Optimization of a DSL for a Linear Algebra approach to OLAP

João Afonso • João Fernandes

2016/2017

Tech Report

Presentation 1

Presentation 2

Final report🔗

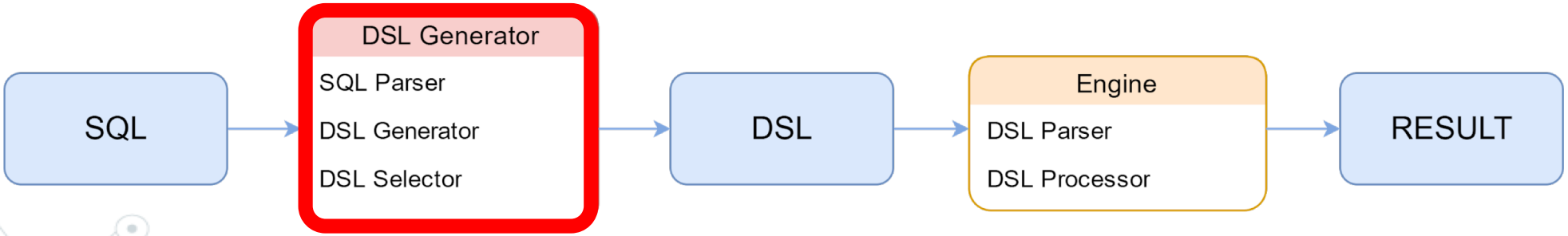
13 Feb

- ◎ Project Introduction
  - The challenge
  - HPC goal
- ◎ Work methodology definition
- ◎ Work for the week
  - Read all the given code and documentation

The challenge

◎ To automatically translate any SQL query to the already defined DSL

- For now, limited to SELECT queries



HPC goal

- Provide the most efficient DSL query for any SQL query and test machine
- Do it fast!

Work methodology

- Weekly presentation of the current progress:
  - Every Wednesday at 3pm
  - Professors:
    - Alberto Proença
    - José Nuno Oliveira
  - Collaborators:
    - Filipe Oliveira
    - Rogério Pontes

22 Feb

◎ Study the given git repositories
◎ Study the given documentation
◎ Define a clear path to develop the project
◎ Determine the necessary resources

Already Implemented

- A “hard”coded prototype of a simplified TPC-H query
  - Some code can be reused to implement the actual DSL engine;
- A DSL validator only

Necessary steps

- Implement a SQL to DSL converter
  - Implement a SQL parser
  - Design a data structure that can represent a SQL query
  - Load SQL to intermediate data structure
  - Design a data structure that can represent the DSL
  - Generate all possible conversion paths from SQL to DSL
  - Find a way to choose the most efficient one

Necessary resource

- Decide on a programming language
  - Preferably ANSI C
  - Possibly Java or C++
- SQL grammar in the chosen programming language

01 Mar

- ◎ Understand the SQL to Diagram conversion process
  - Simple queries validation with professor JNO
- ◎ Definition of the initial steps in SQL processing

Possible mistake?

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
r0	1	1	1	1	1	1	1	0	0	0
r1	0	0	0	0	0	0	0	1	1	0
r2	0	0	0	0	0	0	0	0	0	1

Figure 2: *Dense projection matrix produced from the collection of raw data (adapted from TPC-H benchmark lineitem table), from column #9 (return flag column) with 2-way association achieved recurring to GQuarks.*

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
r0	0	0	0	0	0	0	0	0	0	0
r1	0	0	0	0	0	0	0	0	0	0
r2	0	0	0	0	0	0	0	0	0	0
r3	1	1	1	1	1	1	1	0	0	0
r4	0	0	0	0	0	0	0	1	1	1

Figure 3: *Dense projection matrix produced from the collection of raw data (adapted from TPC-H benchmark lineitem table), from column #10 (line status column) with 2-way association achieved recurring to GQuarks.*

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
r0	0	0	0	0	0	0	0	0	0	0
r1	0	0	0	0	0	0	0	0	0	0
r2	0	0	0	0	0	0	0	0	0	0
r3	1	1	1	1	1	1	1	0	0	0
r4	0	0	0	0	0	0	0	0	0	0
r5	0	0	0	0	0	0	0	0	0	0
r6	0	0	0	0	0	0	0	0	0	0
r7	0	0	0	0	0	0	0	0	0	0
r8	0	0	0	0	0	0	0	0	0	0
r9	0	0	0	0	0	0	0	1	1	0
r10	0	0	0	0	0	0	0	0	0	0
r11	0	0	0	0	0	0	0	0	0	0
r12	0	0	0	0	0	0	0	0	0	0
r13	0	0	0	0	0	0	0	0	0	0
r14	0	0	0	0	0	0	0	0	0	1

Figure 4: *Projection matrix produced from the Khatri-Rao product between return flag and line status columns from lineitem table, from columns #9 and #10.*

Given the matrices A with dimensions $(m \times n)$ and B with dimensions $(o \times p)$, the resulting projection matrix has dimensions $(q \times r)$, being $q = m = o$ and $r = m \times o$. The resulting matrix dimensions can also be expressed as $((m \times o) \times n)$ as shown in figure 4.

