Advanced Data Management Technologies

Unit 12 — Generalized Multidimensional Join

J. Gamper

Free University of Bozen-Bolzano
Faculty of Computer Science
IDSE

Acknowledgements: I am indebted to M. Böhlen for providing me the lecture notes.

Outline

Outline

Goals

- We need an algebraic operator for **Ad Hoc OLAP** queries (i.e., no data cube is pre-computed and aggregates are computed on the detail data):
  - The operator must allow us to easily express OLAP queries.
    - Grouping should be independent from detail data.
    - More general specification of groups over which aggregates are computed.
  - The operator must admit efficient evaluation algorithms.
  - The operator must interact seamlessly with the other operators of the relational algebra (query optimization).

Motivating Application

- Analysis of network data
  - Collect, correlate, and analyze huge amounts of data across the network.
- Example queries:
  - Network usage: For each IP address, what fraction of the total traffic is due to web flows?
  - Principal components: On an hourly basis, what fraction of the total traffic is from IP subnets whose hourly traffic is within 10% of the max?
  - Pattern identification: Break down all flows recorded on US election day by all combinations of source router, destination router, and protocol.

Data Warehouse Schema

- IP Flows star schema
  - `UserDim(UserId, Name, Addr, Balance, ...)`
  - `IPRegDim(IPKey, UserId, IP, Port, Mask, ASystem, ...)`
  - `RouterDim(RouterId, Addr, Type, ...)`
  - `HourDim(HourKey, Start, End)`
  - `FlowFact(RouterId, SourceKey, DestKey, Protocol, StartTime, EndTime, NumPackets, NumBytes, ...)`
- Denormalized IP Flow schema
  - `Flow(RouterId, SourceIP, SourcePort, SourceAS, DestIP, DestPort, DestAS, Protocol, StartTime, EndTime, NumPackets, NumBytes, ...)`

Network Usage (Relational)/1

- Network usage: For each IP address, what fraction of the total traffic is due to web flows?
- Step 1: Determine total traffic.

IP

Key	Addr
1	1.2.0
2	2.5.0

Flow

Key	Nbts	Type
1	25	web
1	5	ftp
1	10	web
2	15	web

Join

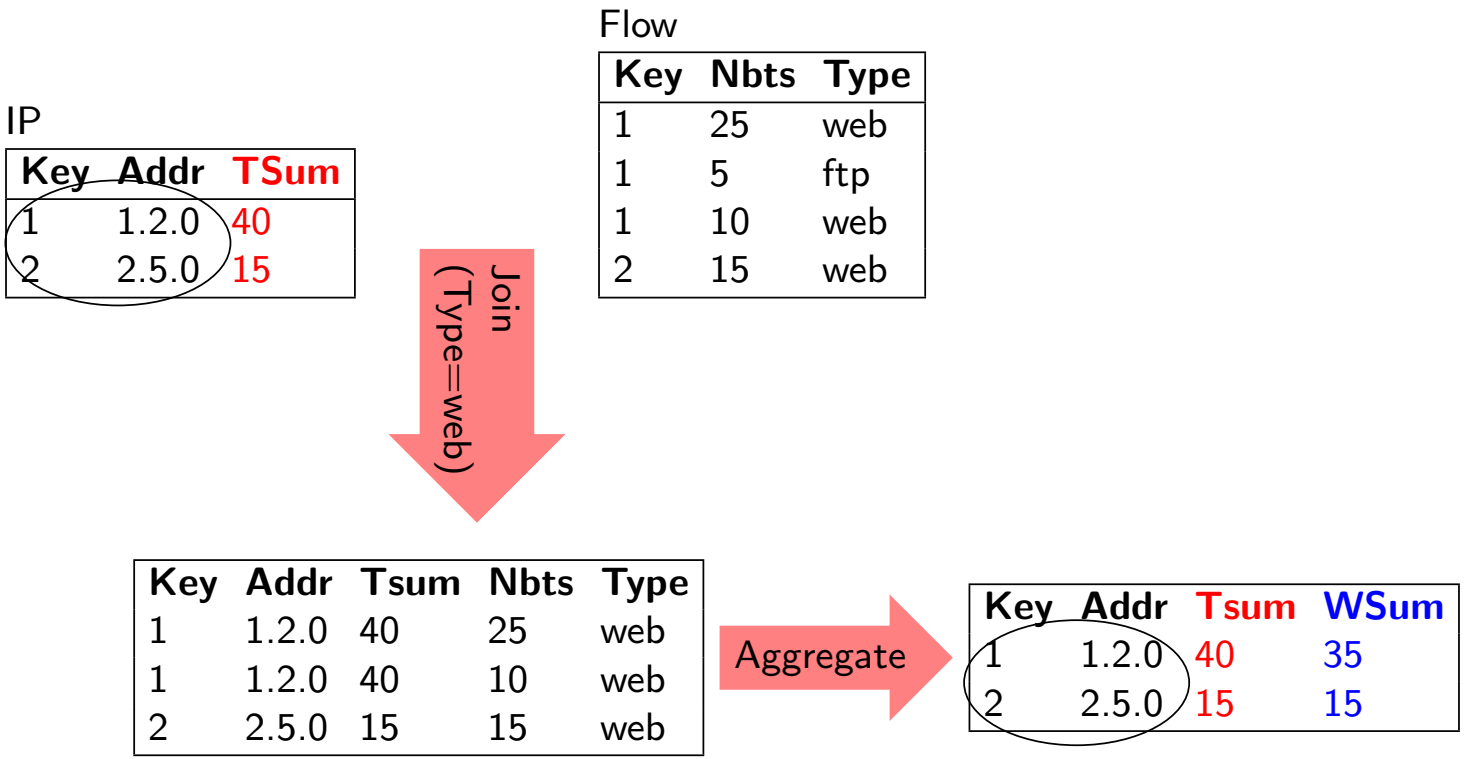
Key	Addr	Nbts	Type
1	1.2.0	25	web
1	1.2.0	5	ftp
1	1.2.0	10	web
2	2.5.0	15	web

Aggregate

Key	Addr	TSum
1	1.2.0	40
2	2.5.0	15

Network Usage (Relational)/2

- Step 2: Determine web traffic.



