Views

April 6

```sql
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;
```

```sql
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;
```

```sql
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"

```sql
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;
```

```sql
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 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;
```

```
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')
```

```
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;
```

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, …);
```

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

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

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')
```

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

View Query	User Query
SELECT L_V	SELECT L_Q
FROM R_V	FROM R_Q
WHERE C_V	WHERE C_Q

When are we allowed to rewrite this query?

Query Rewriting

View Query

```
SELECT LV
FROM RV
WHERE CV
```

User Query

```
SELECT LQ
FROM RQ
WHERE CQ
```

$$R_V \subseteq R_Q$$

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 LV
FROM RV
WHERE CV
```

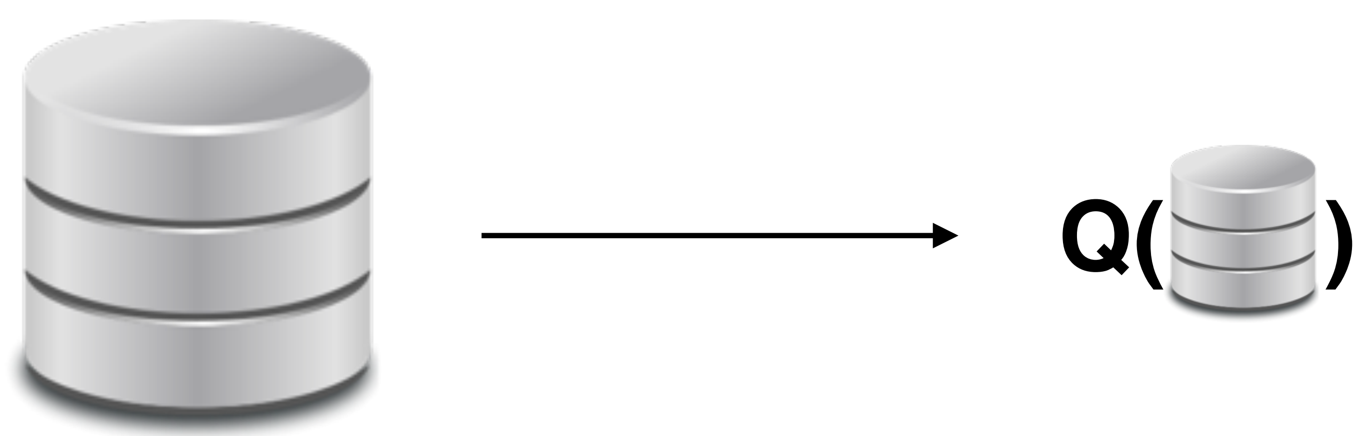
User Query

```
SELECT LQ
FROM RQ
WHERE CQ
```

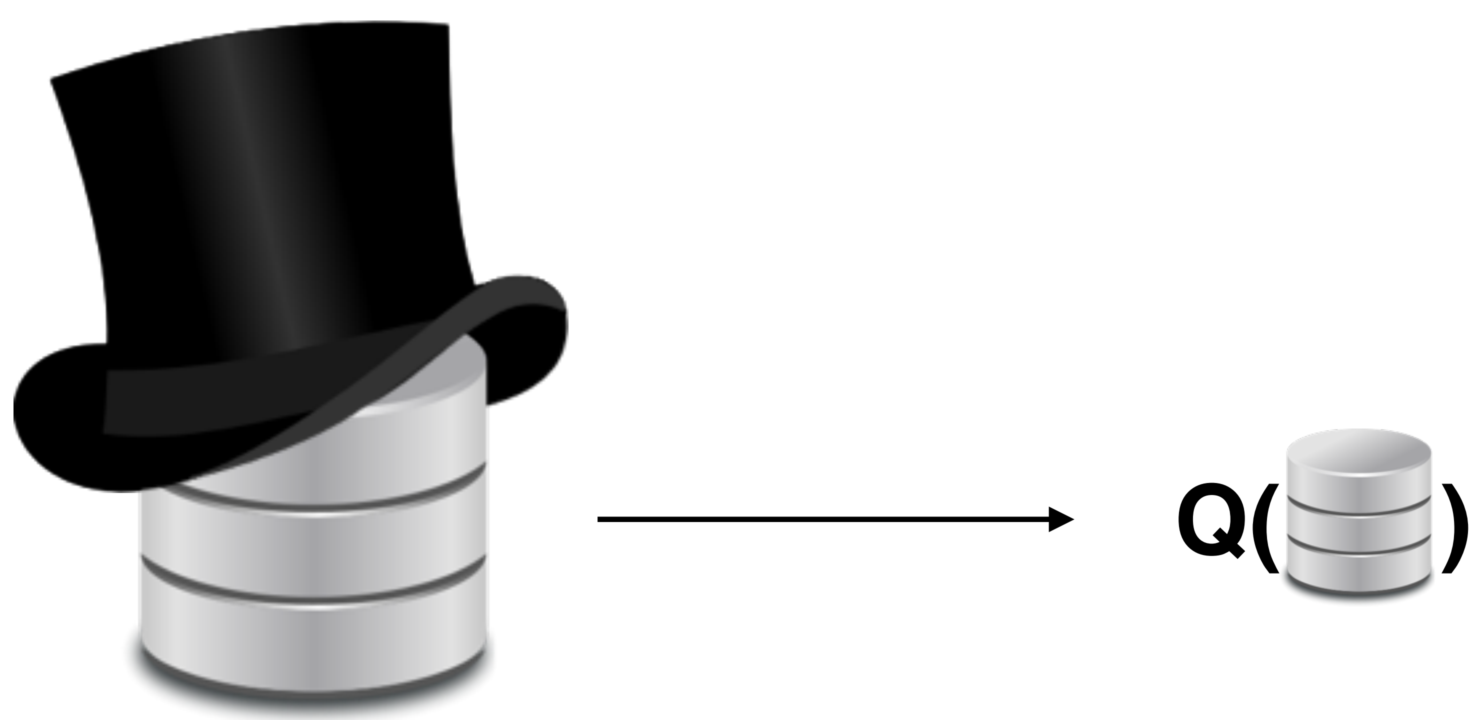
```
SELECT LQ
FROM (RQ-RV), VIEW
WHERE CQ
```