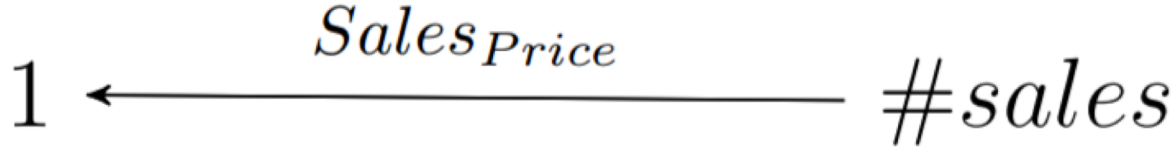
Deveria ser:

q = m x o

r = n = p

SQL to Diagram

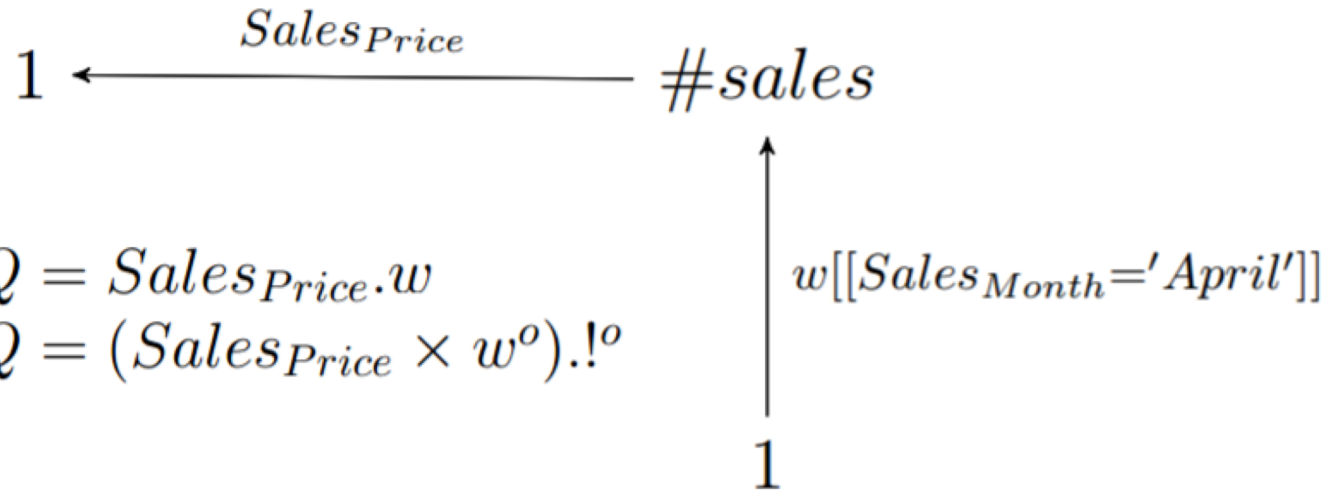
```
SELECT Sales.Price
FROM Sales
```



Sales	
Price	Month
10.4	April
11.6	May
10.1	June
15.0	December
16.3	April

SQL to Diagram

```
SELECT sum(Sales.Price)
FROM Sales
WHERE Sales.Month='April'
```

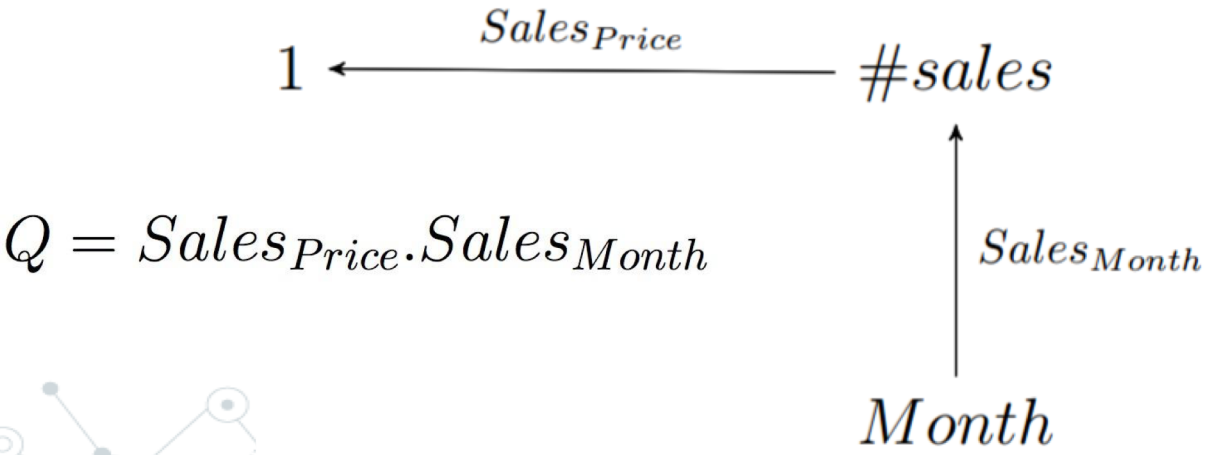


$$Q = Sales_{Price}.w$$
$$Q = (Sales_{Price} \times w^o).!^o$$

Sales	
Price	Month
10.4	April
11.6	May
10.1	June
15.0	December
16.3	April

SQL to Diagram

```
SELECT sum(Sales.Price)
FROM Sales
GROUP BY Sales.Month
```



Sales	
Price	Month
10.4	April
11.6	May
10.1	June
15.0	December
16.3	April

SQL to Diagram

- Aggregation functions
  - SUM
  - COUNT (Bang Operator)
  - AVG (SUM & COUNT)