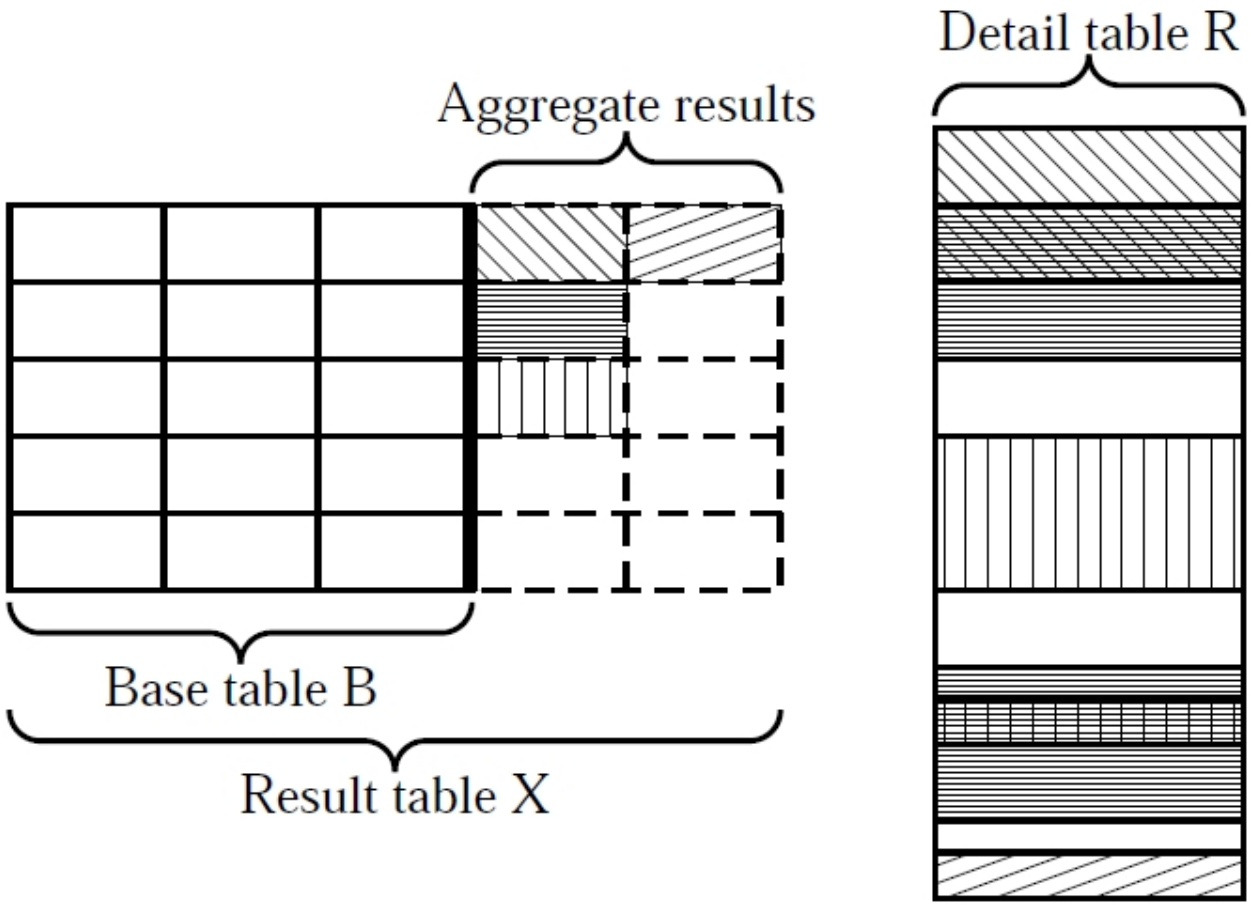
- Combination of joins and aggregations with different groupings are required.
- OLAP extensions provide some support, but some queries are still difficult to express and might be expensive.

Outline

The Generalized MD-join/1

- MD($\mathbf{b}, \mathbf{r}, (l_1, \ldots, l_m), (\theta_1, \ldots, \theta_m)$)
  - **b** is the base table (the "groups");
  - **r** is the detail table (fact data);
  - l_1 to l_m are lists of aggregate functions;
  - θ_1 to θ_m are possibly complex conditions over **b** and **r** describing what data in **r** is to be aggregated for each l_i, respectively.
- Result: base table **b** extended with the aggregates in l_1 to l_m.
- Salient feature: decouples grouping from aggregation
  - SQL does not do this!

The Generalized MD-join/2



Network Usage (GMD-join)/1

- Network usage: For each IP address, what fraction of the total number of flows is due to web traffic?
- Step 1: total traffic
  - $\mathbf{x}_1 = \text{MD}(\text{IP}, \text{Flow}, (\text{SUM}(\textit{NBts})/\textit{TSum}), (\text{IP.Key} = \text{Flow.Key}))$
  - Schema of $\mathbf{x}_1$: (Key, Addr, TSum)
- Step 2: web traffic
  - $\mathbf{x}_2 = \text{MD}(\mathbf{x}_1, \text{Flow}, (\text{SUM}(\textit{NBts})/\textit{WSum}), (\mathbf{x}_1.\text{Key} = \text{Flow.Key} \wedge \text{Flow.Source} = \text{'web'}))$
  - Schema of $\mathbf{x}_2$: (Key, Addr, TSum, WSum)
- Sequences of GMDJs instead of multiple aggregate-join expressions.

