Lecture 13: SQL

Monday, October 29, 2001

Outline

- Subqueries (5.3)
- Duplicates (5.4)
- Database Modifications (5.6)
- Defining Relation Schema in SQL (5.7)
- Defining Views (5.8)
- Indexes 5.7.7

Subqueries

A subquery producing a single tuple:

```
SELECT Purchase.product
FROM   Purchase
WHERE  buyer =
           (SELECT  name
            FROM    Person
            WHERE   ssn = "123456789");
```

In this case, the subquery returns one value.

If it returns more, it's a run-time error.

Can say the same thing without a subquery:

```
SELECT Purchase.product
FROM   Purchase, Person
WHERE  buyer = name AND ssn = "123456789"
```

This is equivalent to the previous one when the ssn is a key; otherwise they are different.

Subqueries Returning Relations

Find companies who manufacture products bought by Joe Blow.

```
SELECT  Company.name
FROM    Company, Product
WHERE   Company.name=Product.maker
        AND  Product.name  IN
             (SELECT Purchase.product
              FROM   Purchase
              WHERE Purchase .buyer = "Joe Blow");
```

Here the subquery returns a set of values

Subqueries Returning Relations

Equivalent to:

```
SELECT  Company.name
FROM    Company, Product, Purchase
WHERE   Company.name= Product.maker
        AND  Product.name  = Purchase.product
        AND  Purchase.buyer = "Joe Blow"
```

Is this query equivalent to the previous one ?

Beware of duplicates !

Removing Duplicates

```
SELECT  Company.name
FROM    Company, Product, Purchase
WHERE   Company.name= Product.maker
        AND  Product.name  = Purchase.product
        AND Purchase.buyer = "Joe Blow"
```

←Multiple copies

```
SELECT DISTINCT Company.name
FROM    Company, Product, Purchase
WHERE   Company.name= Product.maker
        AND  Product.name  = Purchase.product
        AND Purchase.buyer = "Joe Blow"
```

← Single copies

Removing Duplicates

```
SELECT DISTINCT Company.name
FROM    Company, Product
WHERE   Company.name= Product.maker
        AND  Product.name  IN
             (SELECT Purchase.product
              FROM   Purchase
              WHERE Purchase.buyer = "Joe Blow");
```

```
SELECT DISTINCT Company.name
FROM    Company, Product, Purchase
WHERE   Company.name= Product.maker
        AND  Product.name  = Purchase.product
        AND  Purchase.buyer = "Joe Blow"
```

Now they are equivalent

Subqueries Returning Relations

You can also use: s > ALL R
s > ANY R
EXISTS R

Product (pname, price, category, maker)
Find products that are more expensive than all those produced By "Gizmo-Works"

```
SELECT  name
FROM    Product
WHERE  price >  ALL (SELECT price
                     FROM    Purchase
                     WHERE  maker="Gizmo-Works")
```

Question for Database Fans and their Friends

- Can we express this query as a single SELECT-FROM-WHERE query, without subqueries ?
- Hint: show that all SFW queries are monotone (figure out what this means). A query with **ALL** is not monotone

Conditions on Tuples

```
SELECT  DISTINCT Company.name
FROM    Company, Product
WHERE   Company.name= Product.maker
        AND  (Product.name,price)  IN
             (SELECT Purchase.product, Purchase.price)
              FROM   Purchase
              WHERE Purchase.buyer = "Joe Blow");
```


Correlated Queries

Movie (title, year, director, length)
Find movies whose title appears more than once.

```
SELECT DISTINCT title
FROM   Movie AS x
WHERE  year < ANY
             (SELECT  year
              FROM    Movie
              WHERE  title =  x.title);
```

correlation

Note (1) scope of variables (2) this can still be expressed as single SFW

Complex Correlated Query

Product (pname, price, category, maker, year)

- Find products (and their manufacturers) that are more expensive than all products made by the same manufacturer before 1972

```
SELECT DISTINCT pname, maker
FROM   Product AS x
WHERE  price > ALL (SELECT price
                    FROM   Product AS y
                    WHERE  x.maker = y.maker AND y.year < 1972);
```

Powerful, but much harder to optimize !

13

Conserving Duplicates

The UNION, INTERSECTION and EXCEPT operators operate as sets, not bags.

```
(SELECT name
 FROM   Person
 WHERE  City="Seattle")

   UNION ALL

(SELECT name
 FROM   Person, Purchase
 WHERE  buyer=name AND store="The Bon")
```

14

Modifying the Database

Three kinds of modifications

- Insertions
- Deletions
- Updates

Sometimes they are all called "updates"

15

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.

16

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

17

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:

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

18

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	-	-

19

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

20

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.

21

Updates

Example:

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

22

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

23

Data Types in SQL

- Character strings (fixed of varying length)
- Bit strings (fixed or varying length)
- Integer (SHORTINT)
- Floating point
- Dates and times

Domains (=types) will be used in table declarations.

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

24

Creating Tables

Example:

```
CREATE    TABLE Person(

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

 );
```

25

Deleting or Modifying a Table

Deleting:

Example:

```
DROP Person;
```

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?

26

Default Values

Specifying default values:

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

The default of defaults: NULL

27

Indexes

REALLY important to speed up query processing time.

Suppose we have a relation

Person (name, age, city)

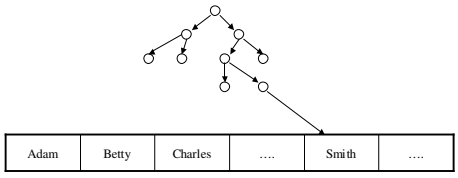
```
SELECT *
FROM    Person
WHERE   name = "Smith"
```

Sequential scan of the file Person may take long

28

Indexes

- Create an index on name:



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

29

Creating Indexes

Syntax:

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

30

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"
```

31

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?

32

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

33

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)

34

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"
```

35

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"
```

36

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

37

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)
```

38

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. One copy ? More copies ?

39