

# Column-Oriented Databases Lab

# Do I Really Need a Column-Oriented Database?

LAB DISCUSSION

# 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*
  - I. *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:*

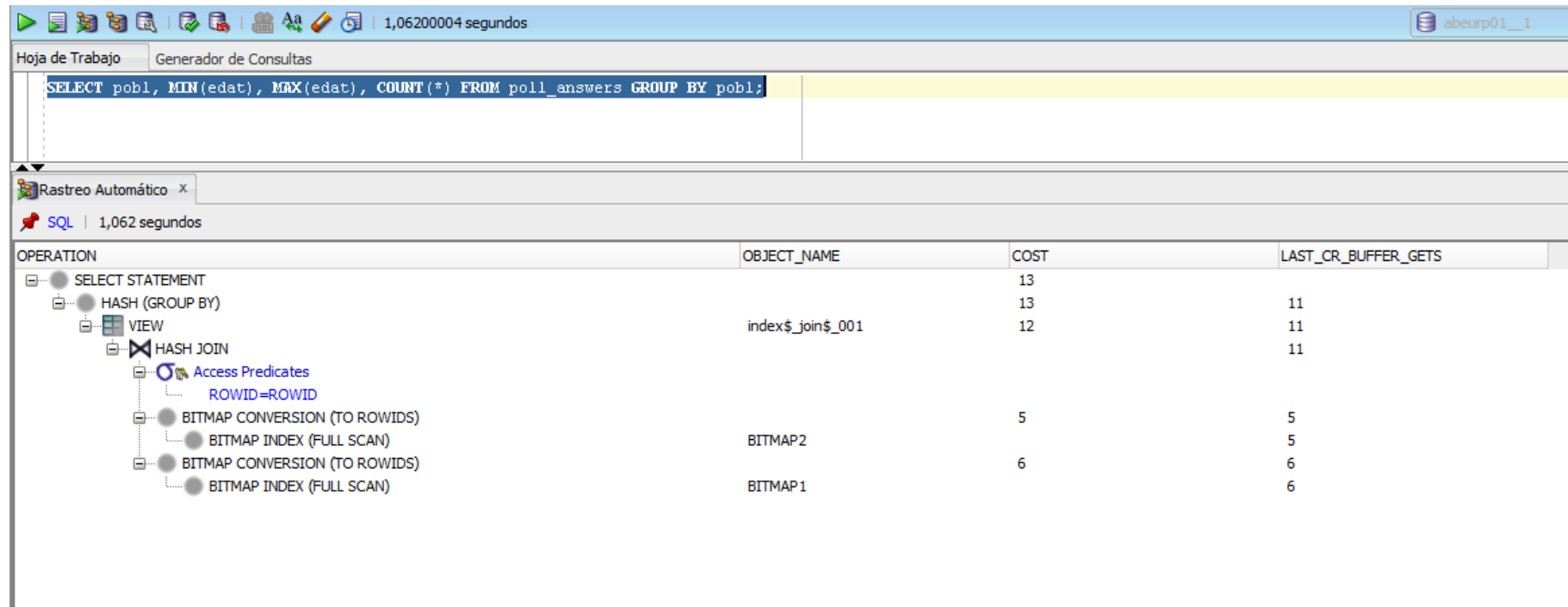
```
SELECT llibreId, SUM(numUnitats) FROM compres c, llibre l
WHERE cllibreId = lllibreId AND editorial = 'RBA'
GROUP BY llibreId
```

*Database:*

```
Compres(llibreIdFK, date, preu, numUnitats)
Llibre(llibreId, autor, any, editorial, ISBN)
```

# Bottlenecks in Row-Oriented DBs

- *Objective: Identify the bottlenecks remaining in a row-oriented database after tuning it with bitmaps and materialized views*
- *Tasks:*
  1. (5') *Understand the access plan below and identify what operations are taking more computation time*
  2. (5') *Discussion*

