Views
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;
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;

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

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

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;
D: Database
Q: Query
Q(D): Result

ΔD: Change
Insert
Delete
Update

We have: Q(D), ΔD

We want: Q(D + ΔD)

\[ \text{we have fast} \]

\[ Q(D) + ΔQ(D, ΔD) \text{ fast} \]
QD: $R \Rightarrow R \cup \neg Q$

QAR: $Q(R \lor Q)$

$Q \land R \quad Q(D \lor D) : R \lor \neg Q$

Q(D): \text{TS}

Q(D): \text{SS}

Q(D): \text{SS'}

Q(D): \text{SS' SS'}

Q(D): \text{SS'}

Q(D': \text{SS' SS'}

Q(D': \text{SS'})

Q(D': \text{SS'})
DB: R → R ADR

G(R) → (R + R)

G: R (o (0 + 0)) = f (0)

\[
\frac{2}{\frac{1}{2} \times \frac{1}{2}} = \frac{1}{4} = \frac{2}{2} \times \frac{2}{2}
\]

G(0): 2 42
G(0): 2 42
G(0): 2 42
G(0): 2 42
G(0): 2 42
G(0): 2 42
G(0): 2 42
G(0): 2 42
G(0): 2 42
Also works for:

\[
\sqrt{\frac{(SD \times SQ)(A + (S \times SQ))}{A + (SD \times S)}}
\]
\(D = \text{Insert info}\)
\[ D = I_{n \times n} \triangle \]

\[ \text{Insert initial} \]

\[ (0, 0, 0) \rightarrow (0, 0, 0) \]
CREATE MATERIALIZED VIEW salesSinceLastMonth AS
SELECT 1.*
FROM lineitem 1, orders o
WHERE 1.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;

UPDATE salesSinceLastMonth
SET statuscode='CL' WHERE orderkey=22
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;
Motivation — Why are Views Useful?

- Give an example query:

  - Workloads often have repeating patterns:

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

- View Definition

  - 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 supplykey, COUNT(*)
    FROM salesSinceLastMonth
    GROUP BY supplykey;

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

Definition — What is a View / How are they used?

- Views act as normal relations

  - SELECT partkey FROM salesSinceLastMonth
    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')) salesSinceLastMonth
    ORDER BY shipdate DESC LIMIT 10;
Views contain and abstract concepts

- Analogous to a function
- Complex query patterns can be given an shorthand
- Can freely change view logic “in the background” (Change ‘last month’)

- But not quite normal relations...

**View Updates**

- UPDATE salesSinceLastMonth
  SET statusCode = 'q';
  WHERE orderkey = 22;

  - Easy... rows in salesSinceLastMonth go 1-1 with LINEITEM.
  - Can find the row of line item that matches a given row of salesSinceLastMonth and update it.

- INSERT INTO salesSinceLastMonth
  (orderkey, partkey, suppkey, ...)
  VALUES
  (22, 99, 42, ...);

  - Harder...
  - What happens if order #22 doesn’t exist?
  - How does the insertion interact with sequences (e.g., Lineitem.lineno)

- 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;

  - InsteadOf triggers update rows

**View Materialization**

- Views exist because they’re queried frequently...

- Why not use them to make computations faster.
  - Precompute (materialize) the view’s contents (like an index)

- Challenges:
• What happens when the data behind the view changes?
• What happens when the view definition changes?
• What happens when we write a query without realizing we have a view?

▼ Updates to Materialized Views

▼ Let’s say you have a database D and a query Q
  • Q(D) is the result of your query on the database

▼ Let’s say you make a change ΔD (e.g., 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)

▼ Specific query examples
  • Projection
  • Selection
  • Union
  • Cross-Product
  • Aggregation

▼ Interactions with...
  • Insert
  • Delete
  • Update

▼ View Selection

▼ Can we use materialized views without knowing about them?

• 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;

▼ Simplify the query model:

• View: SELECT Lv FROM Rv WHERE Cv

• Query: SELECT Lq FROM Rq WHERE Cq
• When can we rewrite this query?
  • $R_v \subseteq R_q$ (All relations in the view are in the query join)
  • $C_q = C_v \land C'$ (The view condition is weaker than the query condition)
  • $L_q \cap \text{attrs}(R_v) \subseteq L_v$ (The view doesn’t project away attributes needed for the output)
  • $\text{attrs}(C') \cap \text{attrs}(R_v) \subseteq L_v$ (The view doesn’t project away attributes needed for the condition)

  • The whole thing rewrites to:
    • SELECT $L_q$ FROM $(R_q - R_v)$, view WHERE $C'$

• Views for Transactions
Incremental View Maintenance

Not covered by Database Systems: TCB
Materialized Views

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

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

\[ \text{VIEW} \leftarrow Q(D) \]
View Maintenance

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

VIEW $\leftarrow Q(D + \Delta D)$

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

WHEN \( D \leftarrow D + \Delta D \) DO:

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

(ideally) Smaller & Faster Query

(ideally) Fast “merge” operation.
Intuition

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

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

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

\[ \text{VIEW} + \text{SUM}(\Delta D) \sim O(\mid \Delta D \mid) \]
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

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…

- $S_i + S_j = S_k$
- $S_i \times S_j = S_k$
- $S_i + S_0 = S_i$
- $S_i \times S_1 = S_i$
- $S_i \times S_0 = S_0$
- $S_i \times (S_j + S_k) = (S_i \times S_j) + (S_j \times S_k)$

Closed

Additive & Multiplicatively

“zeroes”

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
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 does it need to represent?

