# Materialized Views Selection

# Understanding Objectives

1. Select a set of views to be materialized in the following scenarios:
   1. Disk space is limited and the system is read-only
      1. Only the given user queries can be materialized
      2. Any query can be materialized
   2. Disk space is not limited and the system is read-write

# Application Objectives

1. Given a set of source tables (no more than 6) and some views over them (no more than 3), justify if a specific query over the tables can be rewritten in terms of the (materialized) views

# Answering queries using views

- Principles:
  - Query predicate $\Rightarrow$ View predicate

    Query tuples $\subset$ View tuples
  - Aggregation level must be higher or equal in the query
    - Functional dependencies can be used to check it
  - Query Aggregate must be computable from the view one.
- The problem of deciding whether it is possible to rewrite a query in terms of existing views or not is computationally complex
  - DBMSs restrict the search space to common cases by using rules

# Example of query rewriting

**Have**

```
CREATE MATERIALIZED VIEW euroSales ENABLE QUERY REWRITE
AS
    SELECT d1.city, d2.product, SUM(f.euros) AS sumEuros,
           COUNT(*) AS salesCounter
    FROM sales f, stores d1, products d2
    WHERE f.storeId = d1.Id AND f.productId = d2.Id
    GROUP BY d1.city, d2.product;
```

**Want**

```
SELECT d1.city, AVG(f.euros) AS avgSales
FROM sales f, stores d1
WHERE f.storeId = d1.Id
GROUP BY d1.city;
```

**Get**

```
SELECT city, SUM(sumEuros)/SUM(salesCounter) AS avgSales
FROM euroSales
GROUP BY city;
```

# Example of query rewriting (II)

**Have**

```
CREATE MATERIALIZED VIEW euroSales ENABLE QUERY REWRITE
AS
    SELECT d1.city, d2.product, SUM(f.euros) AS sumEuros,
           COUNT(*) AS salesCounter
    FROM sales f, stores d1, products d2
    WHERE f.storeId = d1.Id AND f.productId = d2.Id
    GROUP BY d1.city, d2.product;
```

**Want**

```
SELECT d1.city, p2.productId, c.id, SUM(f.euros) AS sales
FROM sales f, stores d1, product p2, customer c
WHERE f.storeId = d1.Id AND f.productId = p2.id AND f.customerId = c.id
GROUP BY d1.city, p2.productId, c.id;
```

**Get**

# Example of query rewriting (III)

**Have**

```
CREATE MATERIALIZED VIEW euroSales ENABLE QUERY REWRITE
AS
    SELECT d1.city, d2.product, SUM(f.euros) AS sumEuros,
           COUNT(*) AS salesCounter
    FROM sales f, stores d1, products d2
    WHERE f.storeId = d1.Id AND f.productId = d2.Id
    GROUP BY d1.city, d2.product;
```

**Want**

```
SELECT d1.city, MAX(f.euros) As mx
FROM sales f, stores d1, product p2
WHERE f.storeId = d1.Id AND f.productId = p2.Id AND p2.id = '1'
GROUP BY d1.city;
```