SQL to LA method

- Make “bird’s eye” of the query topology
  - Query has to be written in verbose
    - `SELECT A.attr (...)` instead of `SELECT attr (...)`
  - Possibly implemented as a graph (**YES!**)
  - Considering the **`WHERE`** clauses
- Distinguish the Measures and Dimensions
  - A Measure has to be aggregated
  - A Dimension has to be a member of **`GROUP BY`**

SQL to LA method

- Decide on a output format for the data
  - A ⟶ B **YES** during processing

    or Maybe (A x B) ⟶ 1 **YES** at the end
    - Isomorphic
    - How do we choose between them?
- Find the most efficient path that *(to study...)*

08 Mar

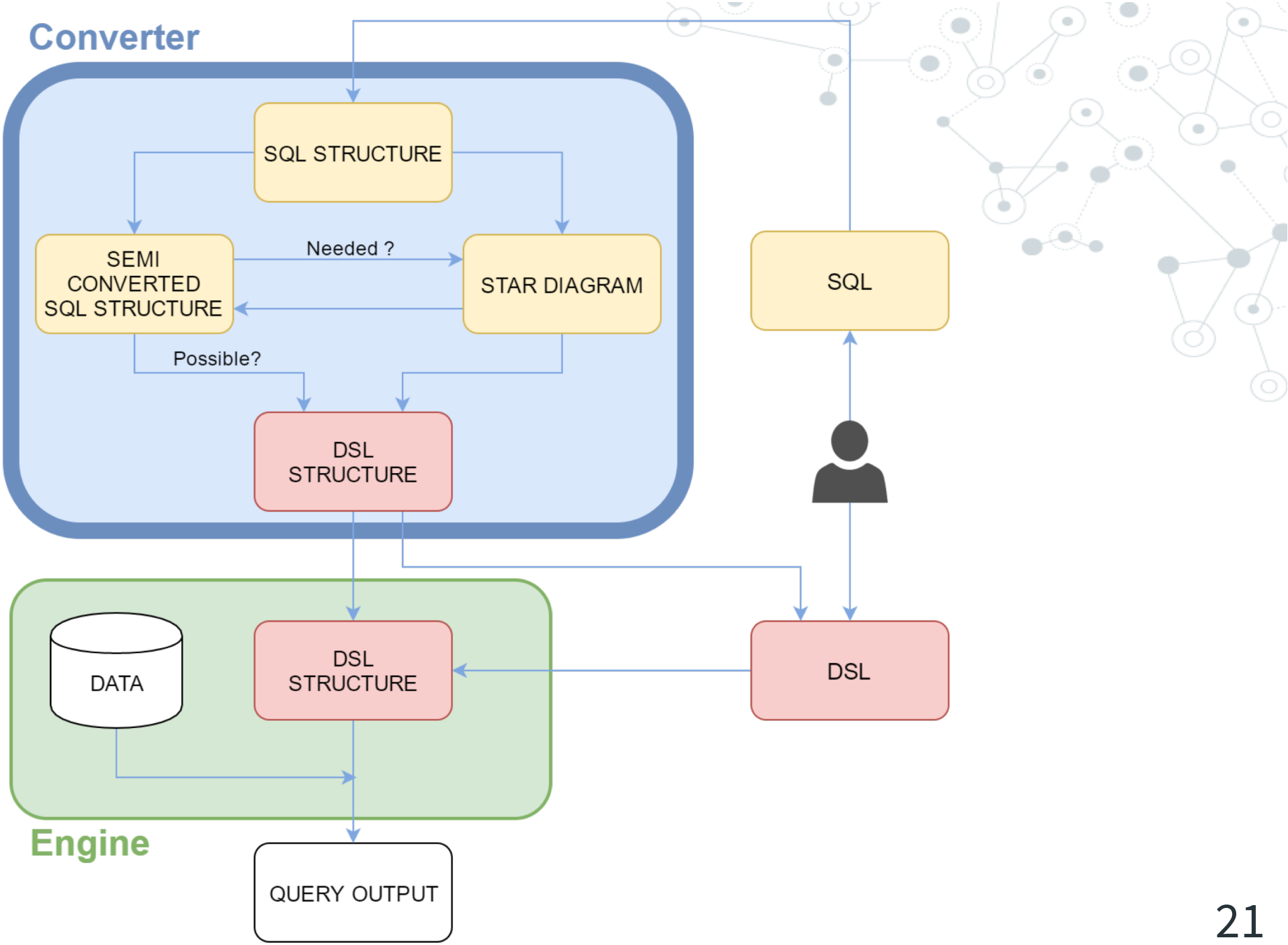
Professor João Saraiva contribution

- Python as prototyping language:
  - C - Slower development, too verbose
  - Java - Too verbose
  - Haskell - Harder to incrementally solve the problem
- C as implementation language:
  - The disadvantages are irrelevant
  - Higher performance

Python Magic

- Functional prototype made in 3 days
- Structure:
  - SQL Parser - Hard coded structure (TODO)
  - In memory SQL to LA - converter.py
  - Engine - engine.py