Materialized Views

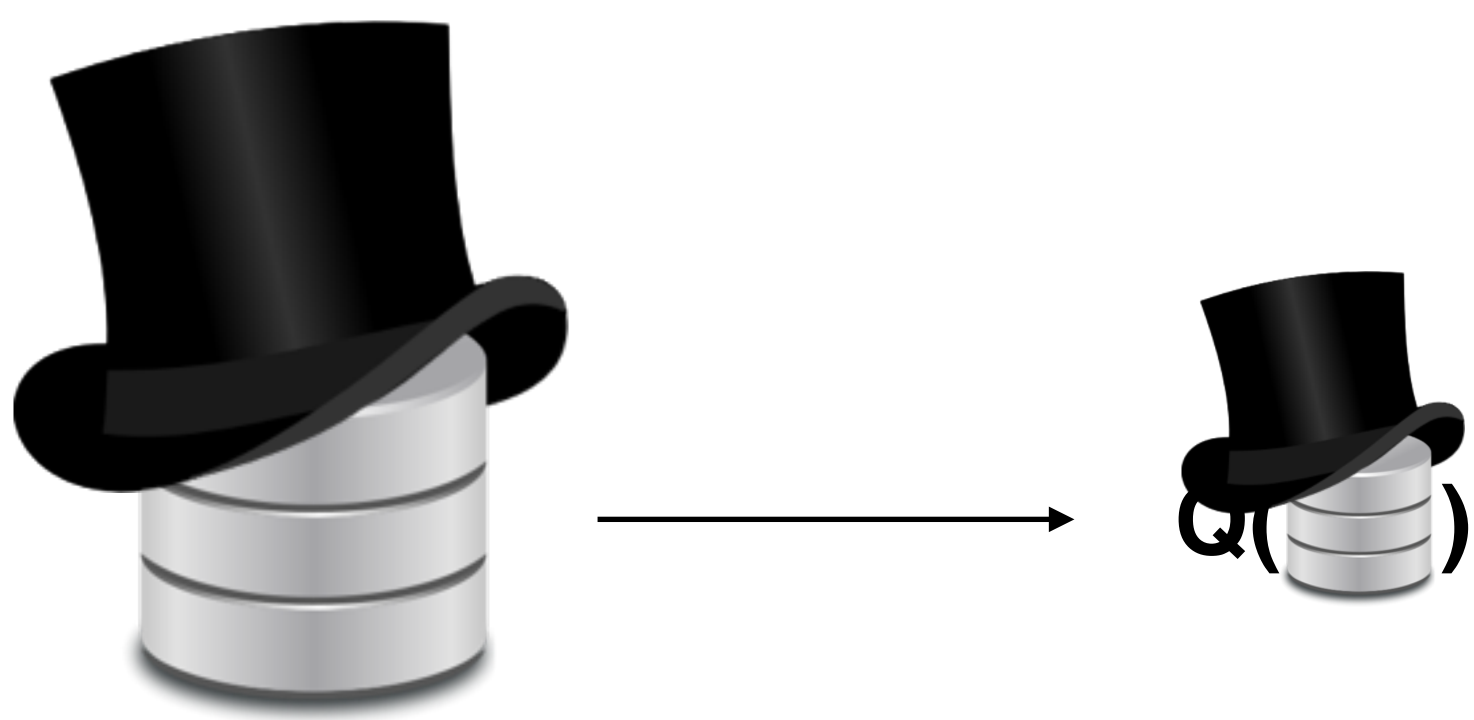
Materialized Views



Materialized Views



Materialized Views



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

View Maintenance

$$\texttt{VIEW} \leftarrow \texttt{Q(D)}$$

View Maintenance

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

View Maintenance

$$\texttt{WHEN } D \leftarrow D + \Delta D \texttt{ DO:}$$

$$\quad \texttt{VIEW} \leftarrow Q(D + \Delta D)$$

Re-evaluating the query from scratch is expensive!

View Maintenance

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

View Maintenance

(ideally) Smaller & Faster Query

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

View Maintenance

(ideally) Smaller & Faster Query

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

(ideally) Fast “merge” operation.

Intuition

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

$Q(D) = \mathrm{SUM}(D)$

Intuition

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

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

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

Intuition

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

$Q(D) = SUM(D)$

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

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

Intuition

R = {1, 2, 3}, S ={5,6}   ΔR = {4}

Q(R,S) = COUNT(R **x** S)

Intuition

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

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

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

Intuition

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

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

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

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

Intuition

+ ~ U

*** ~ x**

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

$$S_i + S_j = S_k$$

$$S_i \times S_j = S_k$$

$$S_i + S_0 = S_i$$

$$S_i \times S_1 = S_i$$

$$S_i \times S_0 = S_0$$

Additive & Multiplicative "zeroes"

$$S_i \times (S_j + S_k) = (S_i \times S_j) + (S_j \times S_k)$$

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

Incremental View Maintenance

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

Incremental View Maintenance

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

<u>Basic Challenges of IVM</u>

What does ΔR represent?

Incremental View Maintenance

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

<u>Basic Challenges of IVM</u>

What does ΔR represent?

How to interpret R $\pm$ ΔR?

Incremental View Maintenance

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

<u>Basic Challenges of IVM</u>

What does ΔR represent?

How to interpret R ± ΔR?

How to compute ΔQ?

What is ΔR?

What is ΔR?

What does it need to represent?

What is ΔR?

What does it need to represent?

Insertions

Deletions

Updates

What is ΔR?

What does it need to represent?

$\boxed{\begin{array}{c}\text{Insertions}\\ \\ \text{Deletions}\end{array}}$

Updates

(Delete Old Record & Insert Updated Record)

What is ΔR?

What is ΔR?

A Set/Bag of Insertions

A Set/Bag of Deletions

What is +?

R

A Set/Bag

ΔR

A Set/Bag of Insertions

A Set/Bag of Deletions

What is +?

R **+** **ΔR**

A Set/Bag **+** A Set/Bag of Insertions

A Set/Bag of Deletions

What is +?

