Views

April 6
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;
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;

SELECT l.partkey, COUNT(*)
FROM lineitem l, orders o
WHERE l.orderkey = o.orderkey
    AND o.orderdate > DATE('2015-03-31')
GROUP BY l.partkey;

SELECT l.suppkey, COUNT(*)
FROM lineitem l, orders o
WHERE l.orderkey = o.orderkey
    AND o.orderdate > DATE('2015-03-31')
GROUP BY l.suppkey;
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;

“orders since last month”

SELECT l.partkey, COUNT(*)
FROM lineitem l, orders o
WHERE l.orderkey = o.orderkey
    AND o.orderdate > DATE('2015-03-31')
GROUP BY l.partkey;

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

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?
UPDATE salesSinceLastMonth
    SET statusCode = 'q';
WHERE orderkey = 22;
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, supplkey, ...)  
VALUES  
    (22, 99, 42, ...);
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.lineneno)
View Updates
View Updates

**Solution 1:** Data Integration
View Updates

**Solution 1:** Data Integration
(CSE 636)
View Updates

**Solution 1:** Data Integration  
(CSE 636)

**Solution 2:** INSTEAD OF triggers
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)
CREATE MATERIALIZED VIEW salesSinceLastMonth AS
SELECT l.*
FROM lineitem l, orders o
WHERE l.orderkey = o.orderkey
AND o.orderdate > DATE('2015-03-31')
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;
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

<table>
<thead>
<tr>
<th>View Query</th>
<th>User Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT ( L_v ) FROM ( R_v ) WHERE ( C_v )</td>
<td>SELECT ( L_Q ) FROM ( R_Q ) WHERE ( C_Q )</td>
</tr>
</tbody>
</table>

When are we allowed to rewrite this query?
Query Rewriting

View Query

\[
\text{SELECT } L_V \\
\text{FROM } R_V \\
\text{WHERE } C_V
\]

User Query

\[
\text{SELECT } L_Q \\
\text{FROM } R_Q \\
\text{WHERE } C_Q
\]

\[R_V \subseteq R_Q\]
All relations in the view are part of the query join

\[C_Q = C_V \land C'\]
The view condition is weaker than the query condition

\[\text{attrs}(C') \cap \text{attrs}(R_V) \subseteq L_V\]
\[L_Q \cap \text{attrs}(R_V) \subseteq L_V\]
The view doesn’t project away needed attributes
What does the query rewrite to?
Query Rewriting

<table>
<thead>
<tr>
<th>View Query</th>
<th>User Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT $L_V$</td>
<td>SELECT $L_Q$</td>
</tr>
<tr>
<td>FROM $R_V$</td>
<td>FROM $R_Q$</td>
</tr>
<tr>
<td>WHERE $C_V$</td>
<td>WHERE $C_Q$</td>
</tr>
</tbody>
</table>

SELECT $L_Q$
FROM ($R_Q$ - $R_V$), VIEW
WHERE $C_Q$
Materialized Views
Materialized Views
Materialized Views
Materialized Views

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

VIEW ← Q(D)
View Maintenance

WHEN D ← D+\Delta D DO:
  VIEW ← Q(D+\Delta D)
View Maintenance

\[
\text{WHEN } D \leftarrow D + \Delta D \text{ DO: }
\]
\[
\text{VIEW } \leftarrow Q(D + \Delta D)
\]

Re-evaluating the query from scratch is expensive!
View Maintenance

\[
\text{WHEN } D \leftarrow D + \Delta D \text{ DO: }
\]

\[
\text{VIEW } \leftarrow \text{VIEW} + \Delta \mathcal{Q}(D, \Delta D)
\]
View Maintenance

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

(ideally) Smaller & Faster Query
View Maintenance

\[
\text{WHEN } D \leftarrow D + \Delta D \text{ DO:} \\
\text{VIEW } \leftarrow \text{VIEW} + \Delta Q(D, \Delta D)
\]