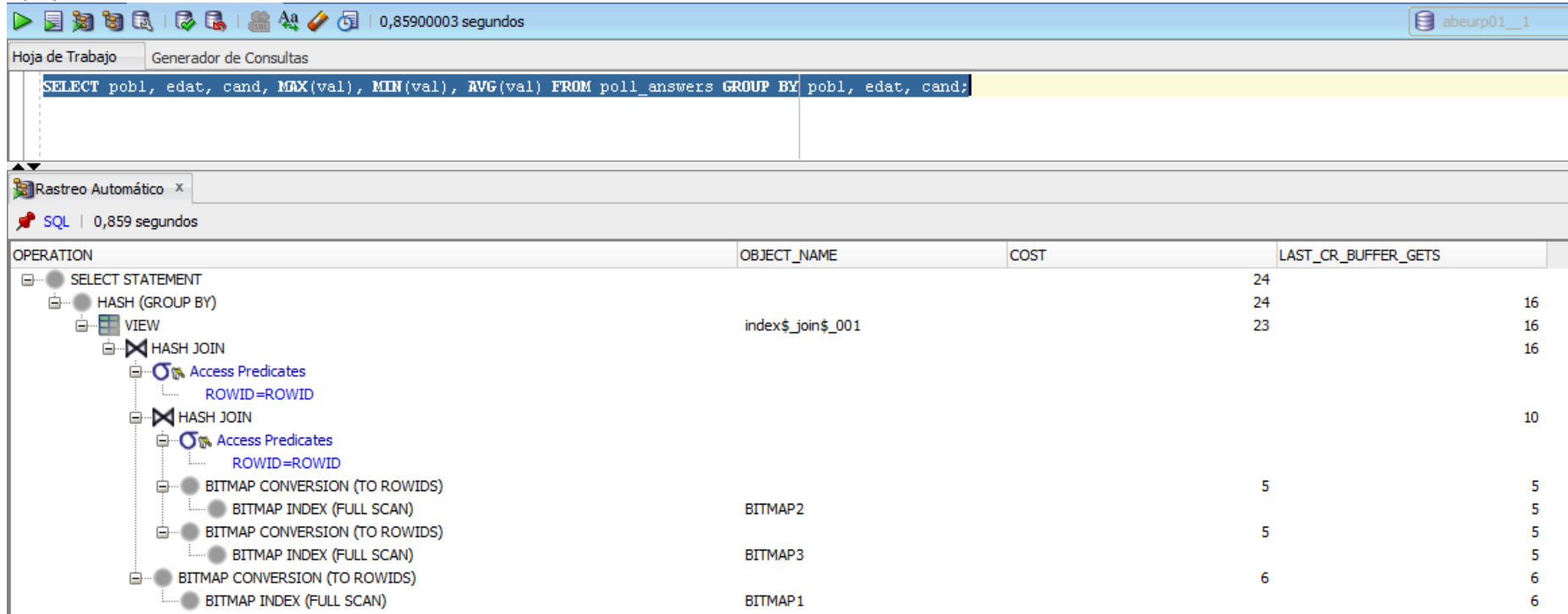


The screenshot shows the Oracle SQL Developer interface. At the top, a toolbar contains various icons for file operations, editing, and execution. Below the toolbar, the 'Hoja de Trabajo' (Worksheet) tab is active, displaying the SQL query: `SELECT pobl, MIN(edat), MAX(edat), COUNT(*) FROM poll_answers GROUP BY pobl;`. The 'Rastreo Automático' (Automatic Trace) window is open, showing the execution plan for the query. The plan is displayed in a table with columns: OPERATION, OBJECT\_NAME, COST, and LAST\_CR\_BUFFER\_GETS. The plan shows a hierarchical structure of operations, starting with a SELECT STATEMENT, followed by a HASH (GROUP BY) operation, a VIEW operation, a HASH JOIN operation, and finally two BITMAP INDEX (FULL SCAN) operations on BITMAP2 and BITMAP1. The costs and buffer gets are listed for each operation.