R	**+**	**ΔR**
A Set/Bag	**+**	A Set/Bag of Insertions A Set/Bag of Deletions
$\mathtt{R}$	$\cup$ $-$	$\Delta\mathtt{R}_{\mathtt{inserted}}$ $\Delta\mathtt{R}_{\mathtt{deleted}}$

What is +?

$$\mathbf{R} \quad + \quad \Delta\mathbf{R}$$

A Set/Bag **+** A Set/Bag of Insertions

A Set/Bag of Deletions

$$\mathtt{R} \cup \Delta\mathtt{R}_{\mathtt{inserted}}$$

$$- \Delta\mathtt{R}_{\mathtt{deleted}}$$

But this breaks closure of '+'!

Incremental View Maintenance

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

Incremental View Maintenance

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

Incremental View Maintenance

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

$$\texttt{Given Q(R,S,...)}$$

$$\texttt{Construct } \Delta\texttt{Q}(\texttt{R},\Delta\texttt{R},\texttt{S},\Delta\texttt{S},\ldots)$$

Delta Queries

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

σ

$|$

R

Delta Queries

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

$$\begin{array}{c}\sigma \\ | \\ R\end{array}$$

R ΔR

Original R

Inserted
Tuples of R

Delta Queries

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

σ

R

σ

R ΔR

Original R

Inserted Tuples of R

Delta Queries

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

σ   σ

R   R   ΔR

Original R

Inserted
Tuples of R

Delta Queries

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

σ

|

R

R ΔR

σ

|

ΔR

Original R

Inserted Tuples of R

Does this work for deleted tuples?

Delta Queries

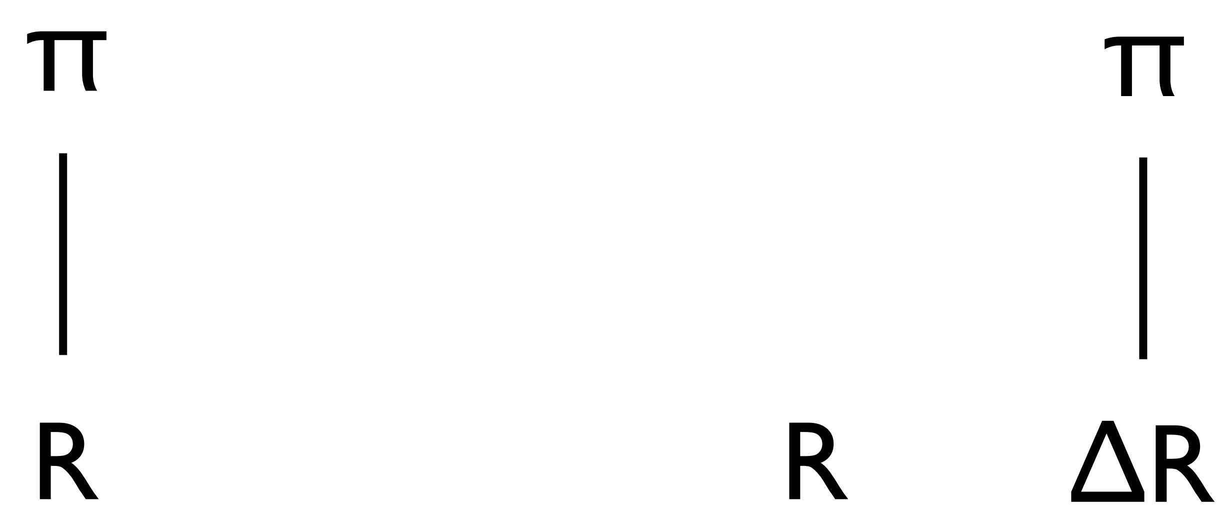
$$\Delta(\pi(R)) = \pi(\Delta R)$$

π π

R R ΔR

Delta Queries

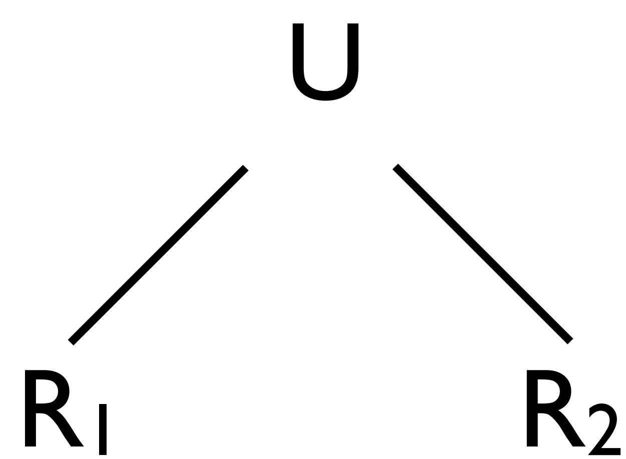
$$\Delta(\pi(R)) = \pi(\Delta R)$$



Does this work (completely) under set semantics?

