Outline

• What is a database?

• What is a database management system?

• An Introduction to Microsoft Access
  – How to create a database
  – How to retrieve data from a database
  – How to build a nice Graphical User Interface on top of a database
Why are we learning this?

- Databases are perhaps the single most important class of corporate applications.

- Databases are surprisingly powerful data modeling and analysis tools in situations where spreadsheets fall short.
  - Students who plan to work in management consulting will soon find this out.

- MS Access is a great example of how easy it is to build powerful applications without the need of a background in technology.
What is a database

- **Boring answer**
  - A structured collection of data
  - Example: A telephone directory

- **Insightful answer**
  - A data-centered mirror of an organization’s business processes
    - Structure of data reflects organizational processes
    - Content of data reflects organization’s history
Example: Northwind Traders
Representing the Real World as Data
What Data Are Businesses Interested In?

- **Entity**
  - a person, place, thing, or event on which we maintain information
  - Examples: Employees, Customers, Products, Warehouses

- **Attribute**
  - characteristic or quality of particular entity
  - Examples: Employee’s SSN, Customer’s Address, Product’s Unit Price

- **Relationships Among Entities**
  - Examples:
    » Customer - *Orders* - Product(s)
    » Order - *Serviced by* - Employee
From Spreadsheets to Databases

- Spreadsheets are great for keeping track of data for one type of entities
  - Participants of a conference
  - Students of a class
  - Customers of a company
  - ....
What is the basic spreadsheet “data model”?

- Each row stores data about one entity
- Each column stores data about an attribute
- Each cell stores data about an attribute of an entity
Spreadsheet limitations

• Things get complicated when we want to keep track of several inter-related entities

• For example:
  – Customers
  – Products
  – Orders

• Let’s try it!
Spreadsheets are awkward for storing relationships

- Main difficulty is that an “Order” is essentially a relationship between one Customer and one or more Products
The reason we build databases is in order to easily retrieve information to answer questions that support managerial decision-making.

For example:

*Who are our top 10 customers based on their total order value in the year 2002?*

Can you do this using a spreadsheet?
Enter Relational Databases

- A relational DB supports storage of data as a set of inter-related tables
  - Each table stores data about a set of Entities
  - Each table row is a record about one such Entity
  - Each record column is a field specifying an attribute of this Entity
  - Each record has a field that acts as a unique identifier of an entity
  - Relationships among entities are specified by referring to this unique identifier from other tables
<table>
<thead>
<tr>
<th>Customer Unique Id</th>
<th>Product Unique Id</th>
<th>Reference to a Customer</th>
<th>Order Unique Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer ID</td>
<td>Company Name</td>
<td>Sales Representative</td>
<td>Order ID</td>
</tr>
<tr>
<td></td>
<td>Contact Name</td>
<td>Address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Contact Title</td>
<td>City</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Address</td>
<td>Region</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Postal Code</td>
<td>Country</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Country</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

### Products Table

<table>
<thead>
<tr>
<th>Product ID</th>
<th>Product Name</th>
<th>Supplier</th>
<th>Category</th>
<th>Quantity Per Unit</th>
<th>Unit Price</th>
<th>Units in Stock</th>
<th>Units On Order</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Chai</td>
<td>Tokyo Traders</td>
<td>Beverages</td>
<td>10 boxes x 20 bags</td>
<td>$10.00</td>
<td>30</td>
<td>20</td>
<td>Liquid</td>
</tr>
<tr>
<td>2</td>
<td>Chong</td>
<td>Grandpa's Market</td>
<td>Condiments</td>
<td>12 - 550 ml bottles</td>
<td>$15.00</td>
<td>17</td>
<td>4</td>
<td>Sauce</td>
</tr>
<tr>
<td>3</td>
<td>Aniseed Syrup</td>
<td>Grandma's Market</td>
<td>Exotic Liquids</td>
<td>24 - 12 oz bottles</td>
<td>$10.00</td>
<td>10</td>
<td>12</td>
<td>Syrup</td>
</tr>
</tbody>
</table>