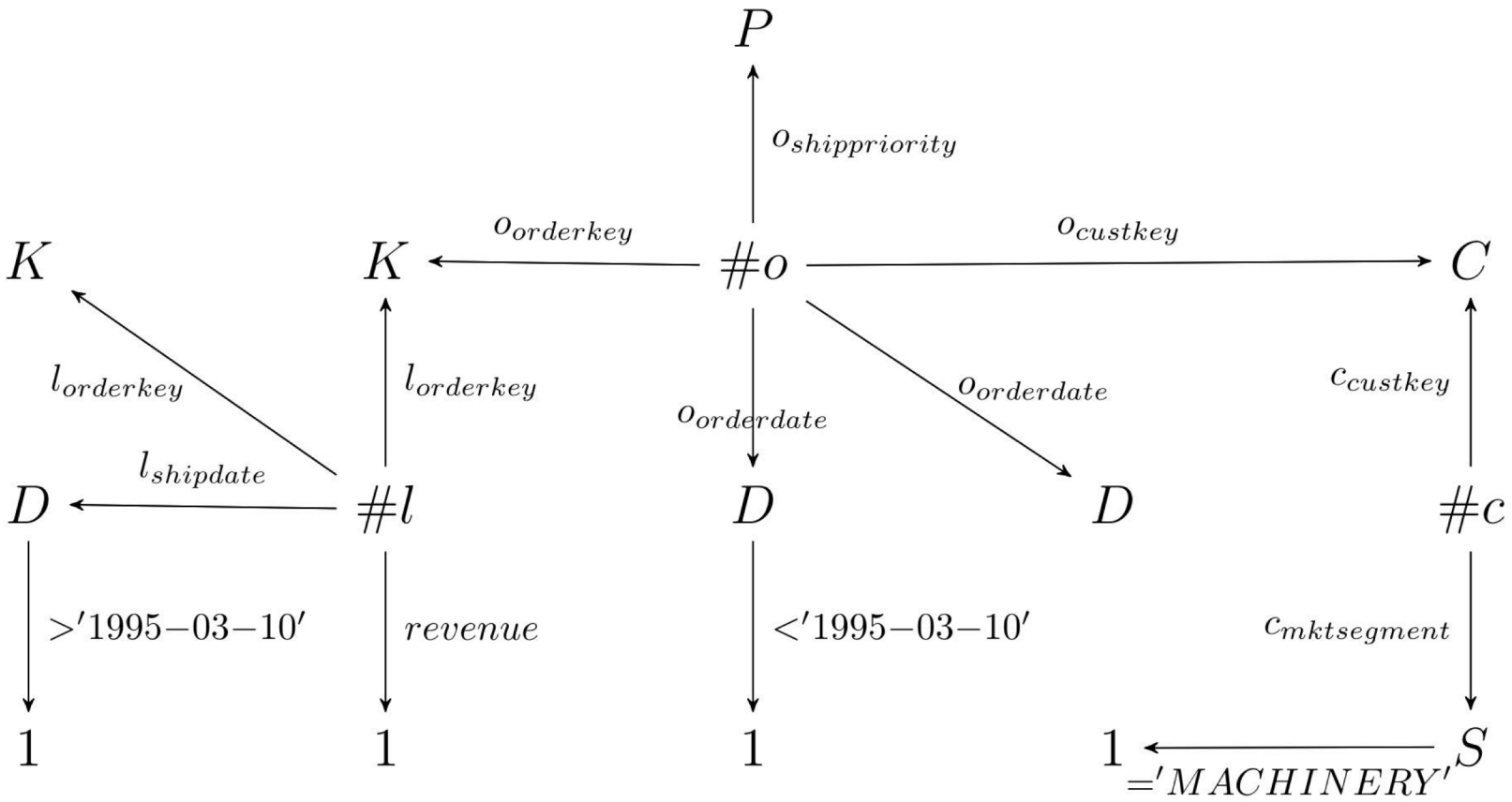
Current Dilemma


Converter
SQL STRUCTURE
SEMI CONVERTED SQL STRUCTURE
Needed ?
STAR DIAGRAM
Possible?
DSL STRUCTURE
SQL
DATA
DSL STRUCTURE
DSL
Engine
QUERY OUTPUT

15 Mar

- ◎ Hands on star diagram to LA conversion
  - Can we multiply every matrix with any product (if the types match) without losing data? **NO**
  - If the result matrix dimensions are the same, the output will also be? **NO**

Methodology PDF



Rethink operators

- Conjunctive Hadamard: `WHERE _ AND _`
- Disjunctive Hadamard: `WHERE _ OR _`
- Dot product
- Problem `JOIN` and `OR`:

```
SELECT A.a, B.b
FROM A, B
WHERE A.id = B.b
OR A.a > 10
```

Rethink operators

- Khatri-Rao
- Horizontal Khatri-Rao:
  - $1 \rightarrow A \leftarrow \#t \rightarrow A$

    $(M^{o} \,.\, N) \bigtriangledown N \;\Leftrightarrow (M^{o} \bigtriangledown N^{o})^{o}$