Delta Queries

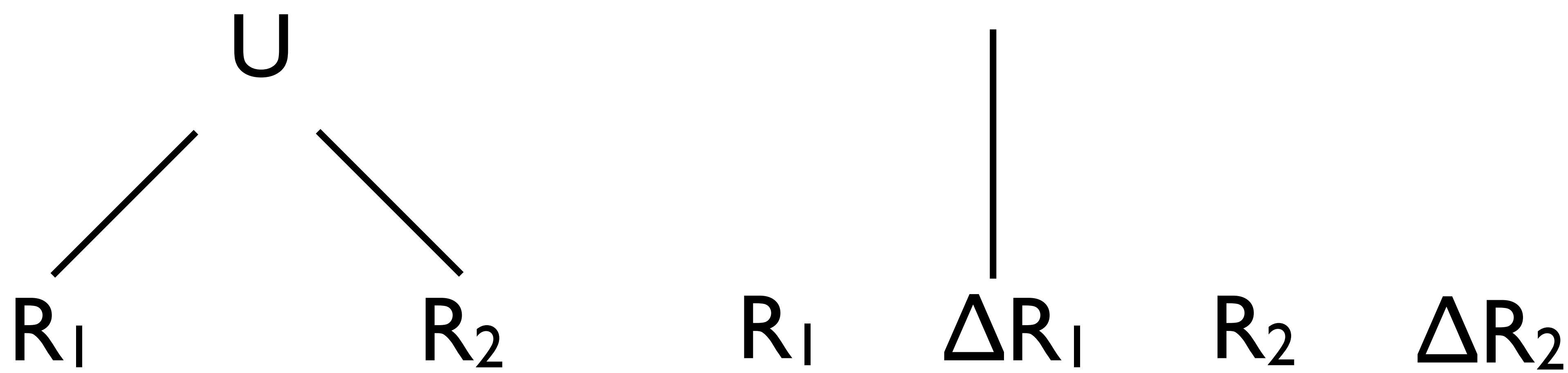
$$\Delta(R_1 \cup R_2)$$



R_1 ΔR_1 R_2 ΔR_2

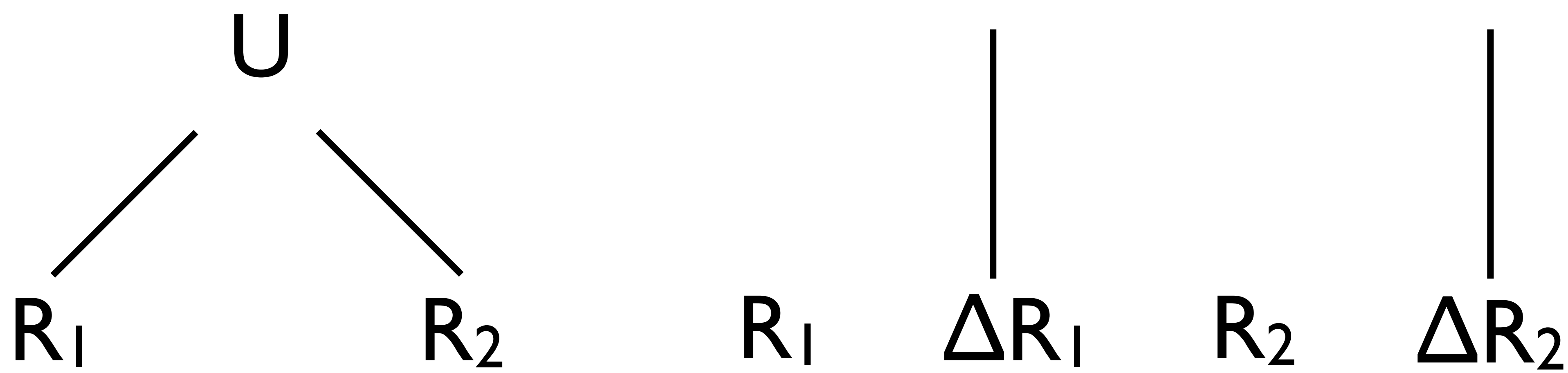
Delta Queries

$$\Delta(R_1 \cup R_2) = \Delta R_1 \cup \Delta R_2$$

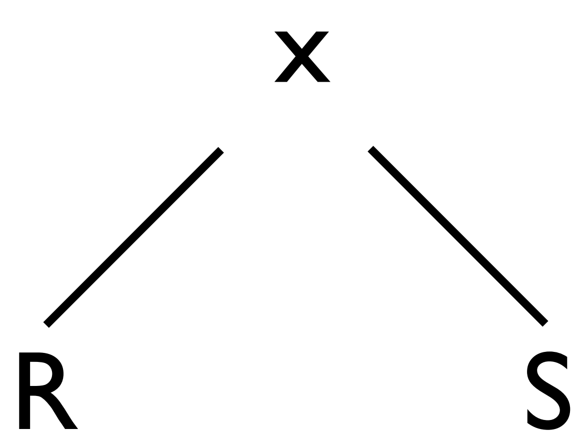


Delta Queries

$$\Delta(R_1 \cup R_2) = \Delta R_1 \cup \Delta R_2$$



Delta Queries



R ΔR S

Delta Queries

R : { 1, 2, 3 }   S : { 5, 6}

Delta Queries

R : { 1, 2, 3 }   S : { 5, 6}

R x S = { <1,5>, <1, 6>, <2,5>, <2,6>, <3,5>, <3,6> }

Delta Queries

$R : \{ 1, 2, 3 \} \qquad S : \{ 5, 6 \}$

$R \times S = \{ \langle 1,5 \rangle, \langle 1, 6 \rangle, \langle 2,5 \rangle, \langle 2,6 \rangle, \langle 3,5 \rangle, \langle 3,6 \rangle \}$

$\Delta R_{inserted} = \{ 4 \}$

$\Delta R_{deleted} = \{ 3,2 \}$

Delta Queries

$R : \{ 1, 2, 3 \} \qquad S : \{ 5, 6 \}$

$R \times S = \{ \langle 1,5 \rangle, \langle 1, 6 \rangle, \langle 2,5 \rangle, \langle 2,6 \rangle, \langle 3,5 \rangle, \langle 3,6 \rangle \}$

$\Delta R_{inserted} = \{ 4 \}$

$\Delta R_{deleted} = \{ 3,2 \}$

$(R+\Delta R) \times S = \{ \langle 1,5 \rangle, \langle 1, 6 \rangle, \boldsymbol{\langle 4,5 \rangle, \langle 4,6 \rangle} \}$

Delta Queries

$$R : \{\ 1,\ 2,\ 3\ \} \qquad S : \{\ 5,\ 6\}$$

$$R \times S = \{\ \langle 1,5\rangle,\ \langle 1,\ 6\rangle,\ \langle 2,5\rangle,\ \langle 2,6\rangle,\ \langle 3,5\rangle,\ \langle 3,6\rangle\ \}$$

$$\Delta R_{inserted} = \{\ 4\ \}$$

$$\Delta R_{deleted} = \{\ 3,2\ \}$$

$$(R+\Delta R) \times S = \{\ \langle 1,5\rangle,\ \langle 1,\ 6\rangle,\ \mathbf{\langle 4,5\rangle,\ \langle 4,6\rangle}\ \}$$

$$\Delta_{inserted}(R \times S) = \Delta R_{inserted} \times S$$

$$\Delta_{deleted}(R \times S) = \Delta R_{deleted} \times S$$

Delta Queries

$R : \{ 1, 2, 3 \} \qquad S : \{ 5, 6 \}$

$R \times S = \{ \langle 1,5 \rangle, \langle 1, 6 \rangle, \langle 2,5 \rangle, \langle 2,6 \rangle, \langle 3,5 \rangle, \langle 3,6 \rangle \}$

