Lecture 06: Constraints and E/R model
Constraints

- Constraint: a property that we want our data to satisfy
- Enforce by taking actions:
  - Forbid an update
  - Or perform compensating updates

- Two approaches:
  - Declarative constraints
  - Triggers
Constraints in SQL

- Keys, foreign keys
- Attribute-level constraints
- Tuple-level constraints
- Global constraints: assertions

The more complex the constraint, the harder it is to check and enforce
Keys

Product(name, category)

OR:

CREATE TABLE Product (  
name CHAR(30) PRIMARY KEY,  
category VARCHAR(20))
Keys with Multiple Attributes

CREATE TABLE Product (  
  name CHAR(30),  
  category VARCHAR(20),  
  price INT,  
  PRIMARY KEY (name, category))

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
<td>10</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td>20</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Photo</td>
<td>30</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
<td>40</td>
</tr>
</tbody>
</table>
Other Keys

CREATE TABLE Product (  
    productID CHAR(10),  
    name CHAR(30),  
    category VARCHAR(20),  
    price INT,  
    PRIMARY KEY (productID),  
    UNIQUE (name, category))

There is at most one PRIMARY KEY; there can be many UNIQUE
CREATE TABLE Purchase (  
  prodName CHAR(30)  
    REFERENCES Product(name),  
  date DATETIME)  

prodName is a **foreign key** to Product(name)  
name must be a **key** in Product
CREATE TABLE Purchase (  
  prodName CHAR(30),  
  category VARCHAR(20),  
  date DATETIME,  
  store VARCHAR(30),  
  FOREIGN KEY (prodName, category)  
  REFERENCES Product(name, category)  
)

Product(name, category, price)
Purchase(prodName, category, date)
What happens during updates?

- In Purchase: insert/update
- In Product: delete/update

<table>
<thead>
<tr>
<th>Product</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Category</td>
</tr>
<tr>
<td>Gizmo</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Purchase</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ProdName</td>
<td>Store</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
<tr>
<td>Camera</td>
<td>Ritz</td>
</tr>
<tr>
<td>Camera</td>
<td>Wiz</td>
</tr>
</tbody>
</table>
What happens during updates?

- SQL has three policies for maintaining referential integrity:
  - Reject violating modifications (default)
  - Cascade: after a delete/update do a delete/update
  - Set-null set foreign-key field to NULL

```sql
CREATE TABLE Purchase (
  prodName CHAR(30)
  REFERENCES Product(name)
  ON DELETE SET NULL
  ON UPDATE CASCADE)
```
Constraints on Attributes and Tuples

- **Constraints on attributes:**
  - NOT NULL -- obvious meaning...
  - CHECK condition -- any condition!

```sql
CREATE TABLE Purchase (... 
  store VARCHAR(30) NOT NULL, ...)

CREATE TABLE Product(... 
  price INT CHECK(price>0 and price <999))
```

- **Constraints on tuples**
  - CHECK condition

```sql
... CHECK(price*quantity < 1000)...
```
CREATE TABLE Purchase (prodName CHAR(30)
  CHECK (prodName IN SELECT Product.name FROM Product),
  date DATETIME NOT NULL)

How is this different from a foreign key constraint?
CREATE ASSERTION myAssert CHECK NOT EXISTS ( 
  SELECT Product.name 
  FROM Product, Purchase 
  WHERE Product.name = Purchase.prodName 
  GROUP BY Product.name 
  HAVING count(*) > 200 )
Semantic Optimization with Constraints

Product(name, price)
Purchase(buyer, seller, prodName, store)

\[
\begin{align*}
\text{SELECT} & \quad \text{Purchase.store} \\
\text{FROM} & \quad \text{Product, Purchase} \\
\text{WHERE} & \quad \text{Product.name} = \text{Purchase.prodName}
\end{align*}
\]

\[
\begin{align*}
\text{SELECT} & \quad \text{Purchase.store} \\
\text{FROM} & \quad \text{Purchase}
\end{align*}
\]
Semantic Optimization with Constraints

Product(name, price)
Purchase(buyer, seller, prodName, store)

```
SELECT Purchase.store
FROM Product, Purchase
WHERE Product.name = Purchase.prodName
```

SELECT Purchase.store
FROM Purchase

Yes, if Purchase.prodName is a foreign key, and not null
Database Design

- Why do we need it?
  - Need a way to model real world entities in terms of relations
  - Not easy to go from real-world entities to a database schema

- Consider issues such as:
  - What entities to model
  - How entities are related
  - What constraints exist in the domain
  - How to achieve good designs

- Several formalisms exist
  - We discuss E/R diagrams
Database Design Process

Today
Data Modeling

E/R diagrams

Next lecture
Refinement

Relations

SQL Tables

Conceptual Schema

Files

Physical Schema

Next lecture

Conceptual Schema
Conceptual Schema Design

Conceptual Model:

Relational Model: plus FDs
(FD = Functional Dependency)

Normalization: Eliminates anomalies
Entity / Relationship Diagrams

Product

address

entity set

All entities in the same entity set have the same attributes

bys

Association between 2 or more entities

A relationship may have attributes too!
Keys in E/R Diagrams

- Every entity set must have a key

![E/R Diagram with keys]

- name
- price
- category

Product
What is a Relationship?

