THE RELATIONAL MODEL
**HOW TO BUILD A DB APPLICATION**

- Pick an application
- Figure out what to model (**ER model**)  
  - Output: **ER diagram**

- Transform the ER diagram to a **relational schema**

- Refine the relational schema (**normalization**)  
- Now ready to implement the schema and load the data!
**ER Model vs Relational Model**

- **ER model**
  - many concepts: entities, relations, attributes, etc
  - well-suited for capturing the app requirements
  - **not** well-suited for computer implementation

- **Relational model**
  - has just a single concept: relation
  - world is represented with a collection of tables
  - well-suited for efficient manipulations on computers
RELATIONAL MODEL: BASICS
The data is stored in **tables (relations)**

### PRODUCT

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>price</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPad</td>
<td>tablet</td>
<td>$399.00</td>
<td>Apple</td>
</tr>
<tr>
<td>Surface</td>
<td>tablet</td>
<td>$299.00</td>
<td>Microsoft</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
DOMAINS

• Each attribute has an atomic type called domain
• A domain specifies the set of values allowed
• Examples:
  – integer
  – string
  – real

PRODUCT(name: string,
category: string,
price: real,
manufacturer: string)
The schema of a relation:

- relation name + attribute names
- **Product**(name, price, category, manufacturer)
- In practice we add the domain for each attribute

The schema of a database

- a collection of relation schemas
The instance of a relation:
   – a set of tuples or records

The instance of a database
   – a collection of relation instances
### Example

**PRODUCT** (name: *string*,
category: *string*,
price: *real*,
manufacturer: *string*)

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>price</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPad</td>
<td>tablet</td>
<td>$399.00</td>
<td>Apple</td>
</tr>
<tr>
<td>Surface</td>
<td>tablet</td>
<td>$299.00</td>
<td>Microsoft</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Schema vs Instance

• Analogy with programming languages:
  – schema = type
  – instance = value

• Important distinction:
  – schema = stable over long periods of time
  – instance = changes constantly, as data is inserted/updated/deleted
ER to Relational Model
TRANSLATION

• Basic cases:
  – entity set $E$ --> relation with attributes of $E$
  – relationship $R$ --> relation with attributes being keys of related entity sets + attributes of $R$

• Special cases:
  – combining two relations
  – weak entity sets
  – is-a relationships
RUNNING EXAMPLE

Product ➔ makes ➔ Company

Person ➔ buys ➔ Product

Person ➔ employs ➔ Company

<table>
<thead>
<tr>
<th>Attribute</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>category</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>price</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stockprice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**ENTITY SET TO RELATION**

The entity set `Person` is represented by the diagram with attributes `name`, `address`, and `ssn`.

The SQL CREATE TABLE statement for the `Person` table is:

```sql
CREATE TABLE Person (ssn CHAR(11) PRIMARY KEY, name CHAR(40), address CHAR(50))
```
**ENTITY SET TO RELATION**

```
CREATE TABLE Product 
  (name CHAR(40),
   category CHAR(20),
   price REAL,
   PRIMARY KEY(name, category))
```
Product\((name, category, price)\)  
Person\((ssn, name, address)\)  

Buys\((prodname, prodcategory, ssn, date)\)
No need for a Makes relation; instead modify Product:

**Product**\( (\text{name, category, price, company\_name}) \)

**Company**\( (\text{name, stockprice}) \)
What is wrong here?
CREATE TABLE Buys
    (prodname CHAR(40),
     prodcategory CHAR(20),
     ssn CHAR(11),
     date DATE,
     PRIMARY KEY(prodname, prodcategory, ssn)
    FOREIGN KEY (ssn)
        REFERENCES Person,
    FOREIGN KEY (prodname, prodcategory)
        REFERENCES Product(name, category))
Weak Entity Sets

**Team** *(number, affiliated-university, sport)*

- **Affiliation** does not need a separate relation!
- Attribute ‘**name**’ needed as part of the key
**Weak Entity Sets**

- The relation for a weak entity set must include:
  - attributes for its complete key (including those in other entity sets)
  - its own, non-key attributes

- A supporting (double-diamond) relationship is redundant and produces no relation
SUBCLASSES: OPTION 1

- **Product** (name, category, price)
- **SoftwareProduct** (name, category, price, version)
- **EducationalProduct** (name, category, price, age-group)
SUBCLASSES: OPTION 2

- **Product** (name, category, price)
- **SoftwareProduct** (name, version)
- **EducationalProduct** (name, age-group)
**SUBCLASSES: OPTION 3**

- **Product** (name, category, price, version, age-group)
- Use **NULL** to denote that the attribute makes no sense for a specific tuple
**SUBCLASSES Recap**

Three approaches:

1. create a relation for each class with all its attributes
2. create one relation for each subclass with only the key attribute(s) and attributes attached to it
3. create one relation; entities have null in attributes that do not belong to them
RECAP

Relational Model

- relation, attributes
- schema vs instance

ER to Relational

- entity set, relationship $\rightarrow$ relation
- primary keys, foreign keys
- special cases: weak entity sets, subclasses