$\Delta R_{inserted} = \{ 4 \}$

$\Delta R_{deleted} = \{ 3,2 \}$

$(R+\Delta R) \times S = \{ \langle 1,5 \rangle, \langle 1, 6 \rangle, \boldsymbol{\langle 4,5 \rangle, \langle 4,6 \rangle} \}$

$\Delta_{inserted}(R \times S) = \Delta R_{inserted} \times S$

$\Delta_{deleted}(R \times S) = \Delta R_{deleted} \times S$

What if R and S <u>both</u> 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

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

$$\boxed{(R_1 \times R_2)} \cup (R_1 \times \Delta R_2) \cup (\Delta R_1 \times R_2) \cup (\Delta R_1 \times \Delta R_2)$$

The original query

Delta Queries

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

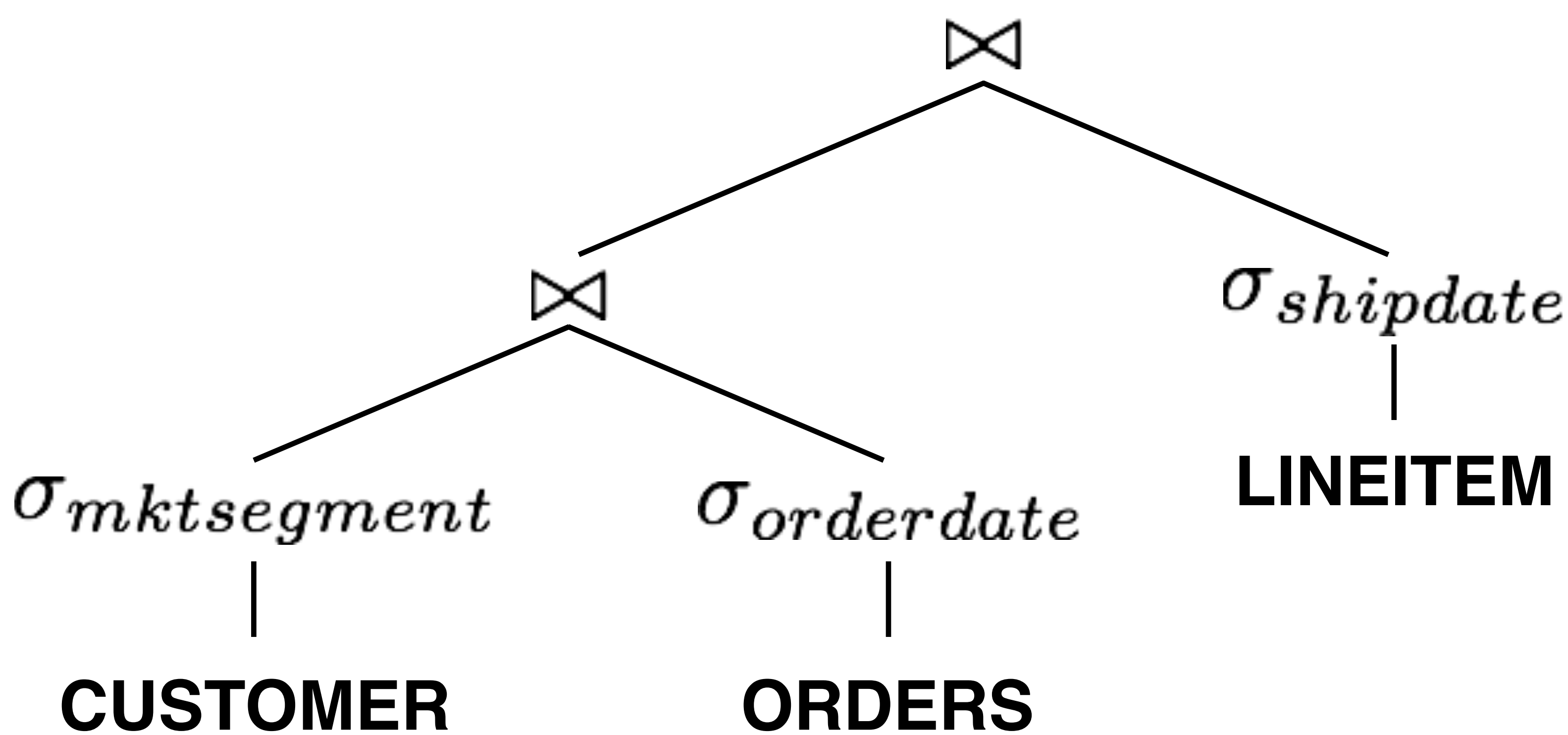
$$\boxed{(R_1 \times R_2)} \cup \boxed{(R_1 \times \Delta R_2) \cup (\Delta R_1 \times R_2) \cup (\Delta R_1 \times \Delta R_2)}$$