  - $1 \rightarrow A \leftarrow \#t$

    $(M^{O} \bigtriangledown N^{O})^{O} \;\Leftrightarrow\; M \bigtriangleright N$

$$Data = \begin{bmatrix} 5 & 15 & 25 & 15 \end{bmatrix}$$

$$M = \begin{bmatrix} 5 \\ 15 \\ 25 \end{bmatrix} > 10 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

$$N = Bitmap(Data) = \begin{matrix} 0 & 1 & 2 & 3 \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} & \begin{matrix} 5 \\ 15 \\ 25 \end{matrix} \end{matrix}$$

$$M^o = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}$$

$$M^o.N = \begin{matrix} 0 & 1 & 2 & 3 \\ \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} \end{matrix}$$

$$(M^o.N) \bigtriangledown N = \begin{matrix} 0 & 1 & 2 & 3 \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} & \begin{matrix} 5 \\ 15 \\ 25 \end{matrix} \end{matrix}$$

$$Data = \begin{bmatrix} 5 & 15 & 25 & 15 \end{bmatrix}$$

$$M = \begin{bmatrix} 5 \\ 15 \\ 25 \end{bmatrix} > 10 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

$$N = Bitmap(Data) = \begin{array}{c} \begin{matrix} 0 & 1 & 2 & 3 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} 5 \\ 15 \\ 25 \end{matrix}$$

$$M^o = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}$$

$$N^o = \begin{array}{c} \begin{matrix} 5 & 15 & 25 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix}$$

$$(M^o \bigtriangledown N^o) = \begin{array}{c} \begin{matrix} 5 & 15 & 25 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix}$$

$$(M^o \bigtriangledown N^o)^o = \begin{array}{c} \begin{matrix} 0 & 1 & 2 & 3 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} 5 \\ 15 \\ 25 \end{matrix}$$

$$Data = \begin{bmatrix} 5 & 15 & 25 & 15 \end{bmatrix}$$

$$M = \begin{bmatrix} 5 \\ 15 \\ 25 \end{bmatrix} > 10 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

$$N = Bitmap(Data) = \begin{matrix} 0 & 1 & 2 & 3 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} 5 \\ 15 \\ 25 \end{matrix}$$

$$M \triangleright N = \begin{matrix} 0 & 1 & 2 & 3 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{matrix} 5 \\ 15 \\ 25 \end{matrix}$$

+ Performance!

22 Mar

- Dependency Strategy
- Cost Model
- HPC, where is it?
- Filters
- Group By Order
- Starting point in methodology

Dependency Strategy

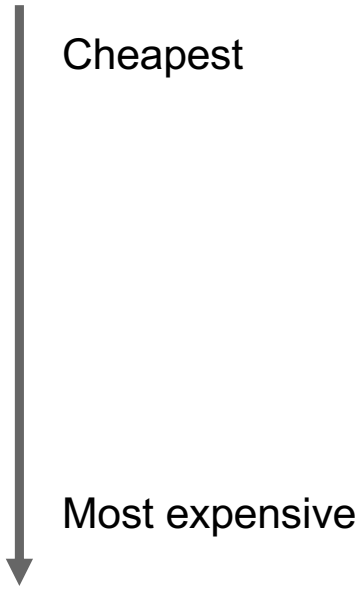
- Allow the engine to easily recognize possible operation level parallelism:
  - Eg.: A x B x C x D
    - ((A x B) x C) x D
    - (A x B) x (C x D)
- Should independent instructions be explicit or not:
  - Eg.: `PAR {A x B; C x D;}`
  - Eg.: `A x B; C x D;` (only)

Filters

- What should we do with filters of the same table?
  - Multiple simple filters:
    - `Hadamard( filter(C1), filter(C2) );`
  - One complex filter
    - `filter(C1 and C2);`
      - Could it be optimized with HEP-Frame?
- The problem here is complex expression solving:
  - (A and B) or C or (D and E)

Cost Model

- Hadamard
- Horizontal Khatri-Rao (Vec Mat)
- Khatri-Rao (Vec Mat)
- Dot Product
- Khatri-Rao (Mat Mat)

