Vocabulary:
- database
- Structured Query Language (SQL)
- Database Management System (DBMS)
- table
- field (column)
- record (row)
- key
  - primary key
  - foreign key
- relational database

Overall goal of database design:
Separate data in separate tables. Strive to not duplicate data anywhere.
Example: products a store carries and suppliers

Could design a single table:

```
PRODUCTS:
ProdID ProdName Descr SupplierName SupplierAddr SupplierPh
```

But consider: many products have the same supplier (Mattel for toys, eg)
What happens if contact leaves supplier? How many records do I have to update?

Better:

```
PRODUCTS:
ProdID ProdName Descr SupplierID

SUPPLIERS:
SupplierID SuppName SuppAddr SuppPh
```

How to go from "real-world" ideas to database models?

1. What is objective?
   Design db to track key business elements of Toys Unlimited.
2. List the objects (themes of data to be kept track)
   Employees, Customers, Invoices, Products
3. List the facts about the objects (data related to object).
   (a) Employees:
       employee id, name, SSN, DoB, gender, hire date
   (b) Customers:
       company name, address, city, state, zip, title
   (c) Products:
product name, description, cost, markup
(d) Invoices:
  date, salesperson, customer, quantity, shipping charge
4. Turn the objects and facts into tables and columns.
(a) Tables: Employees, Customers, Products, Invoices
(b) Columns: emp_id, name, etc. (See below.) There are more fields we could have, but we keep it limited here.
(c) Naming conventions:
   i. Table names: usually capitalized first letters. Usually plural.
   ii. No spaces, no special characters.
   iii. Columns which are keys should usually have 'id' in them somewhere. Ex: EmployeeID

Toys Unlimited

Employees:

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>SSN</th>
<th>DoB</th>
<th>Gender</th>
<th>HireDate</th>
</tr>
</thead>
</table>

Customers:

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
</tr>
</thead>
</table>

Products:

<table>
<thead>
<tr>
<th>ProdName</th>
<th>Desc</th>
<th>Cost</th>
<th>Markup</th>
</tr>
</thead>
</table>

Invoices:

<table>
<thead>
<tr>
<th>Date</th>
<th>Salesperson</th>
<th>Customer</th>
<th>Products</th>
</tr>
</thead>
</table>
5. Determine the relationship among objects.
   (a) One-to-one relationships: For any given row in Table A, there is only one row in Table B.
       For any given row in Table B, there is only one row in Table A.
       i. There are no one-to-one relationships in the Toys database.
       ii. Example could be: employee data and private employee data. General information, such as employee name, address, and start date, is kept in one table. To ensure privacy, personal information, such as salary, is kept in another table. For each record in the general employee table, there should be one and only one corresponding record in the private employee data table. The reverse should also be true.

   (b) One-to-many relationships: For any given row in Table A, there are many rows in Table B.
       For any given row in Table B, there is only one row in Table A.
       i. The relationship between customers and invoices is one-to-many. One invoice is related to one customer, but a customer can have many invoices.

   (c) Many-to-many relationships: For any given row in Table A, there are many rows in Table B.
       For any given row in Table B, there are many rows in Table A.
       i. There is a many-to-many relationship between the product table and the invoice table. A product can be associated with many different invoices and an invoice can contain many different products.
6. Determine the key columns. Add key columns where appropriate.
   (a) A key can be an account number, social security number, part number, license number, or
      any other numeric value or combination of characters that are unique.
   (b) Keys must be unique. Do not use any column where possibility of duplicates exist. DBMS
      may enforce this for us at times.
   (c) Often use sequential numerical data. 1, 2, 3, etc.

7. Determine the linking columns.
   (a) If you have been careful about designating key columns, you also have determined the
      linking columns.
   (b) Links provide a way to tie information (rows) in one table to another table. If a table has a
      key column, that column can generally serve as the link.
   (c) Tables are linked together through their key columns.
   (d) Not all tables must be linked to every other table. Example: it makes no sense to link
      employees to products.

<table>
<thead>
<tr>
<th>Employees</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmpID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customers</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Products</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProdID</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Invoices</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvoiceID</td>
</tr>
</tbody>
</table>
8. Determine the constraints.
   (a) What limitations do we place on data going into a table?
   (b) Can help prevent holes in our data.
   (c) Example: shouldn't be able to create a record in the Invoices table (place an order) for a product which doesn't have a record in the Products table.
   (d) Different SW tools handle constraints differently.
9. Evaluate the design model. Tricky until you're an expert.
10. Implement the database using DBMS of your choice.