Materialized Views

March 26, 2018

```
CREATE VIEW salesSinceLastMonth AS
  SELECT l.*
  FROM lineitem l, orders o
  WHERE l.orderkey = o.orderkey
    AND o.orderdate > DATE('2015-03-31')
```

```
SELECT partkey FROM salesSinceLastMonth
ORDER BY shipdate DESC LIMIT 10;
```

```
SELECT suppkey, COUNT(*)
FROM salesSinceLastMonth
GROUP BY suppkey;
```

```
SELECT partkey, COUNT(*)
FROM salesSinceLastMonth
GROUP BY partkey;
```

```
CREATE VIEW salesSinceLastMonth AS
  SELECT l.*
  FROM lineitem l, orders o
  WHERE l.orderkey = o.orderkey
    AND o.orderdate > DATE('2015-03-31')
```

```
SELECT partkey FROM ordersSinceLastMonth
ORDER BY shipdate DESC LIMIT 10;
```

```
SELECT partkey FROM
  (
    SELECT l.*
    FROM lineitem l, orders o
    WHERE l.orderkey = o.orderkey
      AND o.orderdate > DATE('2015-03-31')
  ) AS salesSinceLastMonth
ORDER BY shipdate DESC LIMIT 10;
```

Views

- … contain and abstract complex concepts.
  - Complex query patterns can be given a shorthand.
  - It’s easier to change view logic “in the background”
- … act as normal relations.
  - View references can be expanded inline into nested subqueries.
  - Updates are trickier….

View Updates

What happens when we Insert Into/Update a view?

View Updates

```
UPDATE salesSinceLastMonth
   SET statusCode = ‘q’;
 WHERE orderkey = 22;
```

Rows in `salesSinceLastMonth` correspond 1-1 with rows in `lineitem`. Update `lineitem`!

View Updates

```
INSERT INTO salesSinceLastMonth
  (orderkey, partkey, suppkey, …)
VALUES
  (22, 99, 42, …);
```

Lots of problems…

- What if order # 22 doesn't exist?
- How does the insertion interact with sequences (e.g., `lineitem.lineno`)

View Updates

Solution 1: Data Integration
(CSE 636)

Solution 2: INSTEAD OF triggers

View Updates

```
CREATE TRIGGER salesSinceLastMonthInsert
INSTEAD OF INSERT ON salesSinceLastMonth
REFERENCING NEW ROW AS newRow
FOR EACH ROW
  IF NOT EXISTS (
     SELECT * FROM ORDERS
     WHERE ORDERS.orderkey = newRow.orderKey)
  ) THEN
    INSERT INTO ORDERS(orderkey)
      VALUES (orderkey)
  END IF;
  INSERT INTO LINEITEM VALUES newRow;
END FOR;
```

Can we use views for anything else?

Materialization

Views exist to be queried frequently

Pre-compute and save the view’s contents!
(like an index)

Materialization Challenges

- How do we maintain the views as data changes?
- What if the view is not explicitly referenced?
- What views should be materialized?

Delta Queries

- If D is your Database and Q is your Query:
    - Q(D) is the result of your query on the database.
- Let's say you make a change ΔD (Insert tuple)
    - Q(D+ΔD) is the new result
- If we have Q(D), can we get Q(D+ΔD) faster?
    - Analogy to Sum: {34, 29, 10, 15} + {12} (88+12)

Query Rewriting

```
CREATE MATERIALIZED VIEW salesSinceLastMonth AS
  SELECT l.*
  FROM lineitem l, orders o
  WHERE l.orderkey = o.orderkey
    AND o.orderdate > DATE(’2015-03-31’)
```

```
SELECT l.partkey
FROM lineitem l, orders o
WHERE l.orderkey = o.orderkey
  AND o.orderdate > DATE(’2015-03-31’)
ORDER BY l.shipdate DESC
LIMIT 10;
```

We can use a materialized view to speed the query up

Query Rewriting

View Query

SELECT L_V
FROM R_V
WHERE C_V

User Query

SELECT L_Q
FROM R_Q
WHERE C_Q

When are we allowed to rewrite this query?

Query Rewriting

View Query

SELECT L_V
FROM R_V
WHERE C_V

User Query

SELECT L_Q
FROM R_Q
WHERE C_Q

$$R_V \subseteq R_Q$$

All relations in the view are part of the query join

$$C_Q = C_V \wedge C'$$

The view condition is weaker than the query condition

$$attrs(C') \cap attrs(R_V) \subseteq L_V \qquad L_Q \cap attrs(R_V) \subseteq L_V$$

The view doesn’t project away needed attributes

Query Rewriting

View Query

SELECT L_V
FROM R_V
WHERE C_V

User Query

SELECT L_Q
FROM R_Q
WHERE C_Q

What does the query rewrite to?

