Lecture 16: Views

Wednesday, March 25, 2015

Where We Are

- Today: Views and Quiz #5
- Next week: project presentations
- Should we have a “Best Demo Award”?

Views

- Views are like procedures in SQL
- They are defined by a SQL query
- They return a table of results from the SQL query

Example view:

Employees(ssn, first_name, last_name, role, title, salary)

```
CREATE VIEW  Senior_Staff  AS
   SELECT ssn, first_name, last_name, role, title, salary
   FROM  Employees
   WHERE title LIKE ‘%Senior%’
   ORDER BY salary
```

Senior_Staff(ssn, first_name, last_name, title, salary) = virtual table

We can now use the Senior_Staff view as if it were a table

Another View

Orders(order_id, customer_id, item_id, store)
Items(id, item_name, price)

```
CREATE VIEW  Customer_Sales  AS
        SELECT  o.customer_id, i.sale
        FROM     Orders o, Items i
        WHERE   o.item_id = i.id
```

Customer_Sales(customer_id, sale) = virtual table

Using the view:

```
SELECT  c.customer_id, c.sale, o.store
FROM     Customer_Sales c, Orders o
WHERE  c.customer_id = o.customer_id
AND       c.sale > 100
```

Question: How will this query be computed?

Query Modification

Using the view:

```
SELECT  c.customer_id, c.sale, o.store
FROM    Customer_Sales c, Orders o
WHERE   c.customer_id = o.customer_id
AND     c.sale > 100
```

Modified query (at runtime):

```
SELECT  c.customer_id, c.sale, o.store
FROM    (SELECT x.customer_id, y.sale,
         FROM Orders x, Items y
         WHERE x.item_id = y.id) c, Orders o
WHERE   c.customer_id = o.customer_id
AND     c.sale > 100
```

Another Use of the View

Orders(order_id, customer_id, item_id, store)
Items(id, item_name, price)

```
CREATE VIEW  Customer_Sales  AS
        SELECT  o.customer_id, o.store, i.sale
        FROM     Orders o, Items i
        WHERE   o.item_id = i.id
```

Customer_Sales(customer_id, sale) = virtual table

Using the view:

```
SELECT  c.customer_id
FROM     Customer_Sales
WHERE  c.store = ‘CVS’
```

Questions: Which table(s) will be used to answer this query?
Note that here we don’t want to inline the view definition. Why?

Types of Views

- Virtual views:
  - computed only on-demand
  - always up-to-date

- Materialized views:
  - pre-computed offline
  - requires extra storage
  - may be out-of-date with the base tables

Applications of Views

- Logical Data Independence
  (recall: Physical Data Independence)

- Optimizations
  - vertical partitioning
  - horizontal partitioning

- Security
  - controlled access to attributes and records

Vertical Partitioning

Students(eid, first_name, middle_initial, last_name)
Students_Photo(eid, photo, date_taken)

```
CREATE VIEW  Students_View  AS
        SELECT  s.eid, s.first_name, s.middle_initial,
                s.last_name, p.photo, p.date_taken
        FROM    Students s, Student_Photo p
        WHERE   s.eid = p.eid
```

Using the view:

```
SELECT  eid, middle_initial, last_name
FROM    Students_View
WHERE   first_name = ‘Kai’
```

Question: Which table(s) will be used to answer this query?

Horizontal Partitioning

Students(eid, first_name, middle_initial, last_name)
Students_Photo_2014(eid, photo, date_taken)
Students_Photo_2015(eid, photo, date_taken)

```
CREATE VIEW  Students_Photo_2014_2015  AS
        SELECT  eid, photo, date_taken
        FROM    Student_Photo_2014  UNION
        SELECT  eid, photo, date_taken
        FROM    Student_Photo_2015
```

Using the view:

```
SELECT  s.eid, s.first_name, s.middle_initial, s.last_name,
        p.photo, p.date_taken
FROM    Students s, Students_Photo_2014_2015 p
WHERE   s.eid = p.eid
AND     p.date_taken <= ‘15-SEP-2014’
```

Question: Which table(s) will be used to answer this query?

Security Views

Employees(ssn, first_name, last_name, role, title, salary)

```
CREATE VIEW  All_Employee_View  AS
   SELECT first_name, last_name, role, title
   FROM  Employees
   ORDER BY last_name, first_name
```

```
CREATE VIEW  HR_Employee_View  AS
   SELECT ssn, first_name, last_name, role, title, salary
   FROM  Employees
   WHERE role <> ‘Executive’
   ORDER BY last_name, first_name
```

Question: what data do these two views hide?

Quiz #5 (on Indexes)

Consider the following Movies table:

Movies(id NUMBER, name VARCHAR(64), year NUMBER, runtime NUMBER, rating NUMBER)

Assume that this table contains about 50 million records and it will be updated with new movie records as they are released.

In addition, there are six queries that run frequently on this table and that you are tasked with optimizing. These queries comprise the "typical" workload.

1. SELECT name FROM Movies WHERE year = 2015;
2. SELECT * FROM Movies WHERE year = 2015 AND rating BETWEEN 7 AND 10;
3. SELECT * FROM Movies WHERE rating = 10;
4. SELECT rating, COUNT(*) FROM Movies GROUP BY rating ORDER BY rating;

Quiz #5 (Continued)

5. SELECT DISTINCT year FROM Movies;
6. SELECT * FROM Movies;

For simplicity, assume that the frequency of all six queries is roughly the same.

For each SQL query, decide if a B+ tree index can be used to speed up the query and provide the create index statement for the suggested index. Try to reuse an index whenever it makes sense and avoid creating redundant indexes. If an index can't be used to speed up a given query, briefly state why and what access path should be used instead.

Next 3 Classes

- Project Presentations
- No quizzes :))