Column-Oriented Databases

LAB MATERIAL

DO I REALLY NEED A COLUMN-STORE?

Do I Really Need a Column-Oriented DB?

- Traditional (row-oriented) databases provide means for improving performance in front of read-only queries
 - Vertical partitioning (improves useful read ratio)
 - Each table split in a set of two-columned partitions (key, attribute)
 - Use index-only query plans (no table access)
 - Create a collection of indexes that cover all columns used in a query
 - Use a collection of materialized views such that there is a view with the exact columns needed to answer the query

Tuning for Read-Only Queries

- Objective: Refresh the main tuning techniques for read-only queries
- □ Tasks:
 - 1. (5') With a teammate apply (and understand) the following tuning to the database and query shown below
 - Vertical partitioning
 - II. Index-only query answering
 - III. Materialized views
 - 2. (10') Discuss (under what circumstances) which is best or, in other words, what access plan would result for each strategy
 - 3. (5') Think tank

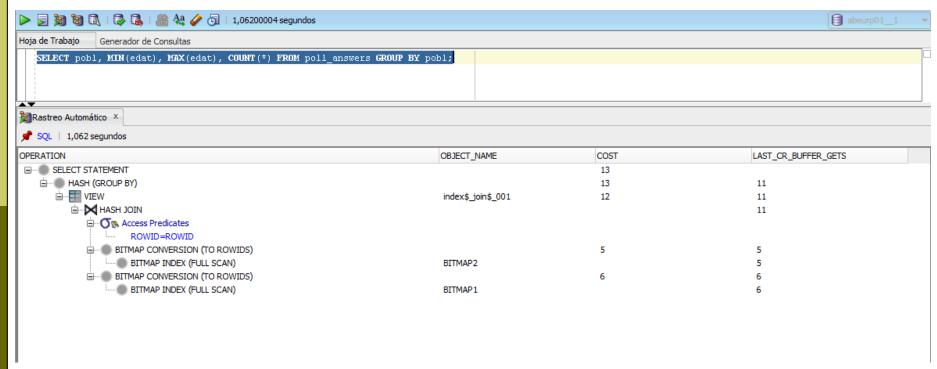
Query:

```
SELECT llibreId, SUM(numUnitats) FROM compres c, llibre l
WHERE c.llibreId = l.llibreId AND editorial = 'RBA'
GROUP BY llibreId
```

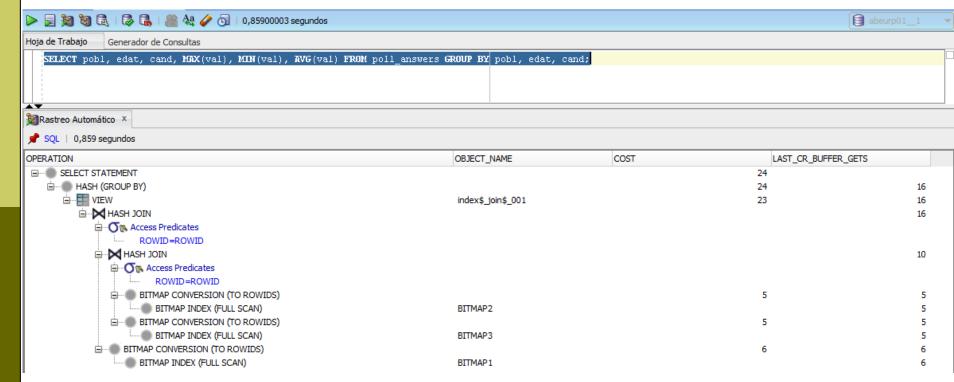
Database:

```
Compres (<u>llibreId<sup>FK</sup></u>, <u>date</u>, preu, numUnitats)
Llibre(llibreId, autor, any, editorial, ISBN)
```

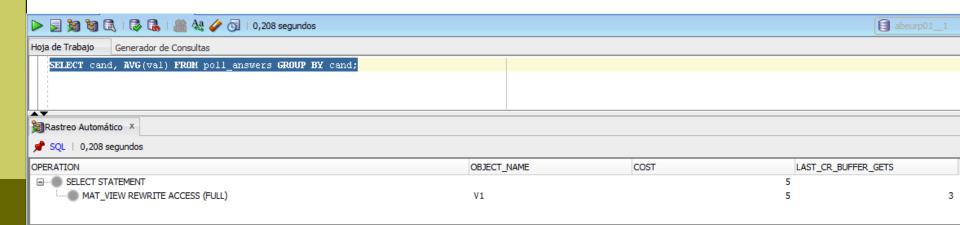
- Objective: Identify the bottlenecks remaining in a row-oriented database after tuning it with bitmaps and materialized views
- Tasks:
 - (5') Understand the access plan below and identify what operations are taking more computation time
 - 2. (5') Discussion



- Objective: Identify the bottlenecks remaining in a row-oriented database after tuning it with bitmaps and materialized views
- Tasks:
 - (5') Understand the access plan below and identify what operations are taking more computation time
 - 2. (5') Discussion



- Objective: Identify the bottlenecks remaining in a row-oriented database after tuning it with bitmaps and materialized views
- Tasks:
 - (5') Understand the access plan below and identify what operations are taking more computation time
 - 2. (5') Discussion



- Objective: Identify the bottlenecks remaining in a row-oriented database after tuning it with vertical fragmentation
- Tasks:
 - (5') Understand the access plan below and identify what operations are taking more computation time
 - 2. (5') Discussion



Conclusions

- Column-oriented DBs implement specific optimization mechanisms that outperform tuned row-oriented RDBMS (up to 50-75% of improvement). This is due to:
 - Implement vertical fragmentation in an efficient manner
 - As result, while row-oriented databases need joins to reconstruct the original tuples, column-oriented DBs do not
 - Compression is not as efficiently applied in row-oriented DBs as in column-oriented DBs
 - Ligthweight compression and fixed-size records
 - Row-oriented DBs do not apply specific query processing techniques tailored for columnar databases
 - Vectorized query processing
- As such, trying to emulate a column-store in a row-store does not yield good performance results

Bibliography

- Daniel J. Abadi, Samuel R. Madden and Nabil Hachem. Column-Stores Vs. Row-Stores: How Different Are They Really?
 - Proceedings of SIGMOD 2008, pp. 967-980
- Hector Garcia-Molina, Jeffrey D. Ullman and Jennifer Widow. Database Systems. Pearson Prentice Hall, 2009
- Hasso Plattner and Alexander Zeier. *In-Memory Data Management*. Springer, 2011
- George P. Copeland , Setrag N. Khoshafian. A Decomposition Storage Model.
 - Proceedings of SIGMOD 1985