The original query

The delta query

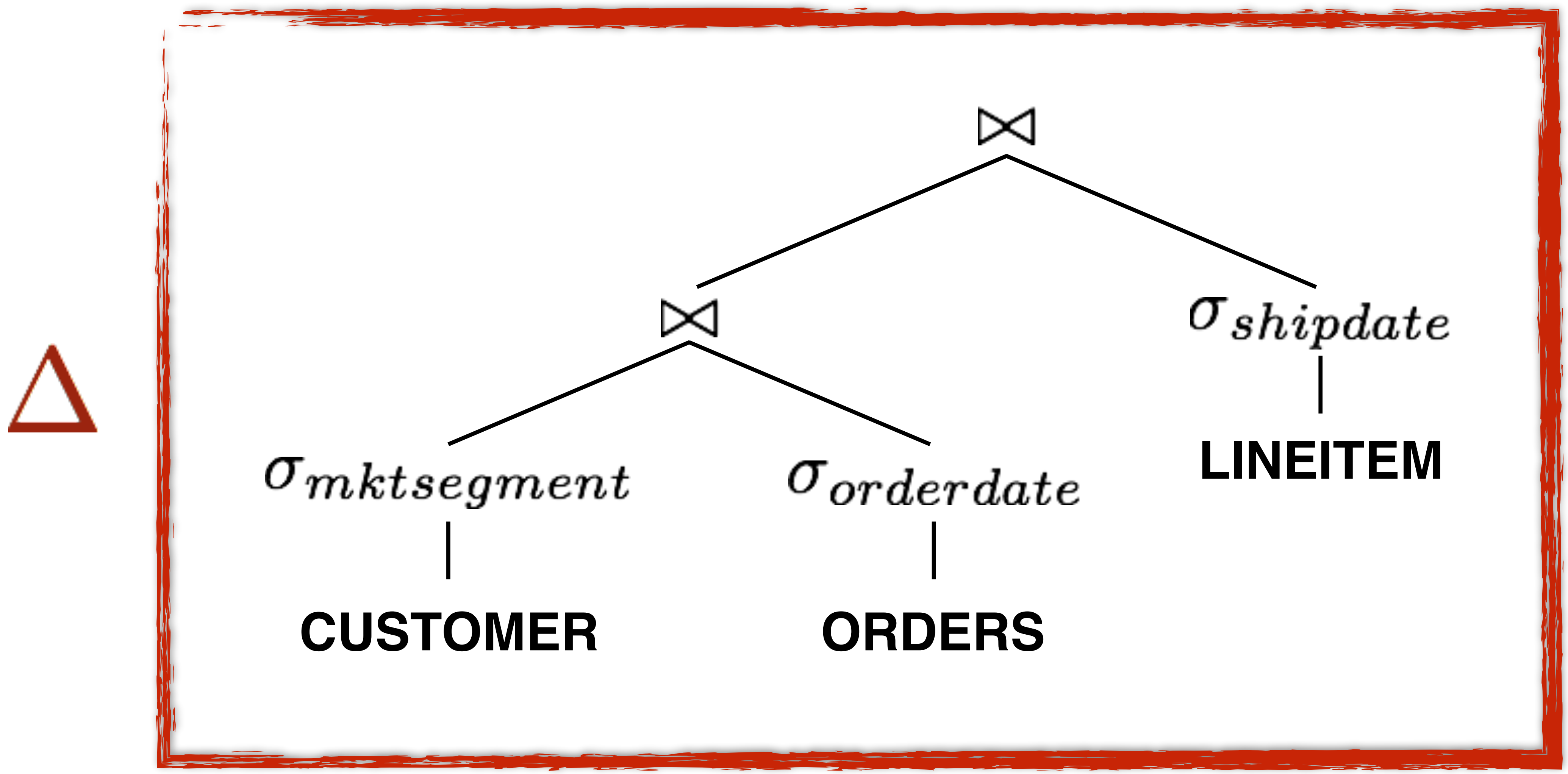
How about an example…

Delta Queries



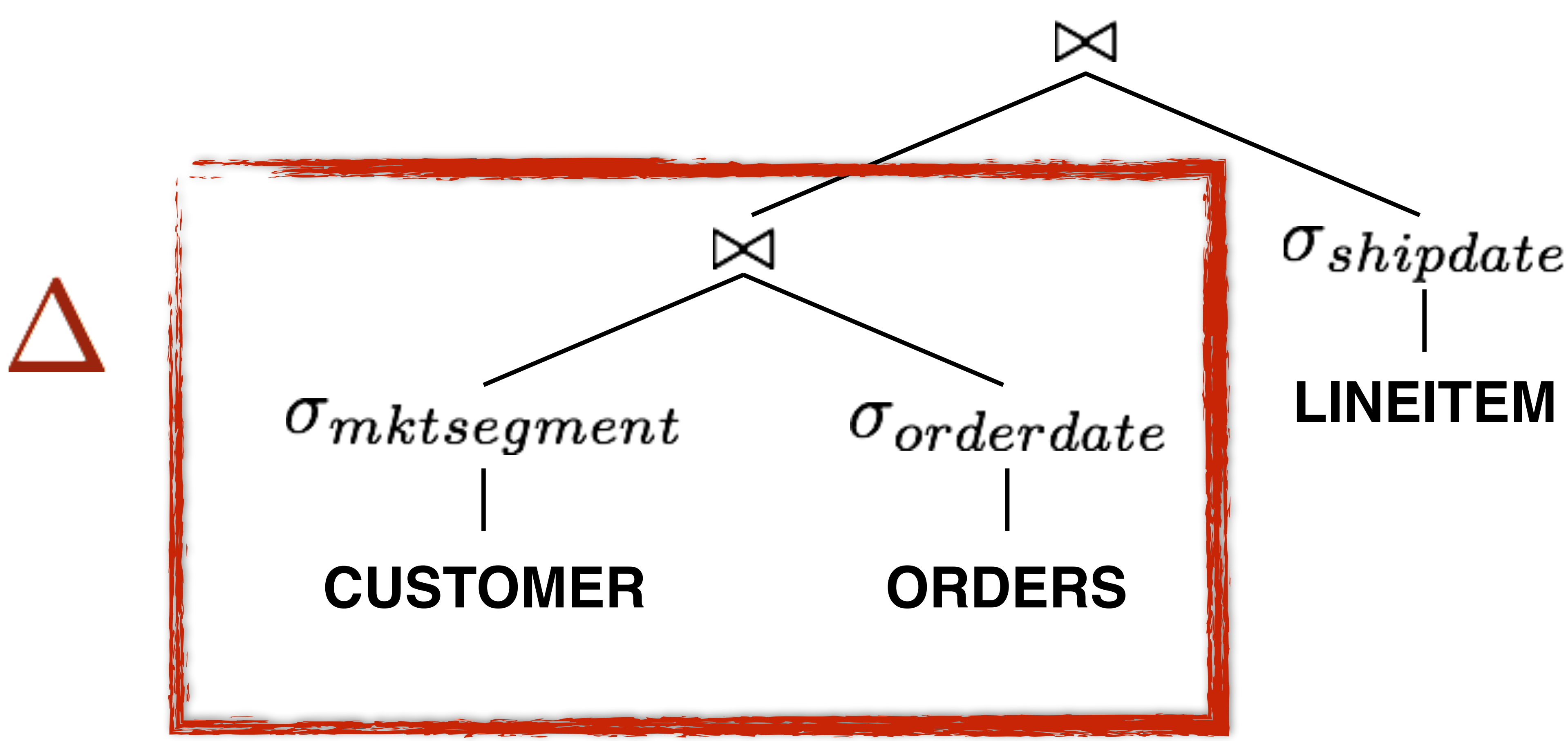
Let’s say you have an insertion into LINEITEM