---

### Orders Table

<table>
<thead>
<tr>
<th>Order ID</th>
<th>Customer</th>
<th>Employee</th>
<th>Order Date</th>
<th>Required Date</th>
<th>Shipped Date</th>
<th>Ship Via</th>
<th>Freight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1025</td>
<td>Buchanan, Steven</td>
<td>Milton</td>
<td>08-Aug-94</td>
<td>01-Sep-94</td>
<td>16-Aug-94</td>
<td>Federal Shipping</td>
<td>$32.00</td>
</tr>
<tr>
<td>1026</td>
<td>Longman, Janet</td>
<td>Jennifer</td>
<td>09-Aug-94</td>
<td>15-Sep-94</td>
<td>20-Sep-94</td>
<td>FedEx Express</td>
<td>$31.00</td>
</tr>
</tbody>
</table>

---

*Unique order number.*
Relational Database Management Systems (DBMS)

- Allows the creation of relational databases
- Supports specialized languages for easy retrieval of data from a set of inter-related tables
- Supports easy construction of a Graphical User Interface on top of the database
- Allows very large table sizes
- Provides security, fault tolerance, multi-user support, etc.
Every statement yields a table of values as output
  – Sometimes there’s only one row in the table!

select columns and/or expressions
from tables
where conditions on the rows
group by group rows together
order by order the rows
Display an Entire Table

SELECT * 
FROM Employees;
Choose Columns

- Choosing a subset of columns is sometimes called a "project" operation

- Display first and last name of all employees

  SELECT FirstName, LastName  
  FROM Employees;

- Display company name and contact name for all customers
Choose Rows

- **Find US Employees**
  
  ```sql
  SELECT FirstName, LastName
  FROM Employees
  WHERE Country = "USA";
  ```

- **Find employees hired after Jan. 1, 1993**
• **Find total inventory value of each product**
  
  ```sql
  SELECT ProductName,
         UnitPrice*UnitsInStock AS TotalValue
  FROM Products;
  ```

• **Nice names for output columns**
  – Name following computed column (e.g., TotalValue) will be used to name output column

• **Find total price for each line item in “Order Details” table**
Sorting

• **Can sort output by contents of a column**
  – sort in ascending or descending order
  – sort by more than one column (second one breaks ties)

• **Sort products by total inventory value**
  
  ```sql
  SELECT ProductName,
          UnitPrice*UnitsInStock AS TotalValue
  FROM Products
  ORDER BY TotalValue DESC;
  ```

• **What are our 10 most expensive products?**
Aggregates

- **Can make calculations on entire columns**
  - sum, avg, max, min, count

- **What is the total value of a given customer order**
  
  ```sql
  SELECT OrderID, Sum([UnitPrice]*[Quantity]*(1-[Discount]))
  AS Subtotal
  FROM [Order Details]
  WHERE OrderID=11001;
  ```
  - returns a table with just one row!

- **What is average unit price of our products?**
Grouping and Aggregates

• Each different value for the GROUP BY fields defines a new group
  – One row of output is produced for each group
  – Several rows of input table may belong to same group. They are aggregated using aggregation operator.

• Compute total value of all orders

```sql
SELECT OrderID,
       Sum([UnitPrice]*[Quantity]*(1-[Discount]))
       AS Subtotal
FROM [Order Details]
GROUP BY OrderID;
```

• Create a table that shows how many line items are in each order
Joins

• Combine rows from one table with rows from another

• Usually join on some common column
  – Don't combine rows unless their value in the common column is the same
  – WHERE clause says the common column must be same in each table

• Produce a list of all products and their categories
  SELECT Products.ProductName,
          Categories.CategoryName
  FROM Categories, Products
  WHERE
       Categories.CategoryID = Products.CategoryID;
More Join examples

• Produce a list of all products and their suppliers

• Produce a list of all suppliers for Tofu
SQL Summary

select columns and/or expressions
from tables
where conditions on the rows
group by group rows together
order by order the rows