(ideally) Smaller & Faster Query

(ideally) Fast “merge” operation.
Intuition

\[ D = \{1, 2, 3, 4\} \quad \Delta D = \{5\} \]

\[ Q(D) = \text{SUM}(D) \]
Intuition

\[ D = \{1, 2, 3, 4\} \quad \Delta D = \{5\} \]

\[ Q(D) = \text{SUM}(D) \]

\[ Q(D + \Delta D) \sim O(|D| + |\Delta D|) \]
Intuition

\[ D = \{1, 2, 3, 4\} \quad \Delta D = \{5\} \]

\[ Q(D) = \text{SUM}(D) \]

\[ Q(D + \Delta D) \sim O( |D| + |\Delta D| ) \]

\[ \text{VIEW} + \text{SUM}(\Delta D) \sim O( |\Delta D| ) \]
**Intuition**

\[ R = \{1, 2, 3\}, \quad S = \{5, 6\} \quad \Delta R = \{4\} \]

\[ Q(R, S) = \text{COUNT}(R \times S) \]
Intuition

\[ R = \{1, 2, 3\}, \quad S = \{5, 6\} \quad \Delta R = \{4\} \]

\[ Q(R, S) = \text{COUNT}(R \times S) \]

\[ Q(R+\Delta R, S) \sim O\left( (|R| + |\Delta R|) \times |S| \right) \]
Intuition

\( R = \{1, 2, 3\}, \quad S = \{5, 6\} \quad \Delta R = \{4\} \)

\[ Q(R,S) = \text{COUNT}(R \times S) \]

\[ Q(R+\Delta R,S) \sim O\left( (|R|+|\Delta R|) \times |S| \right) \]

\[ \text{VIEW} + \text{COUNT}(|\Delta R| \times |S|) \sim O(|\Delta R| \times |S|) \]
Intuition

+ ~ U

* ~ X
Intuition

\[ + \sim U \]
\[ \ast \sim X \]

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…

\[
\begin{align*}
S_i + S_j &= S_k \\
S_i \times S_j &= S_k \\
S_i \times (S_j + S_k) &= (S_i \times S_j) + (S_j \times S_k)
\end{align*}
\]

Closed

Additive & Multiplicative

“zeroes”

\[
\begin{align*}
S_i + S_0 &= S_i \\
S_i \times S_1 &= S_i \\
S_i \times S_0 &= S_0
\end{align*}
\]

Distributive
Rings/Semirings

Ring : < S, +, x, S₀, S₁, - >

Any semiring where every element has an additive inverse...

Sᵢ + (-Sᵢ) = S₀
THE TANGENT ENDS NOW
Incremental View Maintenance

\[
\text{WHEN } D \leftarrow D + \Delta D \text{ DO:}
\]
\[
\text{VIEW} \leftarrow \text{VIEW} + \Delta Q(D, \Delta D)
\]
Incremental View Maintenance

\[
\text{WHEN } D \leftarrow D + \Delta D \text{ DO:} \\
\text{VIEW } \leftarrow \text{VIEW} + \Delta Q(D, \Delta D)
\]

Basic Challenges of IVM

What does \( \Delta R \) represent?
Incremental View Maintenance

WHEN D ← D + ΔD DO:

VIEW ← VIEW + ΔQ(D, ΔD)

Basic Challenges of IVM

What does ΔR represent?
How to interpret R ± ΔR?
Incremental View Maintenance

WHEN $D \leftarrow D + \Delta D$ DO:

$\text{VIEW} \leftarrow \text{VIEW} + \Delta Q(D, \Delta D)$

Basic Challenges of IVM

What does $\Delta R$ represent?

How to interpret $R \pm \Delta R$?

How to compute $\Delta Q$?
What is $\Delta R$?
What is $\Delta R$?

What does it need to represent?
What is $\Delta R$?

What does it need to represent?

- Insertions
- Deletions
- Updates
What is $\Delta R$?

What does it need to represent?

- Insertions
- Deletions
- Updates

(Delete Old Record & Insert Updated Record)
What is $\Delta R$?
What is $\Delta R$?

A Set/Bag of Insertions

A Set/Bag of Deletions
What is $+$?

$R$

A Set/Bag

$\Delta R$

A Set/Bag of Insertions

A Set/Bag of Deletions
What is $+$?

$R + \Delta R$

A Set/Bag of Insertions

A Set/Bag of Deletions
What is +?

\[ R + \Delta R \]

A Set/Bag of Insertions

\[ R \cup \Delta R_{\text{inserted}} \]

A Set/Bag of Deletions

\[ \Delta R_{\text{deleted}} \]
What is $+$?