**Get**

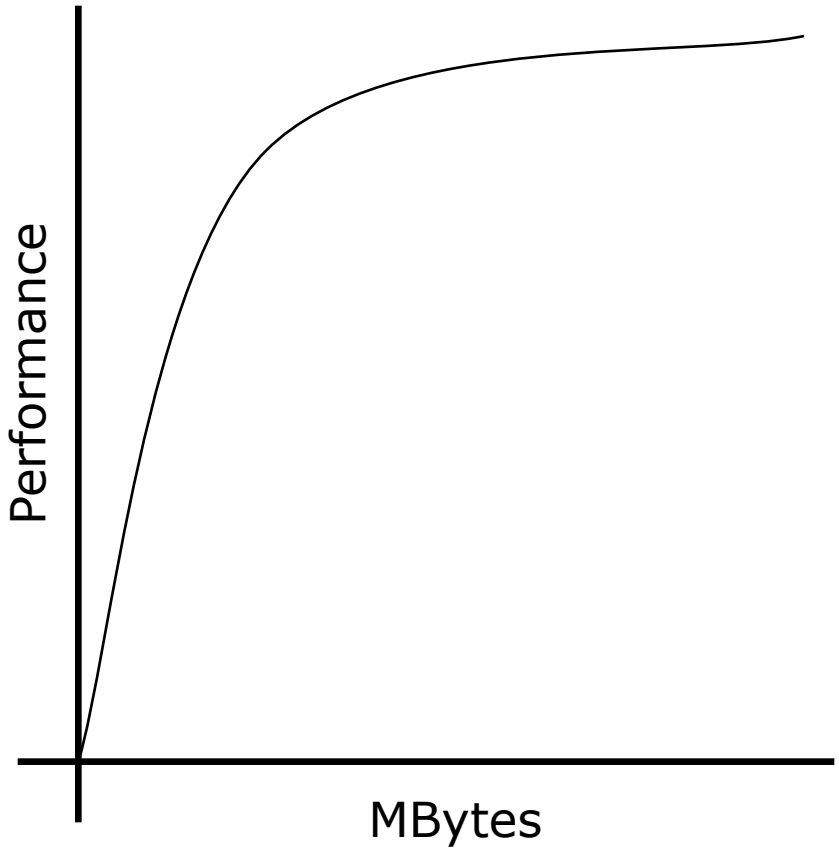
# To improve query performance …

- Build access structures (Indexes)
- Pre-calculate as much as possible
  - Redundant tables
    - Less attributes
    - Less tuples
      - Only those fulfilling the query predicate
      - Only one per combination of values of attributes in the GROUP BY
        - The sparser the basic tuples, (proportionally) the more space aggregates will use (e.g. Twelve days per year may generate twelve months per year)
    - Less space than the table
    - Less I/O to be accessed

# Problems in pre-calculating

- Cost
  - Space
  - Time
    - Query vs Modification frequency
- Consistency and rewriting control
  - Using materialized views
  - Using triggers
    - Advantages
      - Flexible
      - Allows rewriting of any query
      - Maybe efficient
    - Disadvantages
      - Difficulties management (table administration and load)
      - Ad-hoc rewriting must be implemented for each query
        - Users are bound to our rewriting tools

# Materialization trade-off




Is our Update Window enough?

# Combinatorial aggregation explosion

- Choosing the best combination of views to be materialized is NP-complex
  - A fact table with *m* dimension tables with *n* aggregation levels (including the "atomic" and "All" levels) for each one, would generate $n^m$ possible materialized views

# Candidate views to be materialized

Given a workload $W=\{q_1, q_2, q_3, \ldots\}$, and identifying queries by their GROUP BY clause, candidate views $v_i$ are those that:

a) $GB(v_i) = GB(q_j)$

b) $GB(v_i) = \bigcup_{q_j \in Q} GB(q_j)$, where $Q \subseteq W$

Provided predicates allow rewriting

Adding { aggregations / dimensions } to the SELECT if needed

# Algorithm to choose among candidates

Greedy algorithm (guarantees 63% minimum improvement):

Do

a. Consider those candidate views that fit in the available space and time
b. Sort views based on the performance improvement they induce
c. Materialize first view in the list, if it improves performance

While performance improved and available space and time

- Modify the set of materialized views as user needs evolve

# Example of materialized view selection (I)

- Taula CentMilResp(ref, pobl, edat, cand, val)
- D=1seg; C=0
- $B_{CentMilResp}$=10000; |CentMilResp| = 100000
- Ndist(pobl)= 200; Ndist(edat)=100; Ndist(cand)=10
- La informació de control ocupa el mateix que un atribut
- Tots els atributs ocupen el mateix
- Freqüència de les consultes:

35%: SELECT cand, MAX(val) FROM CentMilResp GROUP BY cand

20%: SELECT cand, edat, AVG(val), MAX(val), MIN(val) FROM CentMilResp GROUP BY cand, edat

20%: SELECT pobl, MAX(val) FROM CentMilResp GROUP BY cand, pobl

25%: SELECT pobl, MAX(val) FROM CentMilResp GROUP BY pobl

- Tenim 10140 blocs de disc

# Example of materialized view selection (II)

**C1/Q1** - SELECT cand, MAX(val) FROM CentMilResp GROUP BY cand
**C2/Q2** - SELECT cand, edat, AVG(val), MAX(val), MIN(val) FROM CentMilResp GROUP BY cand, edat
**C3/Q3** - SELECT pobl, MAX(val) FROM CentMilResp GROUP BY cand, pobl
**C4/Q4** - SELECT pobl, MAX(val) FROM CentMilResp GROUP BY pobl

# Example of materialized view selection (II)

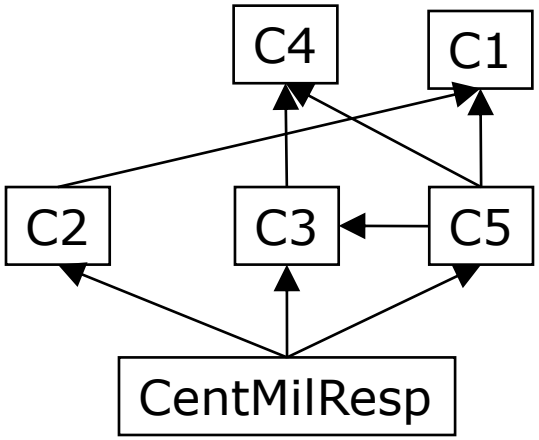
**C1/Q1** - SELECT cand, MAX(val) FROM CentMilResp GROUP BY cand
**C2/Q2** - SELECT cand, edat, AVG(val), MAX(val), MIN(val) FROM CentMilResp GROUP BY cand, edat
**C3/Q3** - SELECT pobl, MAX(val) FROM CentMilResp GROUP BY cand, pobl
**C4/Q4** - SELECT pobl, MAX(val) FROM CentMilResp GROUP BY pobl

**C5** - SELECT cand, pobl, MAX(val) FROM CentMilResp GROUP BY cand, pobl

# Example of materialized view selection (II)

**C1/Q1** - SELECT cand, MAX(val) FROM CentMilResp GROUP BY cand
**C2/Q2** - SELECT cand, edat, AVG(val), MAX(val), MIN(val) FROM CentMilResp GROUP BY cand, edat
**C3/Q3** - SELECT pobl, MAX(val) FROM CentMilResp GROUP BY cand, pobl
**C4/Q4** - SELECT pobl, MAX(val) FROM CentMilResp GROUP BY pobl

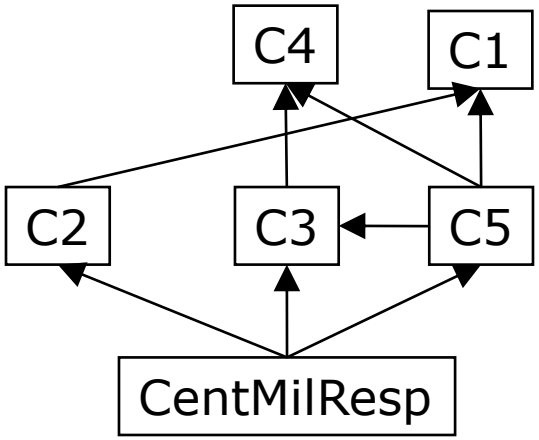
**C5** - SELECT cand, pobl, MAX(val) FROM CentMilResp GROUP BY cand, pobl