Network Usage (GMD-join)/2

- GMD-join can do it in one step
  - $\mathrm{MD}(\mathrm{IP}, \mathrm{Flow}, ((\mathrm{SUM}(\mathit{NBts})/\mathit{TSum}), (\mathrm{SUM}(\mathit{NBts})/\mathit{WSum})), (\theta_1, \theta_2))$
  - θ_1: $(\mathrm{IP.Key} = \mathrm{Flow.Key})$
  - θ_2: $(\mathrm{IP.Key} = \mathrm{Flow.Key} \wedge \mathrm{Flow.Type} =' \mathit{web}')$

- Equivalent RA expression (left outer join, LOJ):

$$\pi[\mathrm{Key}, \mathrm{IP}, \mathit{S1}, \mathrm{SUM}(\mathit{NB})](\\ (\pi[\mathrm{Key}, \mathrm{IP}, \mathrm{SUM}(\mathit{NB})/\mathit{S1}](\mathrm{IP\ LOJ}[\theta_1]\ \mathrm{Flow}))\\ \mathrm{LOJ}[\theta_2]\ \mathrm{Flow})$$

  - θ_1: $(\mathrm{IP.Key} = \mathit{Flow}.\mathrm{Key})$
  - θ_2: $(\mathrm{IP.Key} = \mathrm{Flow.Key} \wedge \mathrm{Flow.Type} =' \mathit{web}')$

Execution of GMD-join/1

- MD(IP, Flow, (SUM(*NBts*)/*TSum*, SUM(*NBts*)/*WSum*), (θ_1, θ_2))
  - θ_1: (IP.Key = Flow.Key)
  - θ_2: (IP.Key = Flow.Key $\wedge$ Flow.Type = *web*)
- Step 1: Init base-result structure

IP

Key	Addr
1	1.2.0
2	2.5.0

$\Longrightarrow$ *Init*

Result table **x**

Key	Addr	TSum	WSum
1	1.2.0	0	0
2	2.5.0	0	0

- Step 2: Scan detail tuples and incrementally update result structure.

Result table **x**

Key	Addr	TSum	WSum
1	1.2.0	25	25
2	2.5.0	0	0

$\Longleftarrow$ *Update*

Flow

Key	Nbts	Type
1	25	web
1	5	ftp
1	10	web
2	15	web

Execution of GMD-join/2

Result table **x**

Key	Addr	TSum	WSum
1	1.2.0	30	25
2	2.5.0	0	0

Update ⟸

Flow

Key	Nbts	Type
~~1~~	~~25~~	~~web~~
1	5	ftp
1	10	web
2	15	web

Result table **x**

Key	Addr	TSum	WSum
1	1.2.0	40	35
2	2.5.0	0	0

Update ⟸

Flow

Key	Nbts	Type
~~1~~	~~25~~	~~web~~
~~1~~	~~5~~	~~ftp~~
1	10	web
2	15	web

Result table **x**

Key	Addr	TSum	WSum
1	1.2.0	40	35
2	2.5.0	15	15

Update ⟸

Flow

Key	Nbts	Type
~~1~~	~~25~~	~~web~~
~~1~~	~~5~~	~~ftp~~
~~1~~	~~10~~	~~web~~
2	15	web

Advantages of the GMD-join

- Flexible Operator that allows a "simple" formulation of very complex queries
  - Evaluation of multiple complex aggregations using just a single operator and a single pass through the detail data.
- Very efficient to evaluate
  - Indexing on the base-result structure constructed at query execution time performs very effectively.
- Can be optimized effectively
  - GMD-joins allow efficient optimization schemes to be developed for complex aggregate queries.

Complex Aggregation Example/1

- Assume the following denormalized IP Flow schema:
  ```
  Flow(RouterId, SourceIP, SourcePort, SourceAS, DestIP,
       DestPort, DestAS, Protocol, StartTime, EndTime,
       NumPackets,NumBytes, ...)
  ```

- **Query:** Determine the cumulative total, two hour moving average, and hourly total of network flow broken down by hour of the day.

Complex Aggregation Example/2

- GMD-Join:
  MD(Hour/**b**, Flow/**r**, $(l_1, l_2, l_3), (\theta_1, \theta_2, \theta_3)$)
  - $l_1 : (SUM(NB)/CSum)$
  - $\theta_1 : (\mathbf{r}.T \leq \mathbf{b}.HEnd)$
  - $l_2 : (SUM(NB)/MSum, COUNT(*)/Mcnt)$
  - $\theta_2 : (\mathbf{r}.T \geq \mathbf{b}.HStart - 60 \wedge \mathbf{r}.T \leq \mathbf{b}.HEnd)$
  - $l_3 : (SUM(NB)/Hsum)$
  - $\theta_3 : (\mathbf{r}.T \geq \mathbf{b}.HStart \wedge \mathbf{r}.T \leq \mathbf{b}.HEnd)$

- Result table

HId	HStart	HEnd	CSum	MAvg	HSum
1	0	59	11	11/2	11
2	60	119	17	17/3	6
3	120	179	33	22/3	16

- Can be expressed in SQL with window functions!

- Base and detail table

Hour

HId	HStart	HEnd
1	0	59
2	60	119
3	120	179

Flow

SIP	DIP	NB	T
5	29	3	30
5	7	8	45
5	29	6	110
7	29	6	161
6	29	10	170

Outline

2D Cumulative Aggregates Example/1

- Consider the following *Lineitem* relation

Lineitem

	Ordkey	*Partkey*	*Suppkey*	*Quantity*	*Price*	*Discount*	*Shipdate*
r_1	O1	P1	S1	2	220	0.00	2008.01.23
r_2	O2	P1	S1	4	440	0.05	2008.01.23
r_3	O3	P2	S1	6	300	0.10	2008.01.23
r_4	O4	P2	S2	7	420	0.10	2008.01.23
r_5	O5	P2	S1	2	100	0.00	2008.01.24
r_6	O6	P1	S2	3	240	0.05	2008.01.24
r_7	O7	P2	S1	9	450	0.05	2008.01.24
r_8	O8	P1	S2	8	640	0.10	2008.01.24

- **Query Q1:** Compute the number of orders per day and disc rate (*CntDD*), the cumulative number of orders per day (*CumCntD*), and the cumulative number of orders per day and disc rate (CumCntDD).