$R + \Delta R$

$\Delta R_{\text{inserted}}$

$\Delta R_{\text{deleted}}$

A Set/Bag of Insertions

A Set/Bag of Deletions

But this breaks closure of ‘$+$’!
Incremental View Maintenance

VIEW ← VIEW + ΔQ(D, ΔD)
Incremental View Maintenance

\[
\text{VIEW} \leftarrow \text{VIEW} + \Delta Q(D, \Delta D)
\]
Incremental View Maintenance

\[
\text{VIEW} \leftarrow \text{VIEW} + \Delta Q(D, \Delta D)
\]

Given \(Q(R, S, \ldots)\)

Construct \(\Delta Q(R, \Delta R, S, \Delta S, \ldots)\)
Delta Queries

\[ \Delta(\sigma(R)) \]

\[ \sigma \]

\[ R \]
Delta Queries

\[ \Delta(\sigma(R)) \]

\( \sigma \)

\[ R \]

Original \( R \)

\[ R \]

\[ \Delta R \]

Inserted Tuples of \( R \)
Delta Queries

\[ \Delta(\sigma(R)) \]

\[ \sigma \quad \sigma \]

\[ \delta \quad \delta \]

Original R

Inserted Tuples of R
Delta Queries

\[ \Delta(\sigma(R)) = \sigma(\Delta R) \]

\[ \sigma \]

Original R

\[ \sigma \]

Inserted Tuples of R
Delta Queries

$$\Delta(\sigma(R)) = \sigma(\Delta R)$$

Original $R$ \quad Inserted Tuples of $R$

Does this work for deleted tuples?
Delta Queries

$\Delta(\pi(R)) = \pi(\Delta R)$
Delta Queries

\[ \Delta(\pi(R)) = \pi(\Delta R) \]

\[ \begin{array}{ccc}
\pi & & \pi \\
\mid & & \mid \\
R & & R & \Delta R
\end{array} \]

Does this work (completely) under set semantics?
Delta Queries

\[ \Delta(R_1 \cup R_2) \]

\[
\begin{array}{c}
U \\
\downarrow \\
R_1 \quad R_2 \\
\end{array}
\]

\[
\begin{array}{c}
R_1 \\
\Delta R_1 \\
R_2 \\
\Delta R_2 \\
\end{array}
\]
Delta Queries

\[ \Delta(R_1 \cup R_2) = \Delta R_1 \cup \Delta R_2 \]
Delta Queries

\[ \Delta(R_1 \cup R_2) = \Delta R_1 \cup \Delta R_2 \]
Delta Queries
Delta Queries

\[ R : \{1, 2, 3\} \quad S : \{5, 6\} \]
Delta Queries

\[ R : \{ 1, 2, 3 \} \quad S : \{ 5, 6 \} \]

\[ R \times S = \{ <1,5>, <1, 6>, <2,5>, <2,6>, <3,5>, <3,6> \} \]
Delta Queries

\[ R : \{ 1, 2, 3 \} \quad S : \{ 5, 6 \} \]

\[ R \times S = \{ <1,5>, <1, 6>, <2,5>, <2,6>, <3,5>, <3,6> \} \]

\[ \Delta R_{\text{inserted}} = \{ 4 \} \]

\[ \Delta R_{\text{deleted}} = \{ 3, 2 \} \]
Delta Queries

\[ R : \{ 1, 2, 3 \} \quad S : \{ 5, 6 \} \]

\[ R \times S = \{ <1,5>, <1, 6>, <2,5>, <2,6>, <3,5>, <3,6> \} \]

\[ \Delta R_{\text{inserted}} = \{ 4 \} \]

\[ \Delta R_{\text{deleted}} = \{ 3, 2 \} \]

\[ (R + \Delta R) \times S = \{ <1,5>, <1, 6>, <4,5>, <4,6> \} \]
Delta Queries

\[ R : \{ 1, 2, 3 \} \quad S : \{ 5, 6 \} \]

\[ R \times S = \{ <1,5>, <1,6>, <2,5>, <2,6>, <3,5>, <3,6> \} \]

\[ \Delta R_{\text{inserted}} = \{ 4 \} \]
\[ \Delta R_{\text{deleted}} = \{ 3, 2 \} \]

\[ (R+\Delta R) \times S = \{ <1,5>, <1,6>, <4,5>, <4,6> \} \]

\[ \Delta_{\text{inserted}}(R \times S) = \Delta R_{\text{inserted}} \times S \]
\[ \Delta_{\text{deleted}}(R \times S) = \Delta R_{\text{deleted}} \times S \]
Delta Queries