- Insertions
- Deletions
- Updates

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

A Set/Bag of Insertions

A Set/Bag of Deletions
What is +?

\[ R \quad + \quad \Delta R \]

A Set/Bag of Insertions

A Set/Bag of Deletions

\[ R \quad \cup \quad \Delta R_{\text{inserted}} \]
\[ - \quad \Delta R_{\text{deleted}} \]

But this breaks closure of ‘+’!
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

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

$\sigma$

$\sigma$

$R$

$R$

$\Delta R$

Original $R$

Inserted Tuples of $R$

Does this work for deleted tuples?
Delta Queries

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

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

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

\[
\text{X} \\
\text{R} \quad \text{S} \quad \text{R} \quad \Delta R \quad \text{S}
\]
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

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

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

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

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

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

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

\[ \text{CUSTOMER} \]
\[ \text{ORDERS} \]
\[ \text{LINEITEM} \]
Delta Queries

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, 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 $\rightarrow$ multiplicity
Multiset Deltas

Insertions = Positive Multiplicity

Deletions = Negative Multiplicity

+ = Bag/Multiset Union
Multiset Deltas

What does Union do?

\{ A \mapsto 1, B \mapsto 3 \} \cup \{ B \mapsto 2, C \mapsto 4 \} = \{ A \mapsto 1, B \mapsto 5, C \mapsto 4 \}

\{ A \mapsto 1 \} \cup \{ A \mapsto -1 \} = \{ A \mapsto 0 \} = \{ \}
Multiset Deltas

What does Union do?

\[
\{ \text{A} \rightarrow 1, \text{B} \rightarrow 3 \} \cup \{ \text{B} \rightarrow 2, \text{C} \rightarrow 4 \} = \{ \text{A} \rightarrow 1, \text{B} \rightarrow 5, \text{C} \rightarrow 4 \}
\]

\[
\{ \text{A} \rightarrow 1 \} \cup \{ \text{A} \rightarrow -1 \} = \{ \text{A} \rightarrow 0 \} = \{
\}
\]

What does Cross Product do?

\[
\{ \text{A} \rightarrow 1, \text{B} \rightarrow 3 \} \times \{ \text{C} \rightarrow 4 \} = \{ \langle \text{A,C} \rangle \rightarrow ?, \langle \text{B,C} \rangle \rightarrow ? \}
\]
Multiset Deltas

What does Union do?

\{ A \rightarrow 1, B \rightarrow 3 \} \cup \{ B \rightarrow 2, C \rightarrow 4 \} = \{ A \rightarrow 1, B \rightarrow 5, C \rightarrow 4 \}

\{ A \rightarrow 1 \} \cup \{ A \rightarrow -1 \} = \{ A \rightarrow 0 \} = \{ \}

What does Cross Product do?

\{ A \rightarrow 1, B \rightarrow 3 \} \times \{ C \rightarrow 4 \} = \{ <A,C> \rightarrow 4, <B,C> \rightarrow ? \}
Multiset Deltas

What does Union do?
\[ \{ A \rightarrow 1, B \rightarrow 3 \} \cup \{ B \rightarrow 2, C \rightarrow 4 \} = \{ A \rightarrow 1, B \rightarrow 5, C \rightarrow 4 \} \]
\[ \{ A \rightarrow 1 \} \cup \{ A \rightarrow -1 \} = \{ A \rightarrow 0 \} = \{ \} \]

What does Cross Product do?
\[ \{ A \rightarrow 1, B \rightarrow 3 \} \times \{ C \rightarrow 4 \} = \{ <A,C> \rightarrow 4, <B,C> \rightarrow 12 \} \]
Multiset Deltas

What does projection do?

\[ \pi_{\text{Attr}1} \{ <A,X> \rightarrow 1, <A,Y> \rightarrow 2, <B,Z> \rightarrow 5 \} \]

\[ = \{ <A> \rightarrow 1, <A> \rightarrow 2, <B> \rightarrow 5 \} \]

\[ = \{ <A> \rightarrow 3, <B> \rightarrow 5 \} \]

This effect seems... familiar
If you find this subject interesting… let’s chat.

http://www.dbtoaster.org