2D Cumulative Aggregates Example/2

Lineitem

	Ordkey	*Partkey*	*Suppkey*	*Quantity*	*Price*	*Discount*	*Shipdate*
r_1	O1	P1	S1	2	220	0.00	2008.01.23
r_2	O2	P1	S1	4	440	0.05	2008.01.23
r_3	O3	P2	S1	6	300	0.10	2008.01.23
r_4	O4	P2	S2	7	420	0.10	2008.01.23
r_5	O5	P2	S1	2	100	0.00	2008.01.24
r_6	O6	P1	S2	3	240	0.05	2008.01.24
r_7	O7	P2	S1	9	450	0.05	2008.01.24
r_8	O8	P1	S2	8	640	0.10	2008.01.24

- Result **x** of Q1

x

	Shipdate	*Discount*	*CntDD*	*CumCntD*	*CumCntDD*
x_1	2008.01.23	0.00	1	4	1
x_2	2008.01.23	0.05	1	4	2
x_3	2008.01.23	0.10	2	4	4
x_4	2008.01.24	0.00	1	8	2
x_5	2008.01.24	0.05	2	8	5
x_6	2008.01.24	0.10	1	8	8

2D Cumulative Aggregates Example/3

- GMD-join formulation

$$\mathrm{MD}(\pi[\mathit{Shipdate}, \mathit{Discount}]\mathit{Lineitem}/\mathbf{b}, \mathit{Lineitem}/\mathbf{r}, (l_1, l_2, l_3), (\theta_1, \theta_2, \theta_3))$$

$$l_1 \equiv (\mathrm{COUNT}(\mathit{Quantity})/\mathit{CntDD})$$
$$\theta_1 \equiv (\mathbf{r}.\mathit{Shipdate} = \mathbf{b}.\mathit{Shipdate} \wedge \mathbf{r}.\mathit{Discount} = \mathbf{b}.\mathit{Discount})$$
$$l_2 \equiv (\mathrm{COUNT}(\mathit{Quantity})/\mathit{CumCntD})$$
$$\theta_2 \equiv (\mathbf{r}.\mathit{Shipdate} \leq \mathbf{b}.\mathit{Shipdate})$$
$$l_3 \equiv (\mathrm{COUNT}(\mathit{Quantity})/\mathit{CumCntDD})$$
$$\theta_3 \equiv (\mathbf{r}.\mathit{Shipdate} \leq \mathbf{b}.\mathit{Shipdate} \wedge \mathbf{r}.\mathit{Discount} \leq \mathbf{b}.\mathit{Discount})$$

2D Aggregates (*CntDD*)

- Groups are defined by identical values on the grouping attributes.
- Can be expressed in SQL
  - `GROUP BY` *Shipdate*, *Discount*

Lineitem/**r**

	Ordkey	*Partkey*	*Suppkey*	*Quantity*	*Price*	*Discount*	*Shipdate*
r_1	O1	P1	S1	2	220	0.00	2008.01.23
r_2	O2	P1	S1	4	440	0.05	2008.01.23
r_3	O3	P2	S1	6	300	0.10	2008.01.23
r_4	O4	P2	S2	7	420	0.10	2008.01.23
r_5	O5	P2	S1	2	100	0.00	2008.01.24
r_6	O6	P1	S2	3	240	0.05	2008.01.24
r_7	O7	P2	S1	9	450	0.05	2008.01.24
r_8	O8	P1	S2	8	640	0.10	2008.01.24

$\mathbf{r}.\mathit{Shipdate} = \mathbf{x}.\mathit{Shipdate} \wedge$
$\mathbf{r}.\mathit{Discount} = \mathbf{x}.\mathit{Discount}$

x

	Shipdate	*Discount*	*CntDD*	*CumCntD*	*CumCntDD*
x_1	2008.01.23	0.00	1	4	1
x_2	2008.01.23	0.05	1	4	2
x_3	2008.01.23	0.10	2	4	4
x_4	2008.01.24	0.00	1	8	2
x_5	2008.01.24	0.05	2	8	5
x_6	2008.01.24	0.10	1	8	8

1D Cumulative Aggregates (*CumCntD*)

- Groups are contiguous rows after sorting the data on the grouping attribute.
- Can be expressed by SQL window functions
  - `SUM(COUNT(*)) OVER (ORDER BY Shipdate ROWS UNBOUNDED PRECEDING)`

.

Lineitem/**r**

	Ordkey	*Partkey*	*Suppkey*	*Quantity*	*Price*	*Discount*	*Shipdate*
r_1	O1	P1	S1	2	220	0.00	2008.01.23
r_2	O2	P1	S1	4	440	0.05	2008.01.23
r_3	O3	P2	S1	6	300	0.10	2008.01.23
r_4	O4	P2	S2	7	420	0.10	2008.01.23
r_5	O5	P2	S1	2	100	0.00	2008.01.24
r_6	O6	P1	S2	3	240	0.05	2008.01.24
r_7	O7	P2	S1	9	450	0.05	2008.01.24
r_8	O8	P1	S2	8	640	0.10	2008.01.24

$\mathbf{r}.Shipdate \leq \mathbf{x}.Shipdate$

x

	Shipdate	*Discount*	*CntDD*	*CumCntD*	*CumCntDD*
x_1	2008.01.23	0.00	1	4	1
x_2	2008.01.23	0.05	1	4	2
x_3	2008.01.23	0.10	2	4	4
x_4	2008.01.24	0.00	1	8	2
x_5	2008.01.24	0.05	2	8	5
x_6	2008.01.24	0.10	1	8	8

2D Cumulative Aggregates (*CumCntDD*)

- Groups are specified along 2 dimensions
- Ordering with contiguous rows might **not** exist.

