Lecture 05
Views, Constraints
Friday, October 6, 2006
Outline

• Data Definition Language (6.6)

• Views (6.7)

• Constraints (Chapter 7)
Defining Views

Views are relations, except that they are not physically stored.

For presenting different information to different users

Employee(ssn, name, department, project, salary)

```
CREATE VIEW Developers AS
    SELECT name, project
    FROM Employee
    WHERE department = "Development"
```

Payroll has access to Employee, others only to Developers
CREATE VIEW CustomerPrice AS
SELECT x.customer, y.price
FROM Purchase x, Product y
WHERE x.product = y.pname

Example
Purchase(customer, product, store)
Product(pname, price)

CustomerPrice(customer, price)  “virtual table”
Purchase(customer, product, store)  
Product(pname, price)  
CustomerPrice(customer, price)

We can later use the view:

```sql
SELECT  u.customer, v.store
FROM      CustomerPrice u, Purchase v
WHERE     u.customer = v.customer  AND  
          u.price > 100
```
What Happens When We Query a View?

SELECT u.customer, v.store
FROM CustomerPrice u, Purchase v
WHERE u.customer = v.customer AND u.price > 100

SELECT x.customer, v.store
FROM Purchase x, Product y, Purchase v,
WHERE x.customer = v.customer AND y.price > 100 AND x.product = y.pname
Types of Views

• Virtual views:
  – Used in databases
  – Computed only on-demand – slow at runtime
  – Always up to date

• Materialized views
  – Used in data warehouses
  – Pre-computed offline – fast at runtime
  – May have stale data
Updying Views: Part 1

Purchase(customer, product, store)
Product(pname, price)

CREATE VIEW Expensive-Product AS
  SELECT pname
  FROM Product
  WHERE price > 100

INSERT INTO Expensive-Product VALUES('Gizmo')
INSERT INTO Product VALUES('Gizmo', NULL)
Updating Views: Part 2

Purchase(customer, product, store)
Product(pname, price)

CREATE VIEW AcmePurchase AS
    SELECT customer, product
    FROM Purchase
    WHERE store = 'AcmeStore'

INSERT INTO Toy-Product
VALUES('Joe', 'Gizmo')

INSERT INTO Product
VALUES('Joe', 'Gizmo', NULL)

Note this
Updateable view
Updating Views: Part 3

Purchase(customer, product, store)
Product(pname, price)

CREATE VIEW CustomerPrice AS
SELECT x.customer, y.price
FROM Purchase x, Product y
WHERE x.product = y.pname

INSERT INTO CustomerPrice
VALUES(‘Joe’, 200)

Non-updateable view

Most views are non-updateable
Constraints in SQL

• A constraint = a property that we’d like our database to hold

• The system will enforce the constraint by taking some actions:
  – forbid an update
  – or perform compensating updates
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
Keys

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

OR:

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

Product(name, category)
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
Foreign Key Constraints

CREATE TABLE Purchase (prodName CHAR(30)
REFERENCE Product(name),
date DATETIME)

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

<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

• OR

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

• (name, category) must be a PRIMARY KEY
What happens during updates?

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

READING ASSIGNMENT: 7.1.5, 7.1.6
Constraints on Attributes and Tuples

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

• Constraints on tuples
  CHECK condition
CREATE TABLE Purchase (  
  prodName CHAR(30)  
  CHECK (prodName IN SELECT Product.name FROM Product),  
  date DATETIME NOT NULL)
General Assertions

CREATE ASSERTION myAssert CHECK NOT EXISTS(
  SELECT Product.name 
  FROM Product, Purchase 
  WHERE Product.name = Purchase.prodName 
  GROUP BY Product.name 
  HAVING count(*) > 200)
Final Comments on Constraints

• Can give them names, and alter later
  – Read in the book !!!

• We need to understand exactly when they are checked

• We need to understand exactly what actions are taken if they fail