OPERATION	OBJECT_NAME	COST	LAST_CR_BUFFER_GETS
SELECT STATEMENT		13	
HASH (GROUP BY)		13	11
VIEW	index\$_join\$_001	12	11
HASH JOIN			11
Access Predicates			
ROWID=ROWID			
BITMAP CONVERSION (TO ROWIDS)		5	5
BITMAP INDEX (FULL SCAN)	BITMAP2	5	5
BITMAP CONVERSION (TO ROWIDS)		6	6
BITMAP INDEX (FULL SCAN)	BITMAP1	6	6

# Bottlenecks in Row-Oriented DBs

- *Objective: Identify the bottlenecks remaining in a row-oriented database after tuning it with bitmaps and materialized views*
- *Tasks:*
  1. (5') *Understand the access plan below and identify what operations are taking more computation time*
  2. (5') *Discussion*



The screenshot shows the Oracle SQL Developer interface. At the top, the title bar indicates the session is 'abeurp01\_1' and the execution time is '0,85900003 segundos'. Below the title bar, the 'Hoja de Trabajo' (Worksheet) tab is active, displaying the following SQL query:

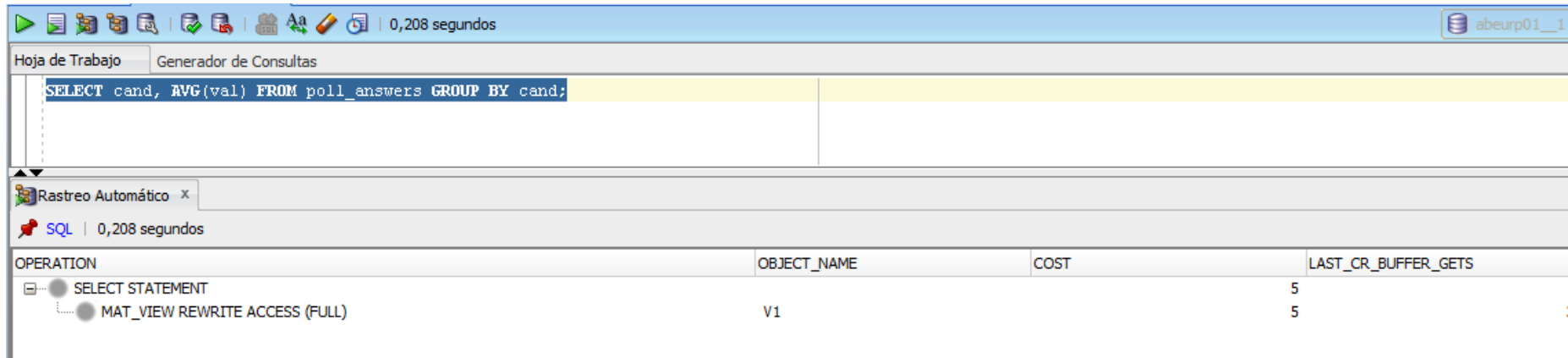
```
SELECT pobl, edat, cand, MAX(val), MIN(val), AVG(val) FROM poll_answers GROUP BY pobl, edat, cand;
```

Below the query editor, the 'Rastreo Automático' (Automatic Trace) window is open, showing the execution plan for the SQL statement. The execution time for the trace is '0,859 segundos'.

OPERATION	OBJECT_NAME	COST	LAST_CR_BUFFER_GETS
SELECT STATEMENT		24	
HASH (GROUP BY)		24	16
VIEW	index\$_join\$_001	23	16
HASH JOIN			16
Access Predicates			
ROWID=ROWID			
HASH JOIN			10
Access Predicates			
ROWID=ROWID			
BITMAP CONVERSION (TO ROWIDS)		5	5
BITMAP INDEX (FULL SCAN)	BITMAP2		5
BITMAP CONVERSION (TO ROWIDS)		5	5
BITMAP INDEX (FULL SCAN)	BITMAP3		5
BITMAP CONVERSION (TO ROWIDS)		6	6
BITMAP INDEX (FULL SCAN)	BITMAP1		6

# Bottlenecks in Row-Oriented DBs

- *Objective: Identify the bottlenecks remaining in a row-oriented database after tuning it with bitmaps and materialized views*
- *Tasks:*
  1. (5') *Understand the access plan below and identify what operations are taking more computation time*
  2. (5') *Discussion*