# Example of materialized view selection (II)

**C1/Q1** - SELECT cand, MAX(val) FROM CentMilResp GROUP BY cand
**C2/Q2** - SELECT cand, edat, AVG(val), MAX(val), MIN(val) FROM CentMilResp GROUP BY cand, edat
**C3/Q3** - SELECT pobl, MAX(val) FROM CentMilResp GROUP BY cand, pobl
**C4/Q4** - SELECT pobl, MAX(val) FROM CentMilResp GROUP BY pobl

**C5** - SELECT cand, pobl, MAX(val) FROM CentMilResp GROUP BY cand, pobl



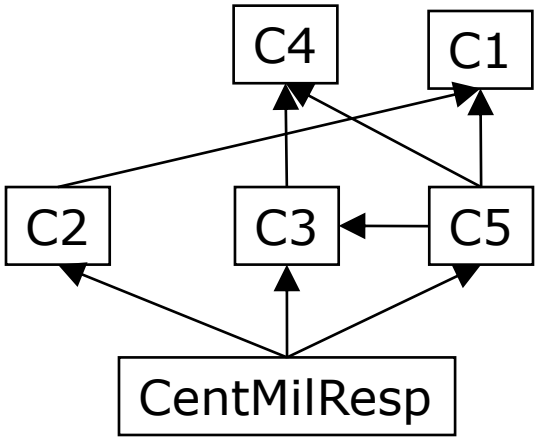

FILES AGREGACIÓ
mín(Nfiles$_0$, Ndist($a_1$) * ... * Ndist($a_n$))

C2: mín(100000, 10 * 100) = 1000
C3: mín(100000, 10 * 200) = 2000

# Example of materialized view selection (II)

**C1/Q1** - SELECT cand, MAX(val) FROM CentMilResp GROUP BY cand
**C2/Q2** - SELECT cand, edat, AVG(val), MAX(val), MIN(val) FROM CentMilResp GROUP BY cand, edat
**C3/Q3** - SELECT pobl, MAX(val) FROM CentMilResp GROUP BY cand, pobl
**C4/Q4** - SELECT pobl, MAX(val) FROM CentMilResp GROUP BY pobl

**C5** - SELECT cand, pobl, MAX(val) FROM CentMilResp GROUP BY cand, pobl




FILES AGREGACIÓ
$mín(Nfiles_0, Ndist(a_1) * ... * Ndist(a_n))$

C2: mín(100000, 10 * 100) = 1000
C3: mín(100000, 10 * 200) = 2000

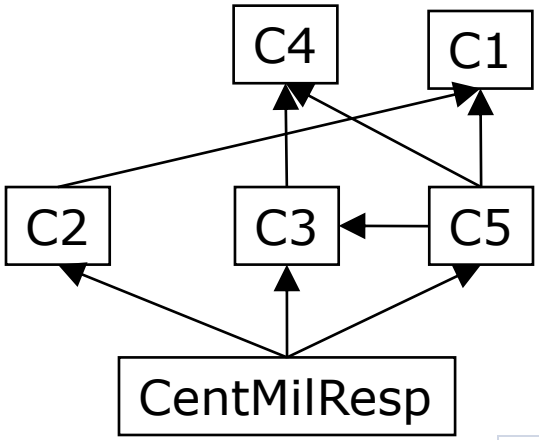
ESPAI AGREGACIÓ
$E_0 * (Natr/Natr_0) * (Nfiles/Nfiles_0)$

C2: 10000 * (6/6) * (1000/100000) = 100
C3: 10000 * (3/6) * (2000/100000) = 100

# Example of materialized view selection (II)

**C1/Q1** - SELECT cand, MAX(val) FROM CentMilResp GROUP BY cand
**C2/Q2** - SELECT cand, edat, AVG(val), MAX(val), MIN(val) FROM CentMilResp GROUP BY cand, edat
**C3/Q3** - SELECT pobl, MAX(val) FROM CentMilResp GROUP BY cand, pobl
**C4/Q4** - SELECT pobl, MAX(val) FROM CentMilResp GROUP BY pobl