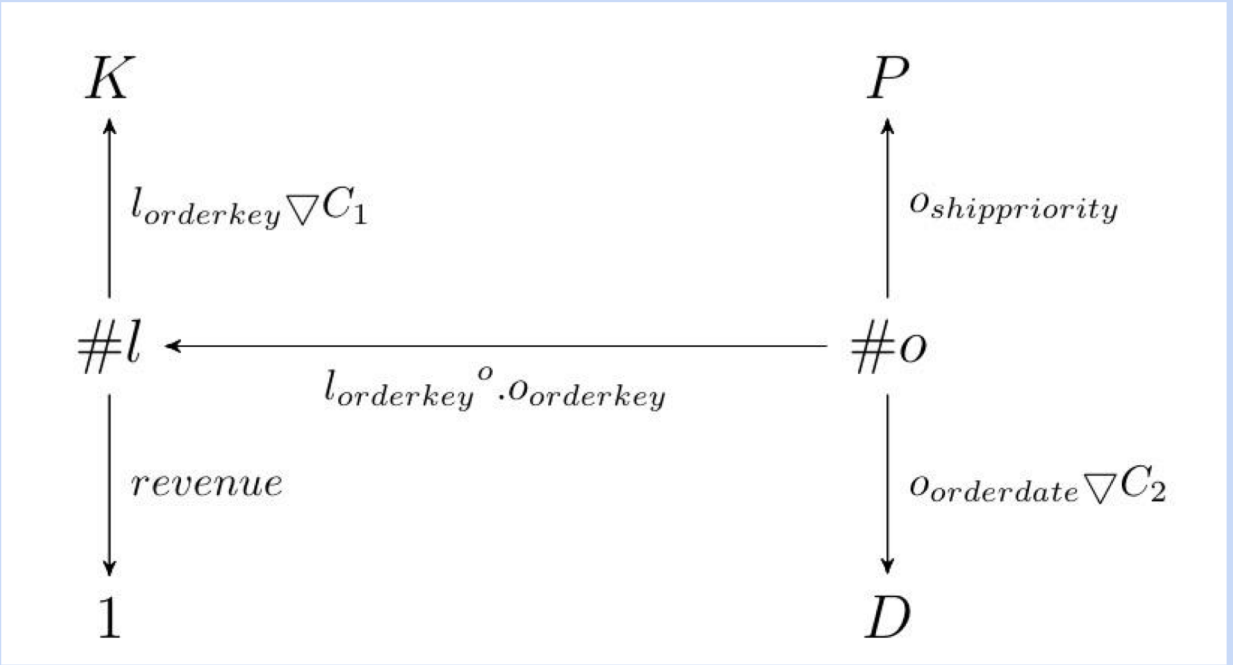


HPC, where is it?

- Should the end code be ported to C?
  - Currently, it runs instantly
  - It does not benefit from parallelism at all
  - Harder to maintain (in case the project keeps going)
- The HPC part of the project comes from the output quality

Group By Order

- TPC-H Query 3:
  - What if the **`GROUP BY`** order was:
    - P x K x D
  - Instead of:
    - P x D x K



Prototype State

1. Solve **WHERE** conditions
    1. **Hadamard** between filters on the same attribute
    2. Apply **Horizontal Khatri-Rao** where the topology allows
    3. **Dot Product** of remaining filters with the corresponding attribute
        1. No more filters
    4. **Khatri-Rao** of conditions with the **GROUP BY** dimensions in the same table
    5. If unique, **Dot Product** of remaining conditions with the table **JOIN**
    6. **Khatri-Rao** of conditions with the first **JOIN** in the table
        1. No more conditions
2. **Khatri-Rao** of same table **GROUP BY** dimensions
3. **Hadamard** of the conditions with the measures of the same table
4. **Dot Product** of the **JOIN**s (create both directions)

30 Mar

◎ The group cannot be present at the weekly meeting on wednesday 29th

05 Apr

◎ Presentation

OR (different tables)

Multiple typed diagrams.

Pros:

1. Introduced a new level of abstraction

Cons:

1. Most of the abstracted operations are replicated for each diagram
    1. Major hit in performance

Can we define a methodology to make this the last step (or possibly delete it)?

26 Apr

- TODO list:
  - Methodology
    - Aggregation functions
    - Subqueries
  - Documentation
    - Apply methodology to all TPC-H queries
  - Code everything

Methodology (revised)

1. Solve in-table operations
    1. Filters
        1. **`Hadamard`** between filters on the same attribute
        2. Apply **`Horizontal Khatri-Rao`** where the topology allows
        3. **`Dot Product`** of remaining filters with the corresponding attribute
    2. Conditions
        1. **`Hadamard`** **`AND/OR`** of all conditions

Methodology (revised)

1. …
2. **Dot Product** of the **JOIN**s (in the opposite direction to the facts table)
3. Replicate graphs depending on the existence of **OR**s
4. While there is more than one dimension matrix
    1. Merge (**Dot Product**) **JOIN**s with the **GROUP BY** and the ***conditions***
    2. **Khatri-Rao** between all dimensions (**Hadamard** `if both are 1`)
5. Join the graphs (**Hadamard** **OR**)
6. Perform the necessary operations depending on the aggregation functions

3 May

- Rethinked operations (remove dot products)
  - JOINS
  - Aggregations

JOINs

- Dot product and transpose are optional on JOINs:
  - One of the attributes must be the primary key
  - The PK bitmap is a diagonal matrix (identity)
- pk = id
- $id^{o} \cdot fk = fk$

