CSE 544
Constraints

Lecture #3
Friday, January 13, 2011
Announcements

• **Tuesday 1/18:**
  – Guest Lecturer: *Bill Howe*

• **Wednesday 1/19:** 9am-11:30am, *Data Models*
  – Room: CSE 403,
  – Two papers to reviews: *What goes around comes around* and *Query answering using views* (Sec. 1-3)

• **Thursday 1/19: Transactions**
  – One paper to review (Sec. 1, 2.1, 2.2, 3.1, and 3.2)

• **Friday 1/21:** 11:30-1pm
  – Sign up to meet with me to discuss your project
Outline and Reading Material

• **Constraints:** Book 3.2, 3.3, 5.8
Constraints

• A constraint = a property that we’d like our database to hold
• Enforce it by taking some actions:
  – Forbid an update
  – Or perform compensating updates
• Two approaches:
  – Declarative integrity constraints
  – Triggers
Integrity 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
Keys

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

OR:  

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

Product(name, price)
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>

Product(name, category, price)
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 (  
    buyer CHAR(30),  
seller CHAR(30),  
product CHAR(30) REFERENCES Product(name),  
store VARCHAR(30))
<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>
Foreign Key Constraints

```sql
CREATE TABLE Purchase(
    buyer VARCHAR(50),
    seller VARCHAR(50),
    product CHAR(20),
    category VARCHAR(20),
    store VARCHAR(30),
    FOREIGN KEY (product, category)
        REFERENCES Product(name, category)
);
```

Purchase(buyer, seller, product, category, store)
Product(name, category, price)
What happens during updates?

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

<table>
<thead>
<tr>
<th>Product</th>
<th>Purchase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ProdName</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Gizmo</td>
</tr>
<tr>
<td>Camera</td>
<td>Camera</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
</tbody>
</table>

Name | Category
--- | ---
Gizmo | gadget
Camera | Photo
OneClick | Photo
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
Constraints on Attributes and Tuples

Attribute level constraints:

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

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

Tuple level constraints:

```
. . . CHECK (price * quantity < 10000) . . .
```
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)
Comments on Constraints

• Can give them names, and alter later

• We need to understand exactly when they are checked

• We need to understand exactly what actions are taken if they fail
Semantic Optimization using Constraints

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

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

When can we rewrite the query?

```
SELECT Purchase.store
FROM   Purchase
```
Semantic Optimization using Constraints

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

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

When can we rewrite the query?

```
SELECT Purchase.store
FROM   Purchase
```

Purchase.product is foreign key AND not null