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

```
(SELECT R.A, R.B
FROM   R)
   INTERSECT
(SELECT S.A, S.B
FROM   S)
```

⇒

```
SELECT R.A, R.B
FROM   R
WHERE
   EXISTS(SELECT *
          FROM S
          WHERE R.A=S.A and R.B=S.B)
```

```
(SELECT R.A, R.B
FROM   R)
   EXCEPT
(SELECT S.A, S.B
FROM   S)
```

⇒

```
SELECT R.A, R.B
FROM   R
WHERE
  NOT EXISTS(SELECT *
             FROM S
             WHERE R.A=S.A and R.B=S.B)
```


Null Values and Outerjoins

- If x=Null then 4*(3-x)/7 is still NULL
- If x=Null then x="Joe" is UNKNOWN
- In SQL there are three boolean values:

FALSE	=	0
UNKNOWN	=	0.5
TRUE	=	1

Null Values and Outerjoins

- C1 AND C2 = min(C1, C2)
- C1 OR C2 = max(C1, C2)
- NOT C1 = 1 – C1

```
SELECT *
FROM Person
WHERE  (age < 25) AND
       (height > 6 OR weight > 190)
```

E.g.
age=20
heigth=NULL
weight=200

Rule in SQL: include only tuples that yield TRUE

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

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

Now it includes all Persons

7

Null Values and Outerjoins

Explicit joins in SQL:
Product(name, category)
Purchase(prodName, store)

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

Same as:

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

But Products that never sold will be lost !

8

Null Values and Outerjoins

Left outer joins in SQL:
Product(name, category)
Purchase(prodName, store)

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

9

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

Name	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz
OneClick	NULL

10

Outer Joins

- Left outer join:
  - Incluce the left tuple even if there's no match
- Right outer join:
  - Incluce 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

11

Modifying the Database

Three kinds of modifications
- Insertions
- Deletions
- Updates

Sometimes they are all called "updates"

12

Insertions

General form:

```
INSERT INTO R(A1,…., An) VALUES (v1,…., vn)
```

Example: Insert a new purchase to the database:

```
INSERT INTO Purchase(buyer, seller, product, store)
       VALUES ('Joe', 'Fred', 'wakeup-clock-espresso-machine',
               'The Sharper Image')
```

Missing attribute → NULL.
May drop attribute names if give them in order.

13

Insertions

```
INSERT INTO PRODUCT(name)

   SELECT DISTINCT Purchase.product
   FROM   Purchase
   WHERE  Purchase.date > "10/26/01"
```

The query replaces the VALUES keyword.
Here we insert *many* tuples into PRODUCT

14

Insertion: an Example

Product(<u>name</u>, listPrice, category)
Purchase(prodName, buyerName, price)

prodName is foreign key in Product.name

Suppose database got corrupted and we need to fix it:

Product

name	listPrice	category
gizmo	100	gadgets

Purchase

prodName	buyerName	price
camera	John	200
gizmo	Smith	80
camera	Smith	225

Task: insert in Product all prodNames from Purchase

15

Insertion: an Example

```
INSERT INTO Product(name)

SELECT DISTINCT prodName
FROM   Purchase
WHERE  prodName NOT IN (SELECT name FROM Product)
```

name	listPrice	category
gizmo	100	Gadgets
camera	-	-

16

Insertion: an Example

```
INSERT INTO Product(name, listPrice)

SELECT DISTINCT prodName, price
FROM Purchase
WHERE prodName NOT IN (SELECT name FROM Product)
```

name	listPrice	category
gizmo	100	Gadgets
camera	200	-
camera ??	225 ??	-

← Depends on the implementation

17

Deletions

Example:

```
DELETE FROM PURCHASE

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.

18

Updates

Example:

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

19

Data Definition in SQL

So far we have see 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 peformance

20

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)

21

Creating Tables

Example:

```
CREATE     TABLE Person(

        name                    VARCHAR(30),
        social-security-number  INT,
        age                     SHORTINT,
        city                    VARCHAR(30),
        gender                  BIT(1),
        Birthdate               DATE

);
```

22

Deleting or Modifying a Table

Deleting:

Example:

```
DROP Person;
```

Exercise with care !!

Altering: (adding or removing an attribute).

Example:

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

ALTER  TABLE   Person
      DROP  age;
```

What happens when you make changes to the schema?

23

Default Values

Specifying default values:

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

The default of defaults: NULL

24

Indexes

REALLY important to speed up query processing time.

Suppose we have a relation

Person (name, age, city)

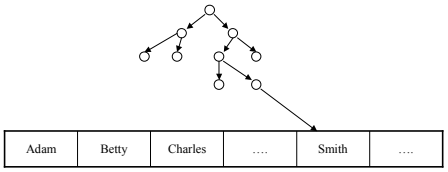
```
SELECT *
FROM    Person
WHERE   name = "Smith"
```

Sequential scan of the file Person may take long

25

Indexes

- Create an index on name:



- B+ trees have fan-out of 100s: max 4 levels !

26

Creating Indexes

Syntax:

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

27

Creating Indexes

Indexes can be created on more than one attribute:

Example:

```
CREATE INDEX doubleindex ON
                Person (age, city)
```

Helps in:

```
SELECT *
FROM    Person
WHERE age = 55 AND city = "Seattle"
```

But not in:

```
SELECT *
FROM    Person
WHERE city = "Seattle"
```

28

Creating Indexes

Indexes can be useful in range queries too:

```
CREATE INDEX ageIndex ON  Person (age)
```

B+ trees help in:

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

Why not create indexes on everything?

29

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

30

A Different View

Person(name, city)
Purchase(buyer, seller, product, store)
Product(name, maker, category)

```
CREATE VIEW  Seattle-view  AS

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

We have a new virtual table:
Seattle-view(buyer, seller, product, store)

31

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"
```

32

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.poduct = Product.name  AND
        Product.category = "shoes"
```

33

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

34

Updating Views

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

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

```
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)
```

35

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

36