Database Management Systems
CS 457/657

Lectures 8: ER Diagram, Constraints
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 exists
  – We discuss E/R diagrams
Database Design Process

• Conceptual Model:
  - Conceptual Schema
  - Physical Schema

• Relational Model:
  - Tables + constraints
  - And also functional dep.

• Normalization:
  - Eliminates anomalies

Normalization:
- Eliminates anomalies

Physical storage details:
- Physical Schema
Entity / Relationship Diagrams

• Entity set = a class
  – An entity = an object

• Attribute

• Relationship
Keys in E/R Diagrams

• Every entity set must have a key

- **Product**
  - **name**
  - **price**
What is a Relation?

- A mathematical definition:
  - if A, B are sets, then a relation 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** × **Company**:

![Diagram showing the relationship between Product, makes, and Company]
Multiplicity of E/R Relations

• one-one:
  – (name,id)

• many-one
  – (employee,company)

• many-many
  – (book,author)
What does this say?
Multi-way Relationships

• How do we model a purchase relationship between buyers, products and stores?

• Product

• Purchase

• Person

• Store

• Can still model as a mathematical set (Q. how ?)

• A. As a set of triples \( \subseteq \text{Person} \times \text{Product} \times \text{Store} \)
Q: What does the arrow mean?

A: A given person buys a given product from at most one store.

[Arrow pointing to E means that if we select one entity from each of the other entity sets in the relationship, those entities are related to at most one entity in E]
Q: What does the arrow mean?

A: A given person buys a given product from at most one store AND every store sells to every person at most one product.
Converting Multi-way Relationships to Binary

• Purchase
  • ProductOf
    • Product
  • StoreOf
    • Store
  • BuyerOf
    • Person

• date

Arrows go in which direction?
Converting Multi-way Relationships to Binary

- Purchase
- StoreOf
- BuyerOf
- ProductOf
- Product
- Store
- Person

- date

Make sure you understand why!
**Design Principles**

- **What’s wrong?**

- **Product**
  - Purchase
    - President
      - Country
      - Person
    - Person
Design Principles: What’s Wrong?

- Product
- Store
- Purchase
- date
- personName
- personAddr

pick the right kind of entities.
Design Principles: What’s Wrong?

- Product
- Purchase
- Dates
- Store
- Person
- Date

Don’t complicate life more than it already is.
From E/R Diagrams to Relational Schema

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

**Product** *(prod-ID, category, price)*

<table>
<thead>
<tr>
<th>prod-ID</th>
<th>category</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo55</td>
<td>Camera</td>
<td>99.99</td>
</tr>
<tr>
<td>Pokemn19</td>
<td>Toy</td>
<td>29.99</td>
</tr>
</tbody>
</table>
N-N Relationships to Relations

- Represent this in relations

Order relations:
- Orders
  - prod-ID
  - cust-ID
  - date

Shipment relations:
- Shipping-Co
  - date
  - name
  - address
Orders\textbf{(prod-ID, cust-ID, date)}

Shipment\textbf{(prod-ID, cust-ID, name, date)}

Shipping-Co\textbf{(name, address)}
Represent this in relations
N-1 Relationships to Relations

Orders(\text{prod-ID}, \text{cust-ID}, \text{date1}, \text{name}, \text{date2})

Shipping-Co(\text{name}, \text{address})

• Remember: no separate relations for many-one relationship
Multi-way Relationships to Relations

Purchase \((\text{prod-ID}, \text{ssn}, \text{name})\)
Modeling Subclasses

• Some objects in a class may be special
  • define a new class
  • better: define a subclass

• Products
  • Software products
  • Educational products

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

- Product
  - name
  - category
  - price
  - isa

  - Software Product
    - platforms
  - Educational Product
    - Age Group
Subclasses to Relations

- Product
  - name
  - category
  - price
- Software Product
  - name
  - category
  - price
  - isa
- Educational Product
  - name
  - category
  - price
  - isa

Other ways to convert are possible

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

- Sw.Product
  - Name
  - platforms
  - Gizmo
  - unix

- Ed.Product
  - Name
  - Age Group
  - Gizmo
  - toddler
  - Toy
  - retired
Modeling UnionTypes With Subclasses

• Say: each piece of furniture is owned either by a person or by a company
Modeling Union Types with Subclasses

Say: each piece of furniture is owned either by a person or by a company

Solution 1. Acceptable but imperfect
Modeling Union Types with Subclasses

Solution 2: better, more laborious
Weak Entity Sets

Entity sets are weak when their key comes from other classes to which they are related.