Query Rewriting

View Query

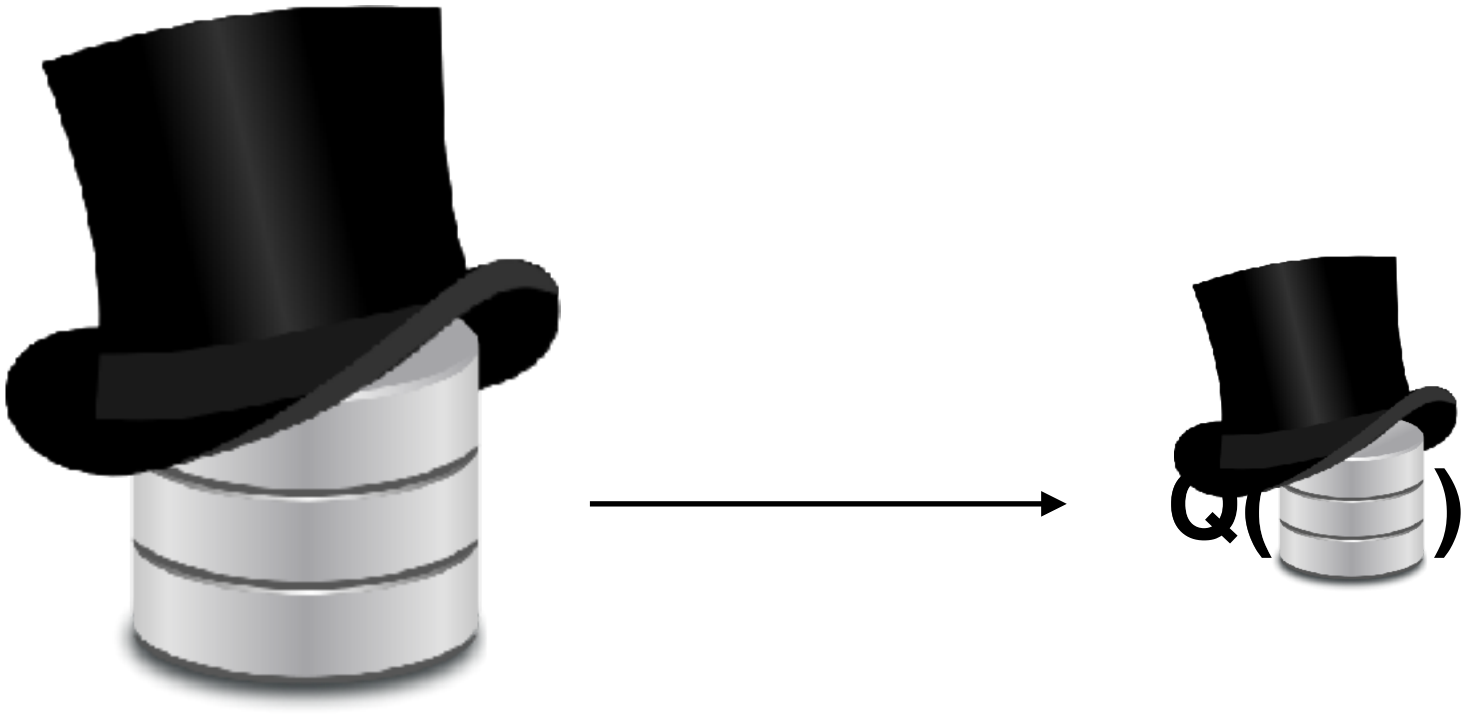
SELECT L_V
FROM R_V
WHERE C_V

User Query

SELECT L_Q
FROM R_Q
WHERE C_Q

SELECT L_Q
FROM $(R_Q{-}R_V)$, VIEW
WHERE C_Q

Materialized Views



When the base data changes, the view needs to be updated

View Maintenance

```
VIEW ← Q(D)
```

View Maintenance

```
WHEN D ← D+ΔD DO:
  VIEW ← Q(D+ΔD)
```

Re-evaluating the query from scratch is expensive!

View Maintenance

(ideally) Smaller & Faster Query

```
WHEN D ← D+ΔD DO:
  VIEW ← VIEW+ΔQ(D,ΔD)
```

(ideally) Fast “merge” operation.

Intuition

$$D = \{1, 2, 3, 4\} \qquad \Delta D = \{5\}$$

$$Q(D) = SUM(D)$$

$$Q(D+\Delta D) \sim O(|D|+|\Delta D|)$$

$$VIEW + SUM(\Delta D) \sim O(|\Delta D|)$$

Intuition

$R = \{1, 2, 3\}, S = \{5,6\}$ $\Delta R = \{4\}$

$Q(R,S) = \mathrm{COUNT}(R \mathbf{x} S)$

$Q(R+\Delta R,S) \sim O((|R|+|\Delta R|) * |S|)$

$\mathrm{VIEW} + \mathrm{COUNT}(|\Delta R|*|S|) \sim O(|\Delta R|*|S|)$

Intuition

+ ~ U

*** ~ x**

Are these kinds of patterns common?

Rings/Semirings

This kind of pattern occurs frequently.

Semiring : $< S, +, x, S_0, S_1 >$

Any set of ‘things’ **S** such that…

Closed

$$S_i + S_j = S_k$$

$$S_i \text{ x } S_j = S_k$$

$$S_i + S_0 = S_i$$

$$S_i \text{ x } S_1 = S_i$$

$$S_i \text{ x } S_0 = S_0$$

Additive &
Multiplicative
“zeroes”

$$S_i \text{ x } (S_j + S_k) = (S_i \text{ x } S_j) + (S_j \text{ x } S_k)$$

Distributive

Rings/Semirings

Ring : < S, +, x, S_0, S_1, - >

Any semiring where every element has an additive inverse…

$$S_i + (-S_i) = S_0$$

THE TANGENT ENDS NOW