Lineitem/**r**

	Ordkey	*Partkey*	*Suppkey*	*Quantity*	*Price*	*Discount*	*Shipdate*
r_1	O1	P1	S1	2	220	0.00	2008.01.23
r_2	O2	P1	S1	4	440	0.05	2008.01.23
r_3	O3	P2	S1	6	300	0.10	2008.01.23
r_4	O4	P2	S2	7	420	0.10	2008.01.23
r_5	O5	P2	S1	2	100	0.00	2008.01.24
r_6	O6	P1	S2	3	240	0.05	2008.01.24
r_7	O7	P2	S1	9	450	0.05	2008.01.24
r_8	O8	P1	S2	8	640	0.10	2008.01.24

$\mathbf{r}.Shipdate \leq \mathbf{x}.Shipdate \wedge$
$\mathbf{r}.Discount \leq \mathbf{x}.Discount$

x

	Shipdate	*Discount*	*CntDD*	*CumCntD*	*CumCntDD*
x_1	2008.01.23	0.00	1	4	1
x_2	2008.01.23	0.05	1	4	2
x_3	2008.01.23	0.10	2	4	4
x_4	2008.01.24	0.00	1	8	2
x_5	2008.01.24	0.05	2	8	5
x_6	2008.01.24	0.10	1	8	8

- 2D cumulative (and higher dimensional) aggregates might be difficult and very expensive in SQL!

Outline

Reduction to SQL

- The GMD-join MD($\mathbf{b}, \mathbf{r}, (l_1, \ldots, l_m), (\theta_1, \ldots, \theta_m)$) can be systematically transformed into SQL as follows

```
SELECT   B1,...,Bh
         f11(CASE WHEN θ1 THEN r.A11 ELSE N11 END) AS C11
         ...
         fmk (CASE WHEN θm THEN r.Amk ELSE Nmk END) AS Cmk
FROM     b LEFT OUTER JOIN r ON θ'
GROUP BY B1,...,Bh
```

- `B1,...,Bh` are the attributes of **b**.
- `fij` are the aggregate functions in l_i.
- `Nij` are the neutral values of aggregate functions.
- θ' is the common subpart of all θ_i
  - Cartesian Product if θ' is empty

Reduction to SQL Example/1

```
SELECT   B.Shipdate, B.Disc,
         COUNT(CASE WHEN L.Shipdate = B.Shipdate AND L.Disc = B.Disc)
                    THEN Quantity
                    ELSE 0 END) AS CntDD,
         COUNT(CASE WHEN L.Shipdate <= B.Shipdate)
                    THEN Quantity
                    ELSE 0 END) AS CumCntD,
         COUNT(CASE WHEN L.Shipdate <= B.Shipdate AND L.Disc <= B.Disc)
                    THEN Quantity
                    ELSE 0 END) AS CumCntDD
FROM     (SELECT DISTINCT Shipdate, Disc FROM Lineitem) B, Lineitem L
GROUP BY B.Shipdate, B.Disc;
```

- Common condition $\theta' = \mathit{true}$ (empty), hence we have a Cartesian product.
- Efficient join techniques (e.g., hash or sort-merge joins) are not applicable.

Reduction to SQL Example/2

- Manual ad hoc optimizations are possible, e.g., compute all aggregates separately.

```
WITH
q1 AS ( SELECT   Shipdate, Disc, COUNT(Quantity) AS CntDD
        FROM     Lineitem
        GROUP BY Shipdate, Disc ),
q2 AS ( SELECT   Shipdate,
                 SUM(COUNT(Quantity)) OVER
                     (ORDER BY Shipdate ROWS UNBOUNDED PRECEDING) AS CumCntD
        FROM     Lineitem
        GROUP BY Shipdate, Disc ),
q3 AS ( SELECT   B.Shipdate, B.Disc, COUNT(Quantity) AS CumCntDD
        FROM     ( SELECT DISTINCT Shipdate, Disc FROM Lineitem ) B
                 LEFT OUTER JOIN
                 ( SELECT DISTINCT Shipdate, Disc FROM Lineitem ) L
                 ON L.Shipdate <= B.Shipdate AND L.Disc <= B.Disc
        GROUP BY Shipdate, Disc ),
SELECT * FROM q1 NATURAL JOIN q2 NATURAL JOIN q3;
```

- 3 scans of base and detail table are required.

Outline

Basic Algorithm for the GMD-join

Algorithm: GMDJBasic($\mathbf{b}, \mathbf{r}, (l_1, \ldots, l_m), (\theta_1, \ldots, \theta_m)$)

```
// Initialize result table
```
x $\leftarrow$ **b** extended with a column for each aggregate;
Initialize **x** to NULL for MAX and MIN, to 0 for SUM and COUNT;

```
// Scan detail table and update aggregates
```
foreach $r \in \mathbf{r}$ **do**
  foreach $x \in \mathbf{x}$ **do**
    foreach $\theta_i \in \{\theta_1, \ldots, \theta_m\}$ **do**
      if $\theta_i(x, r)$ **then**
        Update aggregates l_i in x;

return x;

Hash Index for the GMD-join

Algorithm: GMDJHash($\mathbf{b}, \mathbf{r}, (l_1, \ldots, l_m), (\theta_1, \ldots, \theta_m)$)

```
// Initialize result table
```
$\mathbf{x} \leftarrow \mathbf{b}$ extended with a column for each aggregate;
Initialize $\mathbf{x}$ to NULL for MAX and MIN, to 0 for SUM and COUNT;

```
// Create an index on the result table
```
Construct hash index for $\mathbf{x}$;

```
// Scan detail table and update aggregates
```
foreach $r \in \mathbf{r}$ **do**
  foreach $\theta_i \in \{\theta_1, \ldots, \theta_m\}$ **do**
    Use index to find $\mathbf{x}_i \leftarrow \{x \mid x \in \mathbf{x} \wedge \theta_i(x, r)\}$;
    foreach $x \in \mathbf{x}_i$ **do**
      Update aggregates l_i in x;

return $\mathbf{x}$;

