Lecture 04: SQL

Monday, October 7, 2002

Outline

- Getting around INTERSECT and EXCEPT
- Nulls (6.1.6)
- Outer joins (6.3.8)
- Database Modifications (6.5)
- Defining Relation Schema in SQL (6.6)
- Defining Views (6.7)

INTERSECT and EXCEPT: Not in SQL Server

Null Values and Outerjoins

Null Values and Outerjoins

Null Values and Outerjoins

Null Values and Outerjoins

Unexpected behavior:

SELECT *
FROM Person
WHERE age < 25 OR age >= 25

Some Persons are not included!
Null Values and Outerjoins

Can test for NULL explicitly:
  - `x IS NULL`
  - `x IS NOT NULL`

```sql
SELECT *
FROM Person
WHERE age < 25 OR age >= 25 OR age IS NULL
```

Now it includes all Persons

Null Values and Outerjoins

Left outer joins in SQL:

```sql
Product(name, category)
Purchase(prodName, store)
```

```sql
SELECT Product.name, Purchase.store
FROM Product LEFT OUTER JOIN Purchase ON
Product.name = Purchase.prodName
```

Null Values and Outerjoins

Explicit joins in SQL:

```sql
Product(name, category)
Purchase(prodName, store)
```

```sql
SELECT Product.name, Purchase.store
FROM Product JOIN Purchase ON
Product.name = Purchase.prodName
```

Same as:

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

But products that never sold will be lost!

Outer Joins

- Left outer join:
  - Include the left tuple even if there’s no match
- Right outer join:
  - Include the right tuple even if there’s no match
- Full outer join:
  - Include the both left and right tuples even if there’s no match

Modifying the Database

Three kinds of modifications

- Insertions
- Deletions
- Updates

Sometimes they are all called “updates”
## Insertions

**General form:**

\[
\text{INSERT INTO R(A1, \ldots, An) VALUES (v1, \ldots, vn)}
\]

Example: Insert a new purchase to the database:

\[
\text{INSERT INTO Purchase(buyer, seller, product, store) VALUES ('Joe', 'Fred', 'wakeup-clock-espresso-machine', 'The Sharper Image')}
\]

Missing attribute \(\rightarrow\) NULL
May drop attribute names if give them in order.

### Insertion: an Example

Product(name, listPrice, category)

Purchase(prodName, buyerName, price)

prodName is foreign key in Product.name

Suppose database got corrupted and we need to fix it:

<table>
<thead>
<tr>
<th>Product</th>
<th>Purchase</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>prodName</td>
</tr>
<tr>
<td>listPrice</td>
<td>buyerName</td>
</tr>
<tr>
<td>category</td>
<td>price</td>
</tr>
<tr>
<td>games</td>
<td>games</td>
</tr>
<tr>
<td>100</td>
<td>Smith</td>
</tr>
<tr>
<td>gadgets</td>
<td>cameras</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Task: insert in Product all prodNames from Purchase

### Insertion: an Example

\[
\text{INSERT INTO Product(name)}
\]

\[
\text{SELECT DISTINCT prodName}
\]

\[
\text{FROM Purchase}
\]

\[
\text{WHERE prodName NOT IN (SELECT name FROM Product)}
\]

<table>
<thead>
<tr>
<th>name</th>
<th>listPrice</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>games</td>
<td>100</td>
<td>Gadgets</td>
</tr>
<tr>
<td>camera</td>
<td>225</td>
<td>-</td>
</tr>
</tbody>
</table>

\[\rightarrow\] Depends on the implementation

## Deletions

Example:

\[
\text{DELETE FROM PURCHASE}
\]

\[
\text{WHERE seller = 'Joe' AND product = 'Brooklyn Bridge'}
\]

Factoid about SQL: there is no way to delete only a single occurrence of a tuple that appears twice in a relation.
Updates

Example:

```sql
UPDATE PRODUCT
SET price = price/2
WHERE Product.name IN
(SELECT product
FROM Purchase
WHERE Date = 'Oct, 25, 1999');
```

Data Definition in SQL

So far we have seen the Data Manipulation Language, DML.

Next: Data Definition Language (DDL)

Data types:
- Defines the types.

Data definition: defining the schema.

- Create tables
- Delete tables
- Modify table schema

Indexes: to improve performance

Data Types in SQL

- Characters:
  - CHAR(20) -- fixed length
  - VARCHAR(40) -- variable length

- Numbers:
  - INT, REAL plus variations

- Times and dates:
  - DATE, DATETIME (SQL Server only)

- To reuse domains:
  CREATE DOMAIN address AS VARCHAR(55)

Creating Tables

Example:

```sql
CREATE TABLE Person(  
    name VARCHAR(30),  
    social-security-number INT,  
    age SHORTINT DEFAULT 100,  
    city VARCHAR(30) DEFAULT 'Seattle',  
    gender CHAR(1) DEFAULT 'f',  
    Birthdate DATE
);
```

Deleting or Modifying a Table

Deleting:

Example: DROP Person; Exercise with care !!

Altering: (adding or removing an attribute).

Example:

```sql
ALTER TABLE Person
ADD phone CHAR(16);

ALTER TABLE Person
DROP age;
```

What happens when you make changes to the schema?

Default Values

Specifying default values:

```sql
CREATE TABLE Person(  
    name VARCHAR(30),  
    social-security-number INT,  
    age SHORTINT DEFAULT 100,  
    city VARCHAR(30) DEFAULT 'Seattle',  
    gender CHAR(1) DEFAULT 'f',  
    Birthdate DATE
);
```

The default of defaults: NULL
Indexes

**REALLY** important to speed up query processing time.
Suppose we have a relation

Person (name, age, city)

```
SELECT *
FROM Person
WHERE name = "Smith"
```

Sequential scan of the file Person may take long

---

Creating Indexes

Syntax:

```
CREATE INDEX nameIndex ON Person(name)
```

B+ trees help in:

```
SELECT *
FROM Person
WHERE age > 25 AND age < 28
```

Why not create indexes on everything?

---

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
A Different View

We can later use the view:

```
SELECT name, store
FROM Seattle-view, Product
WHERE Seattle-view.product = Product.name AND Product.category = "shoes"
```

What Happens When We Query a View?

```
SELECT name, Seattle-view.store
FROM Seattle-view, Product
WHERE Seattle-view.product = Product.name AND Product.category = "shoes"
```

```
SELECT name, Purchase.store
FROM Person, Purchase, Product
WHERE Person.city = "Seattle" AND Person.name = Purchase.buyer AND Purchase.product = Product.name AND Product.category = "shoes"
```

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
  - Precomputed offline – fast at runtime
  - May have stale data

Updating Views

How can I insert a tuple into a table that doesn't exist?

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

If we make the following insertion:

```
INSERT INTO Developers
VALUES('Joe', 'Optimizer')
```

It becomes:

```
INSERT INTO Employee
VALUES(NULL, 'Joe', NULL, 'Optimizer', NULL)
```

Non-Updatable Views

```
CREATE VIEW Seattle-view AS
SELECT seller, product, store
FROM Person, Purchase
WHERE Person.city = "Seattle" AND Person.name = Purchase.buyer
```

How can we add the following tuple to the view?

("Joe", "Shoe Model 12345", "Nine West")

We need to add "Joe" to Person first, but we don’t have all its attributes