\[ R : \{ 1, 2, 3 \} \quad S : \{ 5, 6 \} \]

\[ R \times S = \{ <1, 5>, <1, 6>, <2, 5>, <2, 6>, <3, 5>, <3, 6> \} \]

\[ \Delta R_{\text{inserted}} = \{ 4 \} \]

\[ \Delta R_{\text{deleted}} = \{ 3, 2 \} \]

\[ (R + \Delta R) \times S = \{ <1, 5>, <1, 6>, <4, 5>, <4, 6> \} \]

\[ \Delta_{\text{inserted}}(R \times S) = \Delta R_{\text{inserted}} \times S \]

\[ \Delta_{\text{deleted}}(R \times S) = \Delta R_{\text{deleted}} \times S \]

What if \( R \) and \( S \) both change?
Delta Queries

Computing a Delta Query

\[ \Delta(\sigma(R)) = \sigma(\Delta R) \]

\[ \Delta(\pi(R)) = \pi(\Delta R) \]

\[ \Delta(R_1 \cup R_2) = \Delta R_1 \cup \Delta R_2 \]

\[ \Delta(R_1 \times R_2) = ?? \]
Delta Queries

$$(R_1 \cup \Delta R_1) \times (R_2 \cup \Delta R_2)$$
Delta Queries

\[(R_1 \cup \Delta R_1) \times (R_2 \cup \Delta R_2)\]

\[(R_1 \times R_2) \cup (R_1 \times \Delta R_2) \cup (\Delta R_1 \times R_2) \cup (\Delta R_1 \times \Delta R_2)\]
Delta Queries

\[(R_1 \cup \Delta R_1) \times (R_2 \cup \Delta R_2)\]

\[(R_1 \times R_2) \cup (R_1 \times \Delta R_2) \cup (\Delta R_1 \times R_2) \cup (\Delta R_1 \times \Delta R_2)\]

The original query
Delta Queries

\[(R_1 \cup \Delta R_1) \times (R_2 \cup \Delta R_2)\]

\[(R_1 \times R_2) \cup (R_1 \times \Delta R_2) \cup (\Delta R_1 \times R_2) \cup (\Delta R_1 \times \Delta R_2)\]

The original query

The delta query
How about an example…
Delta Queries

Let’s say you have an insertion into LINEITEM
Delta Queries

\[ \Delta((\sigma(C) \bowtie \sigma(O)) \bowtie (\sigma(L))) \]
Delta Queries

\[ \Delta((\sigma(C) \bowtie \sigma(O)) \bowtie (\sigma(L))) \]
Delta Queries

\[ \Delta((\sigma(C) \bowtie \sigma(O)) \bowtie (\sigma(L))) = \emptyset \]
Delta Queries

\[ ((\sigma(C) \bowtie \sigma(O)) \bowtie \Delta(\sigma(L))) \]
Delta Queries

\[ \sigma_{\text{mktsegment}} \]
\[ \sigma_{\text{orderdate}} \]
\[ \sigma_{\text{shipdate}} \]

CUSTOMER
ORDERS
LINEITEM
SELECT *
FROM CUSTOMER C, ORDERS O, DELTA_LINEITEM DL
WHERE C.custkey = O.custkey
  AND DL.orderkey = O.orderkey
  AND C.mktsegment = ...
  AND O.orderdate = ...
  AND DL.shipdate = ...
Multisets

\[ \{ 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 4, 4, 4, 4, 4, 4, 5 \} \]
(not compact)

\[ \{ 1 \rightarrow x3, 2 \rightarrow x5, 3 \rightarrow x2, 4 \rightarrow x6, 5 \rightarrow x1 \} \]
Multiset representation: Tuple \( \rightarrow \) # of occurrences
Multisets

\{ 1, 1, 1, 2, 2, 2, 2, 3, 3, 4, 4, 4, 4, 4, 5 \} (not compact)

\{ 1 \rightarrow x3, 2 \rightarrow x5, 3 \rightarrow x2, 4 \rightarrow x6, 5 \rightarrow x1 \}  
Multiset representation: Tuple \rightarrow \# of occurrences
Multiset Deltas

Insertions = Positive Multiplicity

Deletions = Negative Multiplicity

+ = Bag/Multiset Union