Aggregations

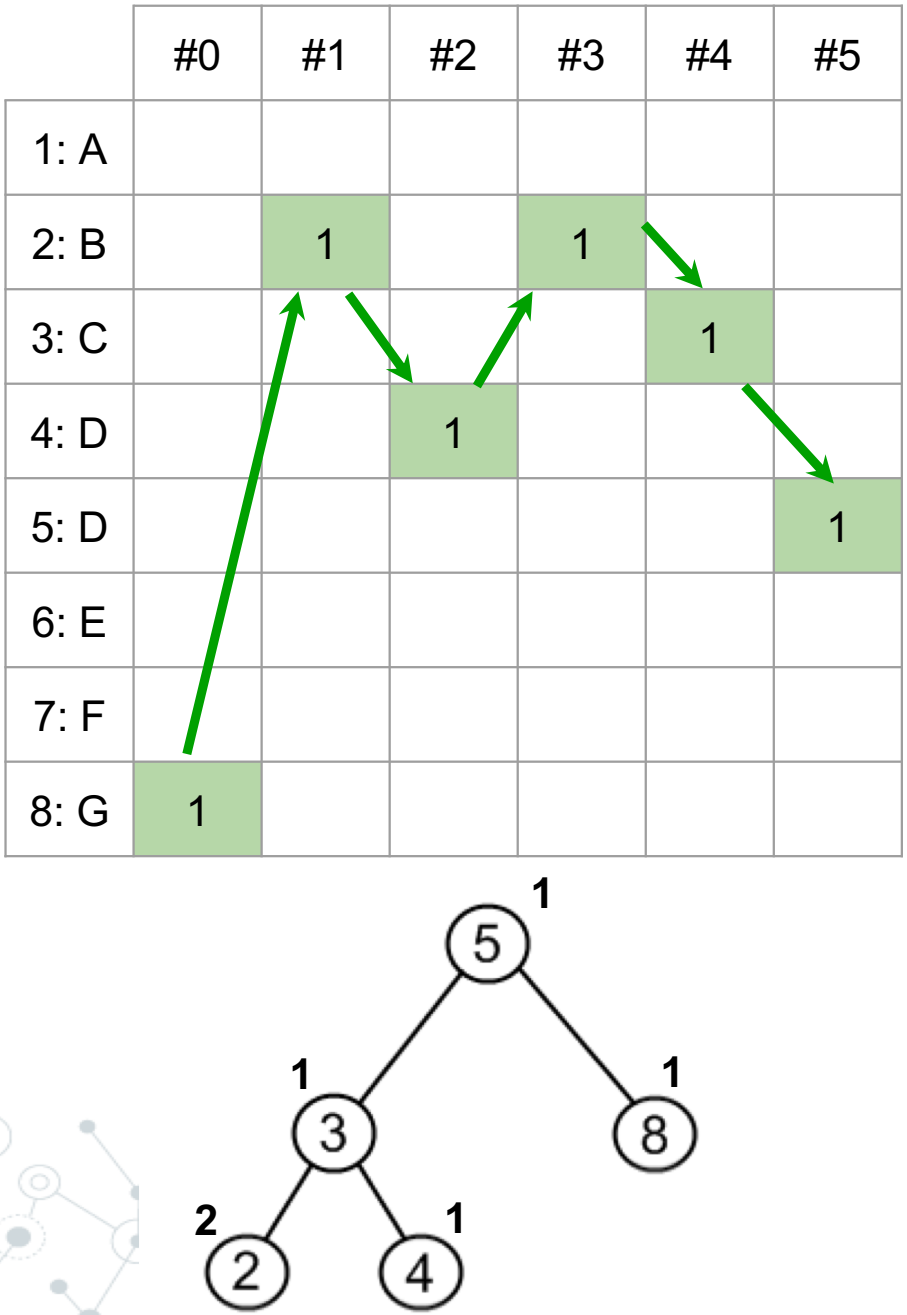
◎ Old version

- Sum: measure . matrix
- Count: bang . matrix
- Avg: Sum/Count

Aggregations

◎ New version (eyes on performance)

- Count: Implement the bang.matrix operation based on a search tree
  - ◉ CHECK NEXT SLIDE
- Sum: Khatri-Rao then Count
- Average: “Parallel” count and sum



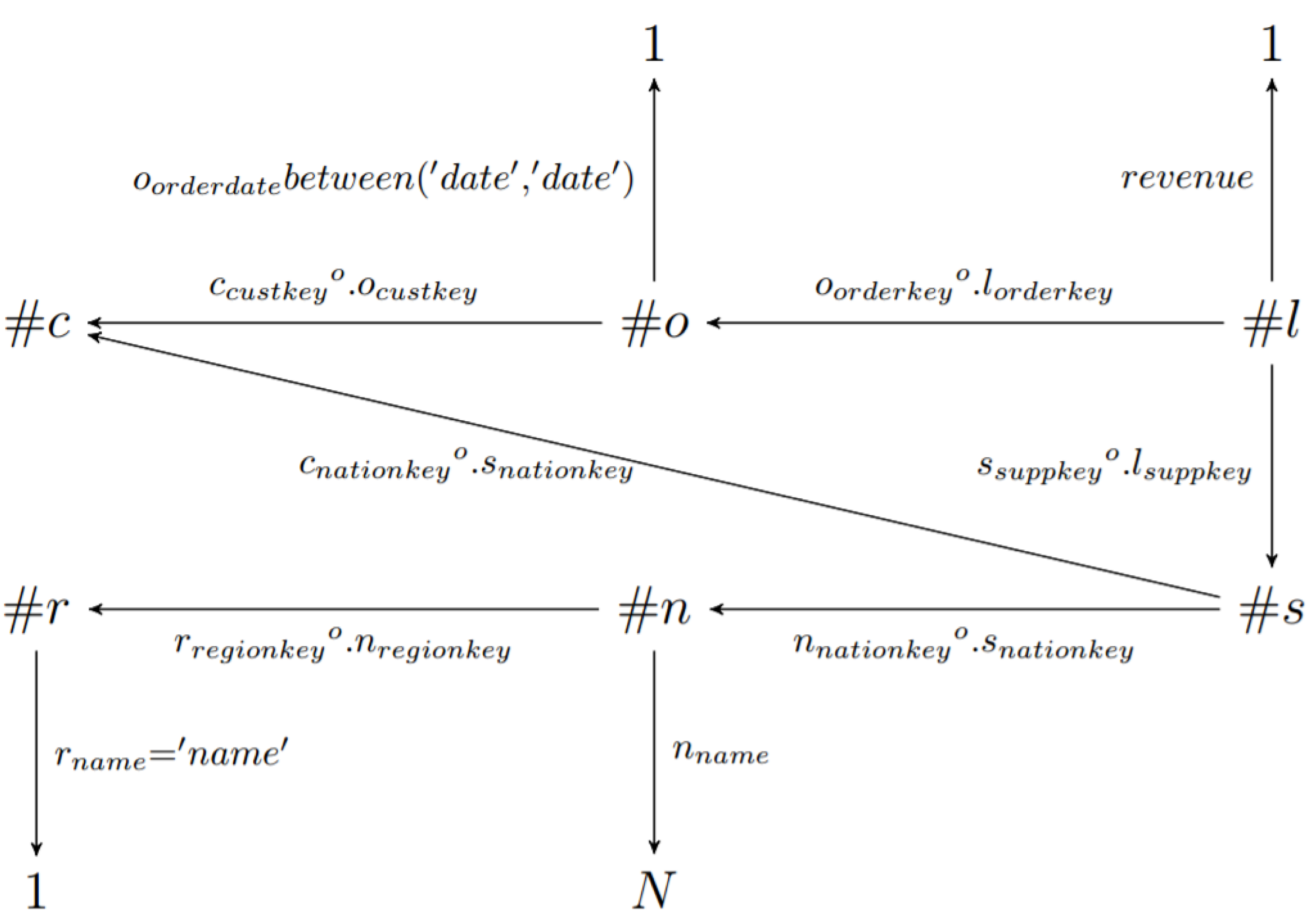
- Query 3
  - ~ **3 600 000 000** empty rows
  - ~ **0 000 011 000** rows with data
- Problem
  - `for(i=0;i<n_rows;i++)...`
  - Explained by the other group

