Lecture 05: SQL

Wednesday, October 9, 2002

Outline

- Indexes
- Defining Views (6.7)
- Constraints and Triggers (Chapter 7)

Indexes

REALLY important to speed up query processing time.

Suppose we have a relation

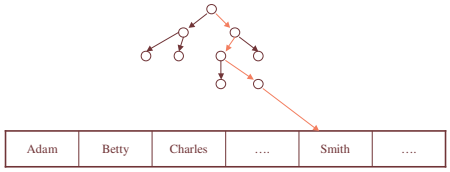
Person (name, age, city)

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

Sequential scan of the file Person may take long

Indexes

- Create an index on name:



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

Creating Indexes

Syntax:

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


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"
```

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?

7

The Index Selection Problem

- We are given a ***workload*** = a set of SQL queries plus how often they run
- What indexes should we build to speed up the workload ?
- FROM/WHERE clauses ➔ favor an index
- INSERT/UPDATE clauses ➔ discourage an index
- Index selection = normally done by people, recently done automatically (SQL Server)

8

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

9

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)

10

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"
```

11

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"
```

12

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

13

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(ssn, name, department, project, salary)
VALUES(NULL, "Joe", NULL, "Optimizer", NULL)
```

14

Non-Updatable Views

Person(name, city)
Purchase(buyer, seller, product, store)

```
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 don't know the Person.name and Purchase.buyer values; if we set them NULL they will not join.

15

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

16

Constraints in SQL

Constraints in SQL:

- Keys, foreign keys — simplest
- Attribute-level constraints
- Tuple-level constraints
- Global constraints: assertions — Most complex

The more complex the constraint, the harder it is to check and to enforce

17

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))
```

18

Keys with Multiple Attributes

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

19

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

20

Foreign Key Constraints

Referential integrity constraints

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

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

21

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

22

Foreign Key Constraints

- OR

```
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

23

What happens during updates ?

Types of updates:

- In Purchase: insert/update
- In Product: delete/update

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

24

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

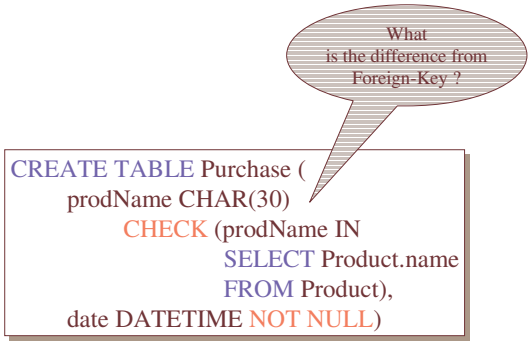
READING ASSIGNEMNT: 7.1.5, 7.1.6

25

Constraints on Attributes and Tuples

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

26



27

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)
```

28

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

29

Triggers in SQL

- A trigger contains an *event*, a *condition*, an *action*.
- Event = INSERT, DELETE, UPDATE
- Condition = any WHERE condition (may refer to the old and the new values)
- Action = more inserts, deletes, updates
- Many, many more bells and whistles...
- Read in the book (it only scratches the surface...)

30