Delta Queries



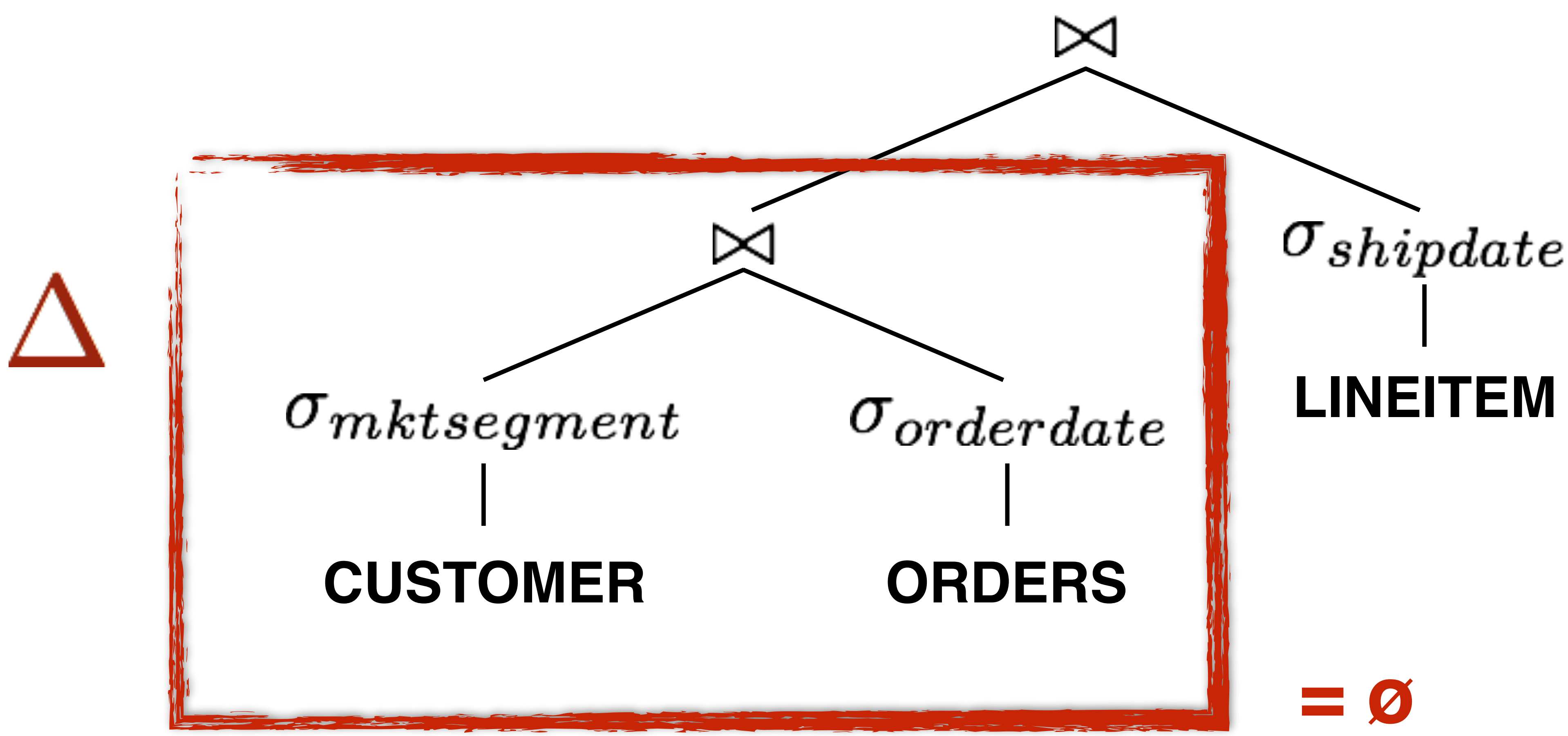
$$\Delta((\sigma(C) \bowtie \sigma(O)) \bowtie (\sigma(L))$$

Delta Queries



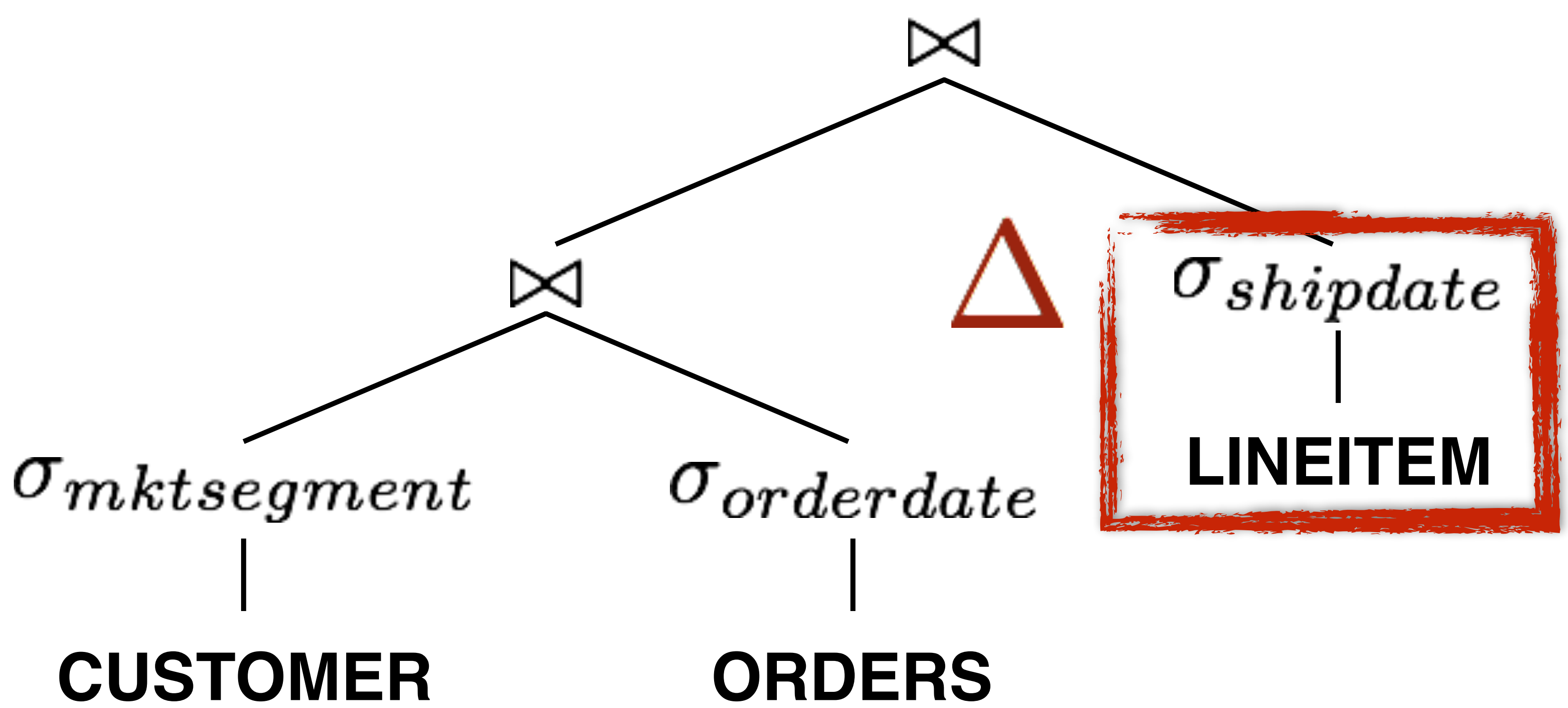
$$\Delta((\sigma(C) \bowtie \sigma(O)) \bowtie (\sigma(L))$$