Methodology Problem?

- TPC-H query 5:
  - What if the graph contains a loop?

Query 5

```
select
        n_name,
        sum(l_extendedprice * (1 - l_discount)) as revenue
from
        customer, orders, lineitem, supplier, nation, region
where
        c_custkey = o_custkey
        and l_orderkey = o_orderkey
        and l_suppkey = s_suppkey
        and c_nationkey = s_nationkey
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = ':1'
        and o_orderdate >= date ':2'
        and o_orderdate < date ':2' + interval '1' year
group by
        n_name
order by
        revenue desc;
```


1
1
$o_{orderdate}between('date','date')$
$revenue$
$c_{custkey}{}^{o}.o_{custkey}$
$o_{orderkey}{}^{o}.l_{orderkey}$
$\#c$
$\#o$
$\#l$
$c_{nationkey}{}^{o}.s_{nationkey}$
$s_{suppkey}{}^{o}.l_{suppkey}$
$\#r$
$\#n$
$\#s$
$r_{regionkey}{}^{o}.n_{regionkey}$
$n_{nationkey}{}^{o}.s_{nationkey}$
$r_{name}='name'$
n_{name}
1
N

30 May

◎ Methodology formalization problems
◎ Query 17 analysis

Formalization problems

- Operations in different tables have distinct solutions from the ones in the same table
  - Primary data types are **table_vector** and **table_bitmap**?
- Some operations use more than one line of SQL code (sometimes separated).
  - Eg. **hadamard** between filters in separated **where** clauses

Query 17 analysis

◎ Execute query:

```
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#13'
    and p_container = 'SM JAR'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    );
```

Query 17 analysis

◎ Get subquery value:

```
select
    0.2 * avg(l_quantity)
from
    lineitem
where
    l_partkey = p_partkey
```

avg = 5.10...

Query 17 analysis

◎ Execute query with replaced value

```
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#13'
    and p_container = 'SM JAR'
    and l_quantity < 5.10...
```

7 June

- Project overview
  - Methodology coverage
  - Hand solved queries
- Report structure

Methodology coverage

Methodology problems

1	12
2	13
3	14
4	15
5	16
6	17
7	18
8	19
9	20
10	21
11	22

Not OLAP
Triangular join
Select From Select

Report

- ◎ Introduction **(1.5 pages)**
  - ○ Context
    - ◉ OLAP
    - ◉ Linear algebra
    - ◉ TPC-H
    - ◉ Previous work
  - ○ Challenges & Goals

Report

- ◎ Foundations
  - ○ Algebraic operations **(2 pages)**
  - ○ Data representation **(1 page)**
    - ◉ Dense vectors
    - ◉ Sparse bitmaps
  - ○ Type diagrams **(1 page)**
    - ◉ Operations to diagrams
  - ○ Simple example **(3 pages)**

Report

◎ Simple example **(3 pages)**

```
SELECT sum(l_extendedprice)
FROM lineitem, orders
WHERE l_orderkey = o_orderkey
   AND o_orderdate > '1996-05-03'
   AND o_totalprice > 100000
```

Report

- ◎ Design & Implementation
  - Performance issues **(1 pages)**
  - The approach **(2 pages)**
  - Validation **(5 pages: q3)**
  - **Coverage**
- ◎ Conclusion
  - Critical analysis **(1 page)**
  - Future work **(1 page)**

14 June

- Solving `Join°.Join`
  - For each FK returns all the equal FKs
- By the id properties:
  - `Join = p°.f = id°.f = f`
  - Note: p = primary key; f = foreign key
- TPC-H query 17 needs this operation:
  - `M.f°.f`
  - Note: the first dot product is an **average** not a **sum**

M.f°.f

- Using a matrix example we got:
  - `M.f°.f=(((M∇f).!°∇ f°).!°)°`
- It's easier even though it doesn't look like it
- As explained in slide 46:
  - `(A.!°)° = avl_to_vec( avl_sum(A) )`
  - Simplified as: `sum(A)`
    - This operation is identical for: `avg`, `min`, `max`, `sum` and `count`.