**C5** - SELECT cand, pobl, MAX(val) FROM CentMilResp GROUP BY cand, pobl




| | |
|---|---|
| C1 | 1 |
| C2 | 100 |
| C3 | 100 |
| C4 | 10 |
| C5 | 134 |

FILES AGREGACIÓ
$mín(Nfiles_0, Ndist(a_1) * ... * Ndist(a_n))$

C2: mín(100000, 10 * 100) = 1000
C3: mín(100000, 10 * 200) = 2000

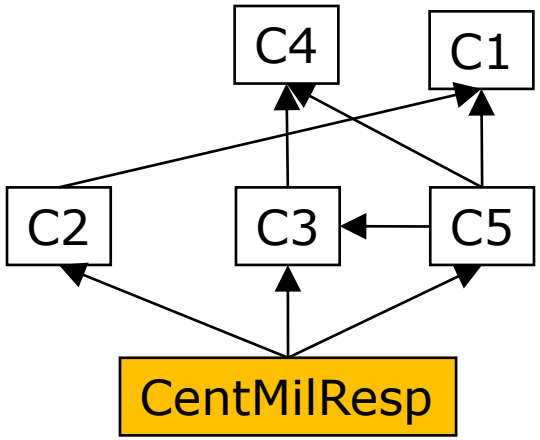
ESPAI AGREGACIÓ
$E_0 * (Natr/Natr_0) * (Nfiles/Nfiles_0)$

C2: 10000 * (6/6) * (1000/100000) = 100
C3: 10000 * (3/6) * (2000/100000) = 100

# Example of materialized view selection (II)

Cost if there is no materialized view:

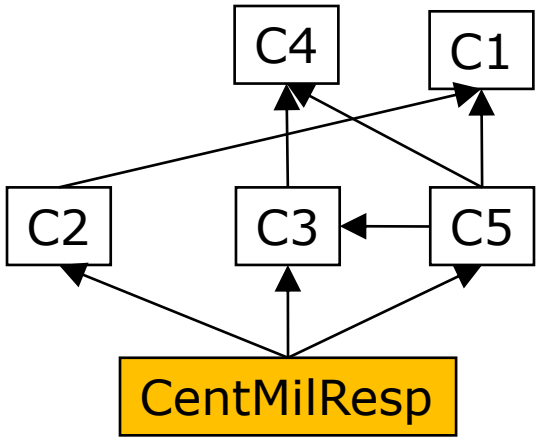
- Time: 10000 sec/query
- Space: 10000 blocks



# Example of materialized view selection (II)

Cost if there is no materialized view:

- Time: 10000 sec/query
- Space: 10000 blocks

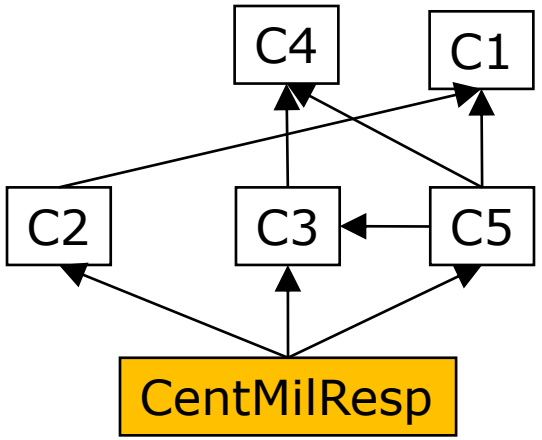



| | Q1 (35%) | Q2 (20%) | Q3 (20%) | Q4 (25%) | Avg |
|---|---|---|---|---|---|
| C1 | 1 | 10000 | 10000 | 10000 | 6500,4 |
| C2 | 100 | 100 | 10000 | 10000 | 4555 |
| C3 | 10000 | 10000 | 100 | 100 | 5545 |
| C4 | 10000 | 10000 | 10000 | 10 | 7502,5 |
| C5 | 134 | 10000 | 134 | 134 | 2107,2 |