Incremental Aggregation/1

- The GMDJ incrementally computes the aggregate values in the result table while scanning the detail table.
- Different types of aggregate functions (AF) need to be distinguished.
  - Distributive AF
  - Algebraic AF
  - Holistic AF

Incremental Aggregation/2

- **Distributive AF**
  - Result over a set A is the aggregation of partial results over subsets A_i
    - $A_1 \cup \ldots A_k = A$, $A_i \cap A_j = \emptyset$
  - $F(A) = G(F(A_1), \ldots, F(A_k))$
  - G is also called super-aggregate
  - Distributive aggregates: MIN, MAX, SUM, COUNT
    - e.g., $COUNT(A) = SUM(COUNT(A_1), \ldots, COUNT(A_k))$
- **Algebraic AF**
  - Computation can be reduced to a number of distributive sub-aggregates
  - $F(A) = G(F_1(A), \ldots, F_m(A))$
  - Algebraic aggregates: AVG
    - $AVG(A) = DIV(SUM(A), COUNT(A))$
- **Holistic AF**
  - Cannot be reduced to a fixed number of sub-aggregates.
  - Incremental computation in GMD-join is not possible.
  - $F(A) = F(A_1 \cup \cdots \cup A_k)$
  - Holistic aggregates: MEDIAN, MODE, RANK

Evaluation of Algebraic AF

Algorithm: GMDJAlgebraic($\mathbf{b}, \mathbf{r}, (l_1, \ldots, l_m), (\theta_1, \ldots, \theta_m)$)

`// Replace algebraic aggregates by distributive sub-aggregates`
Let $l'_i \leftarrow l_i$ for $1 \leq i \leq m$;
foreach *algebraic aggregate* $f_{i_j} \in \{l'_1, \ldots, l'_m\}$ **do**
  Replace f_{i_j} with its distributive sub-aggregates $f_{i_j}^1, \ldots, f_{i_j}^{p_{i_j}}$;

`// Compute aggregates`
$\mathbf{x} \leftarrow$ GMDJHash($\mathbf{b}, \mathbf{r}, (l'_1, \ldots, l'_m), (\theta_1, \ldots, \theta_m)$);

`// Apply the super-aggregates`
foreach $x \in \mathbf{x}$ **do**
  foreach *algebraic function* $f_{i_j} \in \{l_1, \ldots, l_m\}$ **do**
    Let g_{i_j} be the super-aggregate of f_{i_j};
    In $\mathbf{x}$, replace columns $f_{i_i}^1, \ldots, f_{i_j}^{p_{i_j}}$ by a $g_{i_j}(x.f_{i_j}^1, \ldots, f_{i_j}^{p_{i_j}})$;

return $\mathbf{x}$;

Algebraic AF Example/1

- For each combination of source (S) and destination (D), compute the number of flows whose NumBytes (NB) value exceeds the average value of NumBytes.
- Step 1: $\mathbf{x}_1 = \text{MD}(\pi[S, D]\mathbf{r}/\mathbf{b}, \mathbf{r}, l_1, \theta_1)$

  $l_1 = (\text{AVG}(NB)/Avg)$

  $\theta_1 = (\mathbf{r}.S = \mathbf{b}.S \wedge \mathbf{r}.D = \mathbf{b}.D)$
- l_1 contains algebraic aggregate AVG, hence it is translated into
  - $l_1 = (\text{COUNT}(*)/Cnt1, \text{SUM}(NB)/Sum1)$

Detail table **r**

S	D	NB
5	29	3
5	7	8
5	29	6
7	29	4
7	29	6
6	29	10

$\Longrightarrow$

$\mathbf{x}_1$

S	D	Cnt1	Sum1
5	29	2	9
5	7	1	8
7	29	2	10
6	29	1	10

$\Longrightarrow$

$\mathbf{x}_1$

S	D	Avg
5	29	4.5
5	7	8
7	29	5
6	29	10

Algebraic AF Example/2

- Step 2: $\mathbf{x} = \text{MD}(\mathbf{x}_1/\mathbf{b}, \mathbf{r}, l_2, \theta_2)$

  $l_2 = (\text{COUNT}(NB)/cnt2)$

  $\theta_2 = (\mathbf{r}.S = \mathbf{b}.S \wedge \mathbf{r}.D = \mathbf{b}.D \wedge \mathbf{r}.NB > \mathbf{b}.Avg)$

Detail table **r**

S	D	NB
5	29	3
5	7	8
5	29	6
7	29	4
7	29	6
6	29	10

$\Longrightarrow$

x

S	D	Avg	Cnt2
5	29	4.5	1
5	7	8	0
7	29	5	1
6	29	10	0

Transformation Rules

- The GMD-Join interacts with the other operators of the relational algebra.

$$\pi[\mathbf{A}]\mathcal{G}^{\theta}(B, R, \vec{l}, \vec{\theta}) = \mathcal{G}^{\theta}(\pi[\mathbf{A}']B, R, \vec{l}, \vec{\theta}) \quad (attr(\vec{\theta}) \cap \mathbf{B}) \subseteq \mathbf{A}, \mathbf{A}' = \mathbf{A} \setminus \mathbf{C} \quad \text{(E1)}$$

$$\pi[\mathbf{A}]\mathcal{G}^{\theta}(B, R, \vec{l}, \vec{\theta}) = \pi[\mathbf{A}]\mathcal{G}^{\theta}(\pi[\mathbf{A}']B, R, \vec{l}, \vec{\theta}) \quad \mathbf{A}' = (\mathbf{A} \cup (attr(\vec{\theta}) \cap \mathbf{B})) \setminus \mathbf{C} \quad \text{(E2)}$$