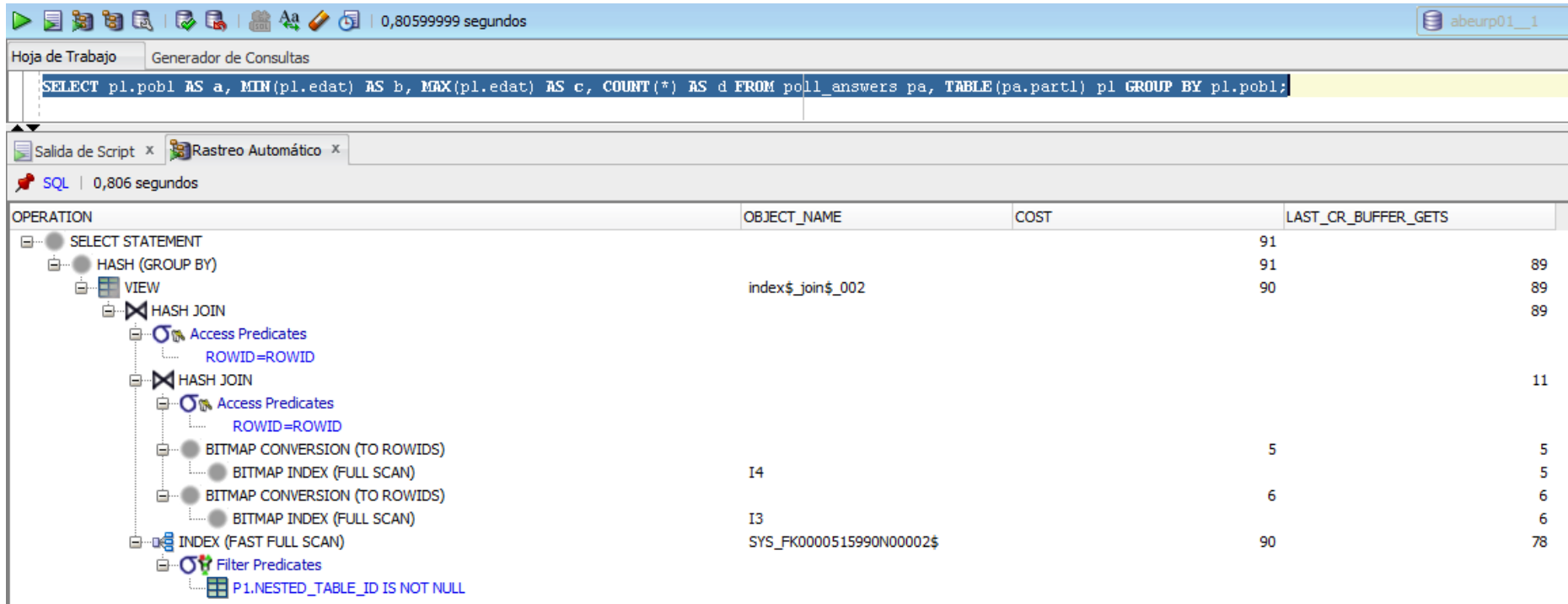


The screenshot shows a database interface with a query editor and an execution plan window. The query is: `SELECT cand, AVG(val) FROM poll_answers GROUP BY cand;`. The execution plan window shows the following operations:

OPERATION	OBJECT_NAME	COST	LAST_CR_BUFFER_GETS
SELECT STATEMENT		5	
MAT_VIEW REWRITE ACCESS (FULL)	V1	5	3

# Bottlenecks in Row-Oriented DBs

- *Objective: Identify the bottlenecks remaining in a row-oriented database after tuning it with vertical fragmentation*
- *Tasks:*
  1. (5') *Understand the access plan below and identify what operations are taking more computation time*
  2. (5') *Discussion*



OPERATION	OBJECT_NAME	COST	LAST_CR_BUFFER_GETS
SELECT STATEMENT		91	
HASH (GROUP BY)		91	89
VIEW	index\$_join\$_002	90	89
HASH JOIN			89
Access Predicates			
ROWID=ROWID			
HASH JOIN			11
Access Predicates			
ROWID=ROWID			
BITMAP CONVERSION (TO ROWIDS)		5	5
BITMAP INDEX (FULL SCAN)	I4		5
BITMAP CONVERSION (TO ROWIDS)		6	6
BITMAP INDEX (FULL SCAN)	I3		6
INDEX (FAST FULL SCAN)	SYS_FK0000515990N00002\$	90	78
Filter Predicates			
P1.NESTED_TABLE_ID IS NOT NULL			



# 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
    - Lightweight 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*