# Example of materialized view selection (II)

Cost if there is no materialized view:

- Time: 10000 sec/query
- Space: 10000 blocks

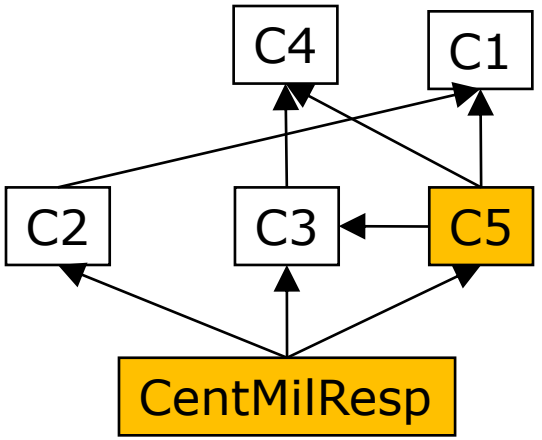



| | Q1 (35%) | Q2 (20%) | Q3 (20%) | Q4 (25%) | Avg |
|---|---|---|---|---|---|
| C1 | 1 | 10000 | 10000 | 10000 | 6500,4 |
| C2 | 100 | 100 | 10000 | 10000 | 4555 |
| C3 | 10000 | 10000 | 100 | 100 | 5545 |
| C4 | 10000 | 10000 | 10000 | 10 | 7502,5 |
| C5 | 134 | 10000 | 134 | 134 | 2107,2 |

# Example of materialized view selection (III)

Cost if C5 is materialized:

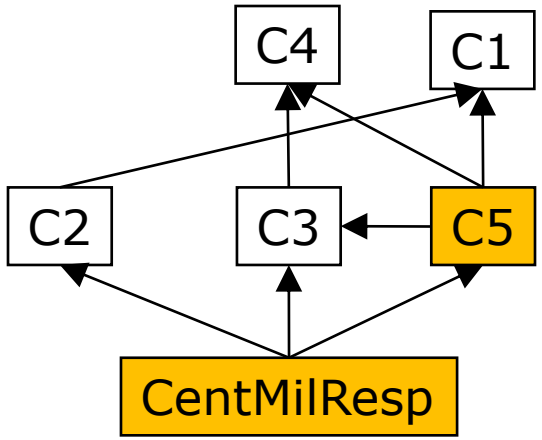
- Time: 2107,2 sec/query
- Space: 10134 blocks



# Example of materialized view selection (III)

Cost if C5 is materialized:

- Time: 2107,2 sec/query
- Space: 10134 blocks

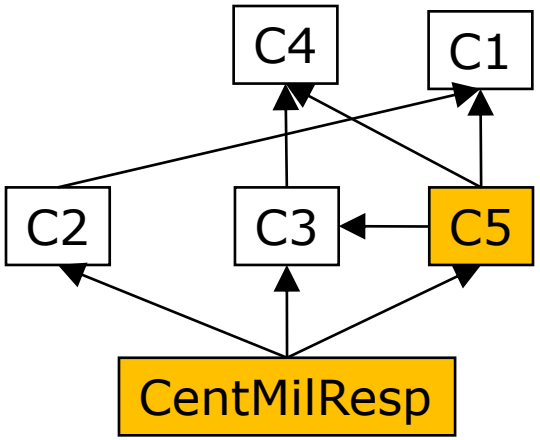



| | Q1 (35%) | Q2 (20%) | Q3 (20%) | Q4 (25%) | Avg |
|---|---|---|---|---|---|
| C1 | 1 | 10000 | 134 | 134 | 2060,7 |
| C2 | 100 | 100 | 134 | 134 | 115,3 |
| C3 | 134 | 10000 | 100 | 100 | 2091,9 |
| C4 | 134 | 10000 | 134 | 10 | 2076,2 |

# Example of materialized view selection (III)

Cost if C5 is materialized:

- Time: 2107,2 sec/query
- Space: 10134 blocks

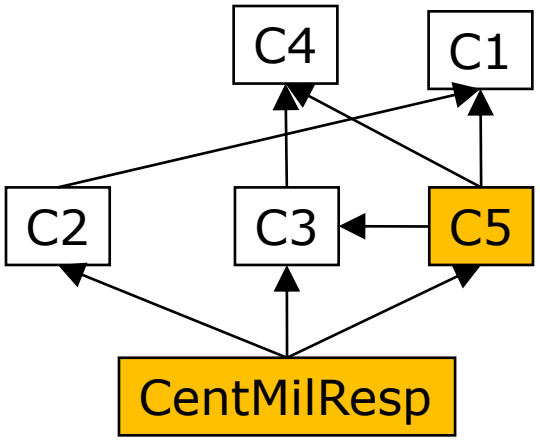



| | Q1 (35%) | Q2 (20%) | Q3 (20%) | Q4 (25%) | Avg |
|---|---|---|---|---|---|
| C1 | 1 | 10000 | 134 | 134 | 2060,7 |
| ~~C2~~ | ~~100~~ | ~~100~~ | ~~134~~ | ~~134~~ | ~~115,3~~ |
| ~~C3~~ | ~~134~~ | ~~10000~~ | ~~100~~ | ~~100~~ | ~~2091,9~~ |
| ~~C4~~ | ~~134~~ | ~~10000~~ | ~~134~~ | ~~10~~ | ~~2076,2~~ |

# Example of materialized view selection (III)

Cost if C5 is materialized:

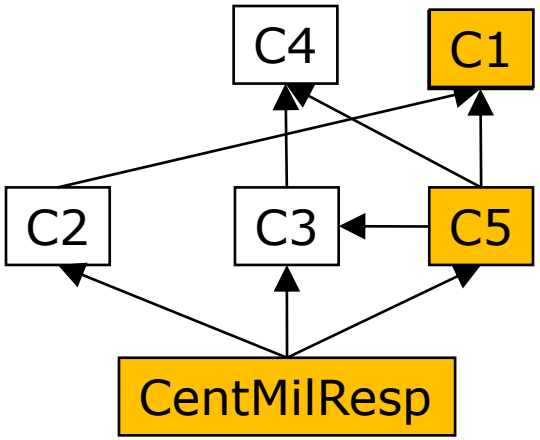
- Time: 2107,2 sec/query
- Space: 10134 blocks




| | Q1 (35%) | Q2 (20%) | Q3 (20%) | Q4 (25%) | Avg |
|---|---|---|---|---|---|
| C1 | 1 | 10000 | 134 | 134 | 2060,7 |
| C2 | 100 | 100 | 134 | 134 | 115,3 |
| C3 | 134 | 10000 | 100 | 100 | 2091,9 |
| C4 | 134 | 10000 | 134 | 10 | 2076,2 |

# Example of materialized view selection (III)

Cost if C5 is materialized:
- Time: 2107,2 sec/query
- Space: 10134 blocks




| | Q1 (35%) | Q2 (20%) | Q3 (20%) | Q4 (25%) | Avg |
|---|---|---|---|---|---|
| C1 | 1 | 10000 | 134 | 134 | 2060,7 |
| C2 | 100 | 100 | 134 | 134 | 115,3 |
| C3 | 134 | 10000 | 100 | 100 | 2091,9 |
| C4 | 134 | 10000 | 134 | 10 | 2076,2 |

Cost if C1 and C5 are materialized:
- Time: 2060,7 sec/query
- Space: 10135 blocks

# Summary

- Pre-aggregation
- Materialized view selection

# Bibliography


- M. Golfarelli and S. Rizzi. *Data Warehouse Design*. McGrau-Hill, 2009
- The BI Verdict: Database explosion.

  www.bi-verdict.com/fileadmin/FreeAnalyses/DatabaseExplosion.htm. Updated, February 2005. Visited, June 2010