Delta Queries



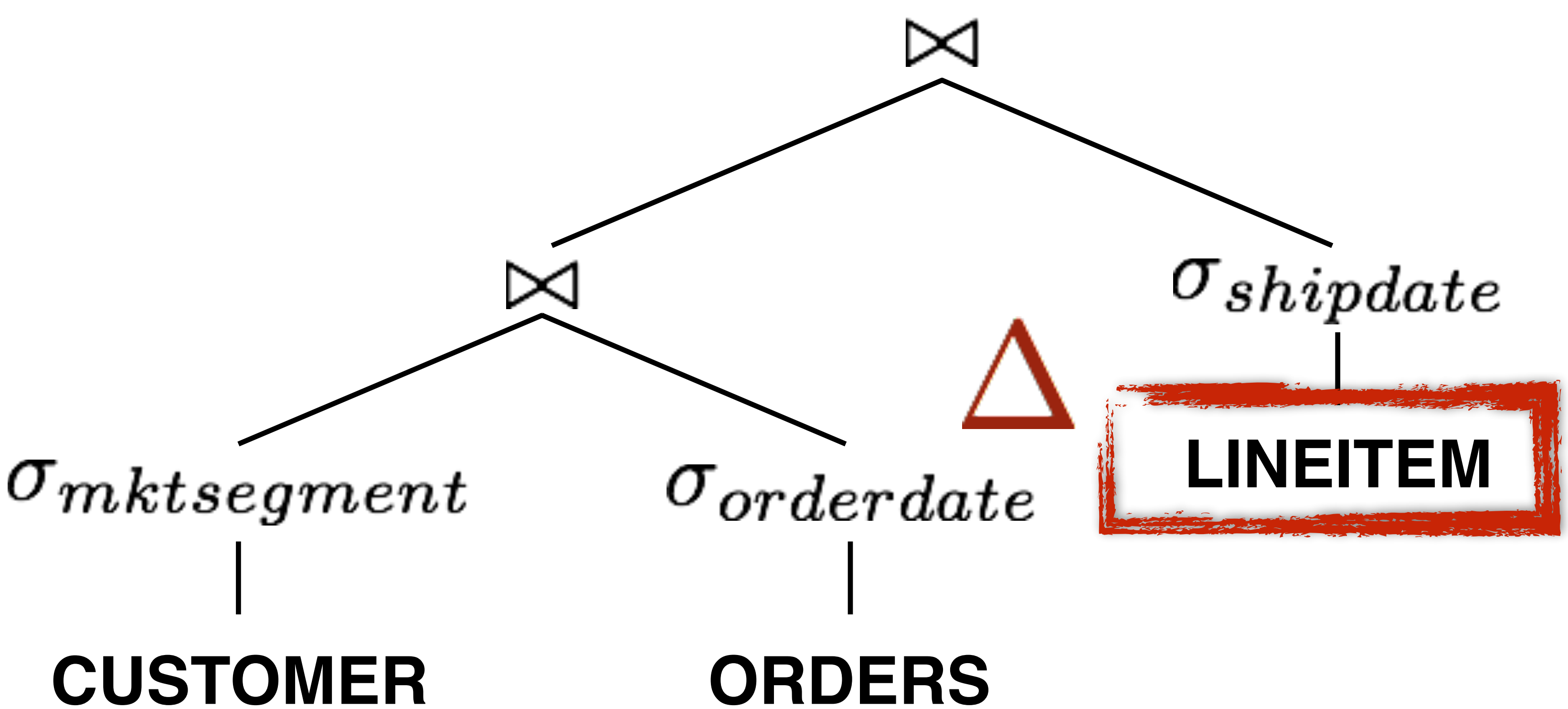
$$\Delta((\sigma(C) \bowtie \sigma(O)) \bowtie (\sigma(L))$$

Delta Queries



$$((\sigma(C) \bowtie \sigma(O)) \bowtie \Delta(\sigma(L))$$

Delta Queries



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, 2, 3, 3, 4, 4, 4, 4, 4, 4, 5 }

(not compact)

{ 1 → x3, 2 → x5, 3 → x2, 4 → x6, 5 → x1 }

Multiset representation: Tuple → # of occurrences

Multisets

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

{ 1 → x3, 2 → x5, 3 → x2, 4 → x6, 5 → x1 }
Multiset representation: Tuple → ~~# of occurrences~~
multiplicity

Multiset Deltas

Insertions = Positive Multiplicity

Deletions = Negative Multiplicity

+ = Bag/Multiset Union