Team

- sport
- number

affiliation

University

- name

Team(sport, number, universityName)
University(name)
What makes good schemas?
Integrity Constraints
Motivation

• An integrity constraint is a condition specified on a database schema that restricts the data that can be stored in an instance of the database.

• ICs help prevent entry of incorrect information
• How? DBMS enforces integrity constraints
  – Allows only legal database instances (i.e., those that satisfy all constraints) to exist
  – Ensures that all necessary checks are always performed and avoids duplicating the verification logic in each application
• 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

• Underline:

No formal way to specify multiple keys in E/R diagrams
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 exactly one company.
Other Constraints

Q: What does this mean?
A: A Company entity cannot be connected by relationship to more than 99 Product entities
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 to enforce.
Key Constraints

• Product(name, category)

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

OR:

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

- Product(name, category, price)

```
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>
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

Referential integrity constraints

- May write just Product if name is PK
Foreign Key Constraints

• Example with multi-attribute primary key

CREATE TABLE Purchase (  
    prodName CHAR(30),  
    category VARCHAR(20),  
    date DATETIME,  
    FOREIGN KEY (prodName, category)  
    REFERENCES Product(name, category))

• (name, category) must be a KEY in Product
What happens when data changes?

Types of updates:
- In Purchase: insert/update
- In Product: delete/update

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<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>ProdName</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<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 when data changes?

- SQL has three policies for maintaining referential integrity:
  - **NO ACTION** reject violating modifications (default)
  - **CASCADE** after delete/update do delete/update
  - **SET NULL** set foreign-key field to NULL
  - **SET DEFAULT** set foreign-key field to default value

  – need to be declared with column, e.g.,
  CREATE TABLE Product (pid INT DEFAULT 42)
Maintaining Referential Integrity

CREATE TABLE Purchase (  
    prodName CHAR(30),
    category VARCHAR(20),
    date DATETIME,
    FOREIGN KEY (prodName, category)
    REFERENCES Product(name, category)
    ON UPDATE CASCADE
    ON DELETE SET NULL
)
Constraints on Attributes and Tuples

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

• Constraints on tuples  
  CHECK condition
Constraints on Attributes and Tuples

```
CREATE TABLE R (  
    A int NOT NULL,  
    B int CHECK (B > 50 and B < 100),  
    C varchar(20),  
    D int,  
    CHECK (C >= 'd' or D > 0))
```
Constraints on Attributes and Tuples

CREATE TABLE Product (
  productID CHAR(10),
  name CHAR(30),
  category VARCHAR(20),
  price INT CHECK (price > 0),
  PRIMARY KEY (productID),
  UNIQUE (name, category))
Constraints on Attributes and Tuples

What does this constraint do?

CREATE TABLE Purchase (  
prodName CHAR(30)  
CHECK (prodName IN  
(SELECT Product.name  
FROM Product)),  
date DATETIME NOT NULL)
CREATE ASSERTION myAssert CHECK
(NOT EXISTS(
    SELECT Product.name
    FROM Product, Purchase
    WHERE Product.name = Purchase.prodName
    GROUP BY Product.name
    HAVING count(*) > 200)
)

• But most DBMSs do not implement assertions
• Because it is hard to support them efficiently
• Instead, they provide triggers
Database Triggers

- **Event-Condition-Action** rules
- **Event**
  - Can be insertion, update, or deletion to a relation
- **Condition**
  - Can be expressed on DB state before or after event
- **Action**
  - Perform additional DB modifications
More About Triggers

• Row-level trigger
  – Executes once for each modified tuple

• Statement-level trigger
  – Executes once for all tuples that are modified in a SQL statement
Database Triggers Example

• When Product.price is updated, if it is decreased then set Product.category = ‘On sale’

CREATE TRIGGER ProductCategories
AFTER UPDATE OF price ON Product
REFERENCING
  OLD ROW AS OldTuple
  NEW ROW AS NewTuple
FOR EACH ROW
WHEN (OldTuple.price > NewTuple.price)
  UPDATE Product
  SET category = ‘On sale’
WHERE productID = OldTuple.productID
CREATE TRIGGER ProductCategory
ON Product
AFTER UPDATE
AS
BEGIN
    UPDATE Product
    SET category='sale' WHERE productID IN
    (SELECT i.productID from inserted i, deleted d
     WHERE i.productID = d.productID
     AND i.price < d.price)
END
What’s SQL code look like in the real world?

• A lot of Stored Procedures
  – Can be executed by the trigger
  – More like a function or method in imperative languages (C++, Java, etc.)
  – There are fine-grained controls like
    • If-else check
    • While loop