$$\sigma[\theta_S]\mathcal{G}^{\theta}(B, R, \vec{l}, \vec{\theta}) = \mathcal{G}^{\theta}(\sigma[\theta_S]B, R, \vec{l}, \vec{\theta}) \quad attr(\theta_s) \subseteq \mathbf{B} \quad \text{(E3)}$$

$$\mathcal{G}^{\theta}(B, R, \vec{l}, \vec{\theta}) = \mathcal{G}^{\theta}(B, \sigma[\theta_1^R \vee \cdots \vee \theta_m^R]R, \vec{l}, \vec{\theta}) \quad \theta_i = \theta_i' \wedge \theta_i^R, attr(\theta_i^R) \subseteq \mathbf{R} \quad \text{(E4)}$$

$$\sigma[>0]\mathcal{G}^{\theta}(B, R, \vec{l}, \vec{\theta}) = \sigma[>0]\mathcal{G}^{\theta}(\sigma[\theta_1^B \vee \cdots \vee \theta_m^B]B, R, \vec{l}, \vec{\theta}) \quad \theta_i = \theta_i' \wedge \theta_i^B, attr(\theta_i^B) \subseteq \mathbf{B} \quad \text{(E5)}$$

$$\mathcal{G}^{\theta}(\mathcal{G}^{\theta}(B, R_1, \vec{l}_1, \vec{\theta}_1), R_2, \vec{l}_2, \vec{\theta}_2) = \mathcal{G}^{\theta}(\mathcal{G}^{\theta}(B, R_2, \vec{l}_2, \vec{\theta}_2), R_1, \vec{l}_1, \vec{\theta}_1) \quad attr(\vec{\theta_2}) \subseteq (\mathbf{B} \cup \mathbf{R}_2) \quad \text{(E6)}$$

$$\mathcal{G}^{\theta}(\mathcal{G}^{\theta}(B, R, \vec{l}_1, \vec{\theta}_1), R, \vec{l}_2, \vec{\theta}_2) = \mathcal{G}^{\theta}(B, R, (\vec{l}_1, \vec{l}_2), (\vec{\theta}_1, \vec{\theta}_2)) \quad attr(\vec{\theta_2}) \subseteq (\mathbf{B} \cup \mathbf{R}) \quad \text{(E7)}$$

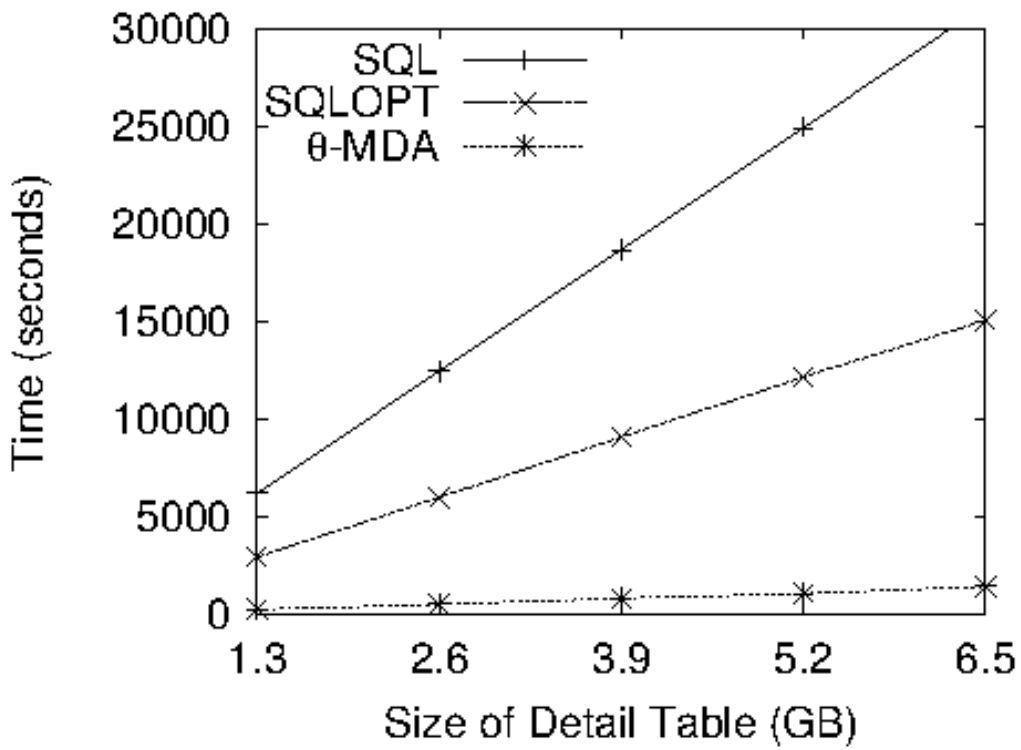
$$\mathcal{G}^{\theta}(\mathcal{G}^{\theta}(B, R_1, \vec{l}_1, \vec{\theta}_1), R_2, \vec{l}_2, \vec{\theta}_2) = \mathcal{G}^{\theta}(B, R_1, \vec{l}_1, \vec{\theta}_1) \to U \bowtie_{\theta_{\mathbf{B}}} \mathcal{G}^{\theta}(B, R_2, \vec{l}_2, \vec{\theta}_2) \to V \quad attr(\vec{\theta_2}) \subseteq (\mathbf{B} \cup \mathbf{R}_2), \quad \text{(E8)}$$
$$\theta_{\mathbf{B}} = (U.\mathbf{B} = V.\mathbf{B})$$

$$\mathcal{G}^{\theta}(B, R, \vec{l}, \vec{\theta}) = \mathcal{G}^{\theta}(B_1, R, \vec{l}, \vec{\theta}) \cup \ldots \cup \mathcal{G}^{\theta}(B_n, R, \vec{l}, \vec{\theta}) \quad B = B_1 \cup \cdots \cup B_n, \quad \text{(E9)}$$
$$B_i \cap B_j = \emptyset$$

