appear in either list
• Check for NULLS by:
  – SELECT Name FROM SalesReps WHERE Quota IS NULL;
SELECT Operators

- SELECT * FROM <table>
  - WHERE Disc*Amt > 50000;  
    (Orders)
  - WHERE Quota BETWEEN 50000 AND 100000;  
    (SalesReps)
    - Range is inclusive (>=50000 and <=100000)
  - WHERE State IN ('CO', 'UT', 'TX');  
    (Offices)
  - WHERE RepNbr IS NOT NULL;  
    (SalesReps)
  - WHERE Phone NOT LIKE '21%';  
    (Offices)

- SQL standard only has 2 wildcards
  - % any string of zero or more characters (* in Access)
  - _ any single character (? in Access)

- Most databases have additional/different wildcards.
  SQL Server has:
  - [list] match any single character in list, e.g., [a-f]
  - [^[list]] match any single character not in list, e.g. [^h-m]

SELECT: COUNT, GROUP BY

<table>
<thead>
<tr>
<th>PartID</th>
<th>Vendor</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>A</td>
</tr>
<tr>
<td>234</td>
<td>A</td>
</tr>
<tr>
<td>345</td>
<td>B</td>
</tr>
<tr>
<td>362</td>
<td>A</td>
</tr>
<tr>
<td>2345</td>
<td>C</td>
</tr>
<tr>
<td>3464</td>
<td>A</td>
</tr>
<tr>
<td>4533</td>
<td>C</td>
</tr>
</tbody>
</table>

- Number of parts from vendor A
  - SELECT COUNT(*) FROM Parts WHERE Vendor = 'A';
  - Result: 4

- Number of parts from each vendor
  - SELECT Vendor, COUNT(*) AS PartsCount FROM Parts GROUP BY Vendor;
  - Result:
    | Vendor | PartsCount |
    |--------|------------|
    | A      | 4          |
    | B      | 1          |
    | C      | 2          |
Examples

• What is the average credit limit of customers whose credit limit is less than $1,000,000?
  – SELECT AVG(CreditLimit) FROM Customers WHERE CreditLimit < 1000000;
• How many sales offices are in the West region?
  – SELECT Count(*) FROM Offices WHERE Region= 'West';
• Increase the price of bulldozers by 30% in all orders
  – UPDATE Orders SET Amt= Amt*1.3 WHERE Prod= 'Bulldozer';
• Delete any sales rep with a NULL quota
  – DELETE FROM SalesReps WHERE Quota IS NULL;

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.
  – 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 columns must have compatible data types
  – Join column is usually key column:
    • Either primary key or foreign key
  – 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
- \( \text{SELECT OrderNbr, Amt, Company, CreditLimit FROM Customers, Orders WHERE Cust = CustNbr;} \) (Implicit syntax)
- \( \text{SELECT OrderNbr, Amt, Company, CreditLimit FROM Customers JOIN Orders ON Customers.CustNbr = Orders.Cust;} \) (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>


custNbr | company | custRep | creditLimit |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></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.
  - \( \text{SELECT OrderNbr, Amt, Company, Name FROM Orders, Customers, SalesReps WHERE Cust = CustNbr AND CustRep = RepNbr AND Amt >= 25000;} \) (Implicit syntax)

<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>

custNbr | company | custRep | creditLimit |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></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>

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 Enterprises</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.
- You can join a table to itself (self-join). See text.

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 second table or not
- OUTER join: Returns all rows from both tables, whether they match or not
Examples

- 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;

- 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)