**Introduction to Data Management**

CSE 414

Unit 6: Conceptual Design

E/R Diagrams

Integrity Constraints

BCNF

(3 lectures)

---

**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

**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**

Underline:

- name
- category
- price
- product

No formal way to specify multiple keys in E/R diagrams

- Person
  - address
  - name
  - ssn

---

**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.

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

Constraints in SQL

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

```
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

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

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, category),
    date DATETIME
)

prodName is a foreign key to Product(name),
name must be a key in Product

What happens when data changes?

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

Product
<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>

Purchase
<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>

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
  - CHECK condition
- Constraints on tuples
  - CHECK condition

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)
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) 
)

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

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) )

Introduction to Data Management
CSE 414
Design Theory and BCNF

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

Relational Schema Design

One person may have multiple phones, but lives in only one city.

Primary key is thus (SSN, PhoneNumber)

What is the problem with this schema?
Relational Schema Design

Anomalies:
- Redundancy = repeat data
- Update anomalies = what if Fred moves to "Bellevue"?
- Deletion anomalies = what if Joe deletes his phone number?

Relation Decomposition

Functional Dependencies (FDs)

Example

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>Phone</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0045</td>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>E3542</td>
<td>Mike</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E1111</td>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E9999</td>
<td>Mary</td>
<td>1234</td>
<td>Lawyer</td>
</tr>
</tbody>
</table>

EmpID → Name, Phone, Position
Position → Phone
but not Phone → Position
Example

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>Phone</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0045</td>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>E3542</td>
<td>Mike</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E1111</td>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E9999</td>
<td>Mary</td>
<td>1234</td>
<td>Lawyer</td>
</tr>
</tbody>
</table>

Position $\rightarrow$ Phone

Example

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>Phone</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0045</td>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>E3542</td>
<td>Mike</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E1111</td>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E9999</td>
<td>Mary</td>
<td>1234</td>
<td>Lawyer</td>
</tr>
</tbody>
</table>

But not Phone $\rightarrow$ Position

Example

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>color</th>
<th>department</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
<td>Green</td>
<td>Toys</td>
<td>49</td>
</tr>
<tr>
<td>Tweaker</td>
<td>Gadget</td>
<td>Green</td>
<td>Toys</td>
<td>49</td>
</tr>
</tbody>
</table>

Do all the FDs hold on this instance?

Example

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>color</th>
<th>department</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
<td>Green</td>
<td>Toys</td>
<td>49</td>
</tr>
<tr>
<td>Tweaker</td>
<td>Gadget</td>
<td>Green</td>
<td>Toys</td>
<td>49</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Stationary</td>
<td>Green</td>
<td>Office-supp.</td>
<td>59</td>
</tr>
</tbody>
</table>

What about this one?

Buzzwords

- FD holds or does not hold on an instance
- If we can be sure that every instance of $R$ will be one in which a given FD is true, then we say that $R$ satisfies the FD
- If we say that $R$ satisfies an FD, we are stating a constraint on $R$

An Interesting Observation

If all these FDs are true:

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>color, category</th>
<th>department</th>
<th>price</th>
</tr>
</thead>
</table>

Then this FD also holds:

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>price</th>
</tr>
</thead>
</table>
An Interesting Observation

If all these FDs are true:
- name → color
- category → department
- color, category → price

Then this FD also holds: name, category → price

Closure of a set of Attributes

Given a set of attributes A₁, ..., Aₙ
The closure is the set of attributes B, notated A₁, ..., Aₙ*.

Example:
1. name → color
2. category → department
3. color, category → price

Closures:
- name* = {name, color}
- color* = {color}

Closure Algorithm

X={A₁, ..., An}.
Repeat until X doesn’t change do:
if B₁, ..., Bₙ → C is a FD and B₁, ..., Bₙ are all in X
then add C to X.

Example:
1. name → color
2. category → department
3. color, category → price

{name, category}* = {name, category, color, department}
Closure Algorithm

\[ X = \{A_1, \ldots, A_n\} \]
Repeat until \( X \) doesn't change do:
if \( B_1, \ldots, B_n \rightarrow C \) is a FD and \( B_1, \ldots, B_n \) are all in \( X \) then add \( C \) to \( X \).

\( \{ \text{name, category} \} \) + = \{ \text{name, category, color, department, price} \}

Example:
\begin{align*}
\{ \text{name, category} \}^* &= \\
&= \{ \text{name, category, price} \}
\end{align*}

Hence:
\begin{align*}
\text{name, category} &\rightarrow \text{color, department, price} \\
\text{name} &\rightarrow \text{color} \\
\text{category} &\rightarrow \text{department} \\
\text{color, category} &\rightarrow \text{price}
\end{align*}

Example

In class:
\( R(A,B,C,D,E,F) \)
\begin{align*}
A, B &\rightarrow C \\
A, D &\rightarrow E \\
B &\rightarrow D \\
A, F &\rightarrow B
\end{align*}
Compute \( (A,B)^* \) \( X = \{A, B, \ldots\} \)
Compute \( (A, F)^* \) \( X = \{A, F, \ldots\} \)

Example

In class:
\( R(A,B,C,D,E,F) \)
\begin{align*}
A, B &\rightarrow C \\
A, D &\rightarrow E \\
B &\rightarrow D \\
A, F &\rightarrow B
\end{align*}
Compute \( (A,B)^* \) \( X = \{A, B, C, D, E\} \)
Compute \( (A, F)^* \) \( X = \{A, F, \ldots\} \)

What is the key of \( R \)?
Find all FD's implied by:

A, B → C
A, D → B
B → D

Step 1: Compute $X^+$, for every $X$:

$A^+ = A$, $B^+ = BD$, $C^+ = C$, $D^+ = D$
$AB^+ = ABCD$, $AC^+ = AC$, $AD^+ = ABCD$
$BC^+ = BCD$, $BD^+ = BD$, $CD^+ = CD$
$ABC^+ = ABD^+ = ACD^+ = ABCD$ (no need to compute—why?)
$BCD^+ = BCD$, $ABCD^+ = ABCD$

Step 2: Enumerate all FD's $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

$AB \rightarrow CD$, $AD \rightarrow BC$, $ABC \rightarrow D$, $ABD \rightarrow C$, $ACD \rightarrow B$