Outline

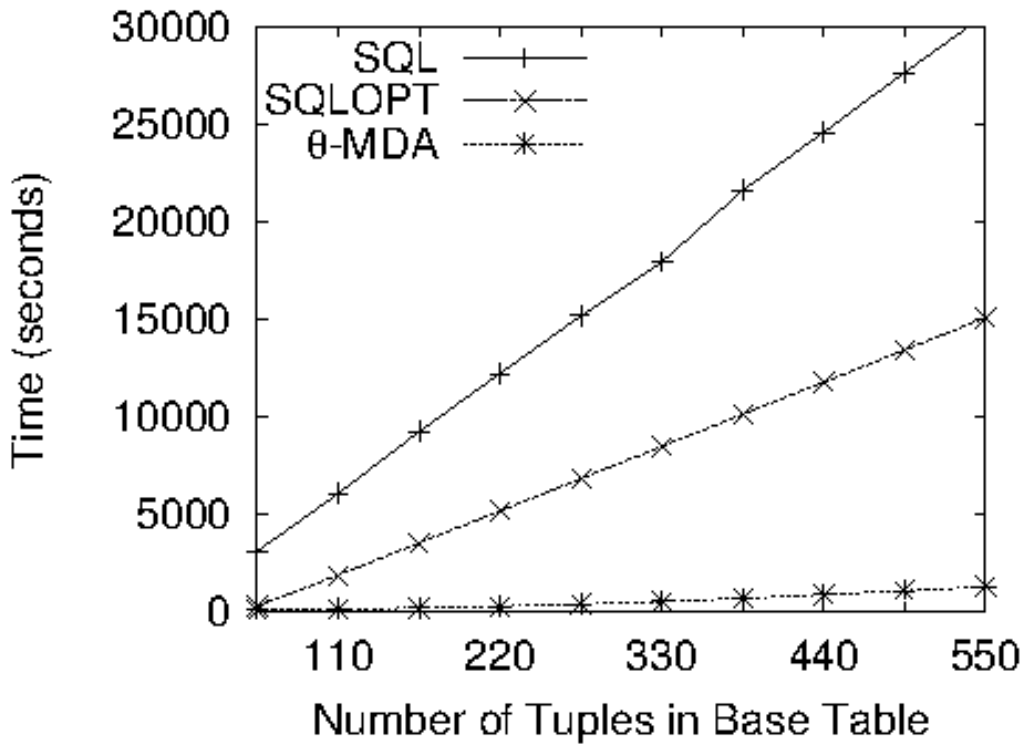
Experimental Evaluation/1

- We use Query Q1 (CntDD, CumCntD, and CumCntDD)
- Varying size of detail table (base table = 550 tuples).



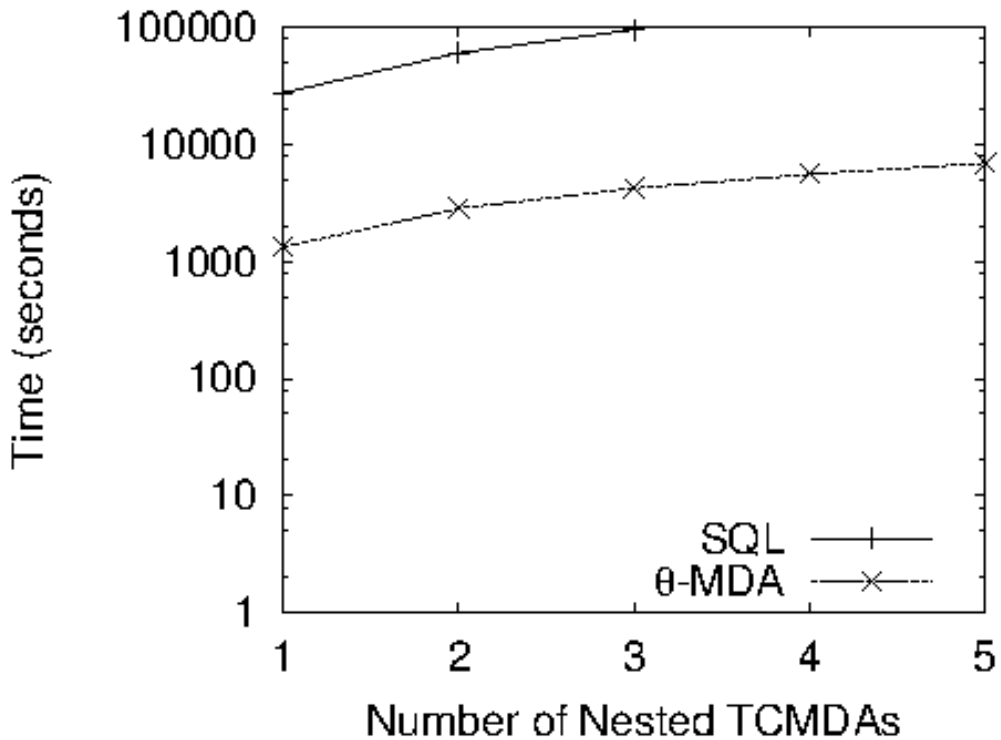
Experimental Evaluation/2

- Varying size of base table (detail table = 50M tuples).



Experimental Evaluation/3

- Varying number of nested GMD-Joins (base table = 550 tuples, detail table = 50M tuples).



Summary

- The GMD-join is an operator for ad hoc OLAP queries
  - Clear separation between the grouping and the aggregation. This provides flexibility.
  - Easy formulation of complex queries.
  - Allows to transform complex predicates into simple predicates.
- Efficient evaluation algorithms exist.
  - Single pass over detail table.
  - Further optimizations are possible to improve the performance.
- The GMDJ interacts with other algebraic operators (transformation rules).
- Intermediate results are typically small (unlike in join-based solutions)
- Systematic transformation to SQL is possible, but might be inefficient.
- 2D aggregates cannot be computed with SQL window functions.