- if A, B are sets, then a relationship R is a subset of $A \times B$
  - $A=\{1,2,3\}$, $B=\{a,b,c,d\}$,
  - $A \times B = \{(1,a),(1,b), \ldots, (3,d)\}$
  - $R = \{(1,a), (1,c), (3,b)\}$

- “makes” is a subset of Product $\times$ Company:
Multiplicity of E/R Relations

- **one-one:**

- **one-many:**

- **many-many:**
What does this mean?
Multi-way Relationships

How do we model a work relationship between employees, departments and locations?
Key Constraints in Multi-way Relationships

Q: What does the arrow mean?

A: A given employee works at a single department and location.
Participation Constraints

Q: Every employee works in at least one department.
Participation Constraints

Q: Every employee works in at least one department, and every department has at least one employee

Employees \rightarrow\left\downarrow{\text{WorksIn}}\rightarrow\left\downarrow{\text{Departments}}\right\downarrow
Participation Constraints

Q: What does this mean?

A: previous + every department has exactly one manager
Constraints in E/R Diagrams

- Finding constraints is part of the modeling process.

- Commonly used constraints:
  - **Keys:** social security number uniquely identifies a person.
  - **Single-value constraints:** a person can have only one father.
  - **Referential integrity constraints:** if you work for a company, it must exist in the database.
  - **Other constraints:** peoples’ ages are between 0 and 150.
Keys in E/R Diagrams

Product
- name
- category
- price

Person
- ssn
- name
- address

No formal way to specify multiple keys
Single Value Constraints

makes

vs

makes
Referential Integrity Constraints

Each product made by at most one company. Some products made by no company.

Each product made by \textit{exactly} one company.
Other Constraints

What does this mean?

Product makes <100 Company
Design Principles

What’s wrong?

Product — Purchase — Person

Country — President — Person

Be faithful to the specifications of the app!
Design Principles

What’s wrong?

Product - Purchase - Store

- personAddr
- personName
- date

Pick the right kind of entities!
What’s wrong?

Don’t complicate your design more than necessary!

Design Principles

Don’t complicate your design more than necessary!
From E/R Diagrams to Relational Schema

- Entity set $\rightarrow$ relation
- Relationship $\rightarrow$ relation
Entity Set to Relation

Product(name, category, price)

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadgets</td>
<td>$19.99</td>
</tr>
</tbody>
</table>
### Relationships to Relations

#### Makes

<table>
<thead>
<tr>
<th>ProductName</th>
<th>ProductCategory</th>
<th>CompanyName</th>
<th>startYear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadgets</td>
<td>GizmoWorks</td>
<td>1963</td>
</tr>
</tbody>
</table>

Watch out for attribute name conflicts.
Modeling Subclasses

- Some objects in a class may be special
- Define a new class
- Better: define a *subclass*

So --- we define subclasses in E/R
Subclasses

Product

- **price**
- **category**
- **name**

isa

Software Product

isa

Educational Product

platforms

ageGroup
Subclasses to Relations

<table>
<thead>
<tr>
<th>Product</th>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>99</td>
<td>gadget</td>
<td></td>
</tr>
<tr>
<td>Camera</td>
<td>49</td>
<td>photo</td>
<td></td>
</tr>
<tr>
<td>Toy</td>
<td>39</td>
<td>gadget</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Educational Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>Gizmo</td>
</tr>
<tr>
<td>Camera</td>
</tr>
</tbody>
</table>
E/R Inheritance

Entity sets overlap

No need for multiple inheritance
Say: each piece of furniture is owned either by a person, or by a company
Modeling Union Types with Subclasses

- Solution 1: acceptable, but imperfect (why?)

![Diagram showing relationships between Person, FurniturePiece, and Company with ownedByPerson and ownedByComp properties.]
Solution 2: better, more laborious

In fact there is no formal way to represent disjoint vs overlapping subclasses.
Weak Entity Sets

En2ty sets are weak when their key comes from classes to which they’re related
Handling Weak Entity Sets

No need to represent policy separately

Employee(ssn, name, dept)
Dependents(ssn, name, age)

We should have an arrow here, otherwise Dependents key would have NULL value!