Oracle Database In-Memory

Fast Analytics in Real-Time

24. hroug godišnja konferencija

Andy Rivenes
Database In-Memory Product Management
Oracle Corporation

Email: andy.rivenes@oracle.com
Twitter: @TheInMemoryGuy
Blog: blogs.oracle.com/in-memory

ORACLE

Safe Harbor Statement

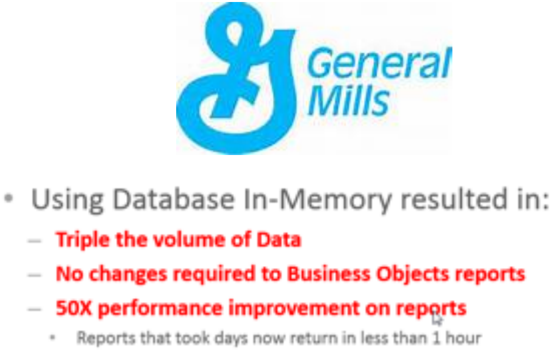
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Why Use Database In-Memory

Improved Reporting Performance

Faster Reports – No Application Changes

- Organizations can use Oracle reporting/analytical applications or existing 3rd party reporting tools
  - No application or data format changes required
- Improves performance (**10x typical**) of reporting applications with existing data warehouse and/or data marts
- Improves performance to ensure SLA’s continue to be maintained
- Increases capacity of mixed workload environments to enable additional growth and performance



- Using Database In-Memory resulted in:
  - **Triple the volume of Data**
  - **No changes required to Business Objects reports**
  - **50X performance improvement on reports**
    - Reports that took days now return in less than 1 hour

Real-Time Analytics

Use Operational Data for Real-Time Analytics

- Real-time analytics on operational data directly -- without the time delay of moving data for reporting
- Enables real-time business intelligence at the point of contact
  - Delivers real-time insight, visibility and agility for critical business operations and decisions
- Enables real-time ad-hoc reporting /analysis and iterative drill-downs on operational data
- No application or data format changes required

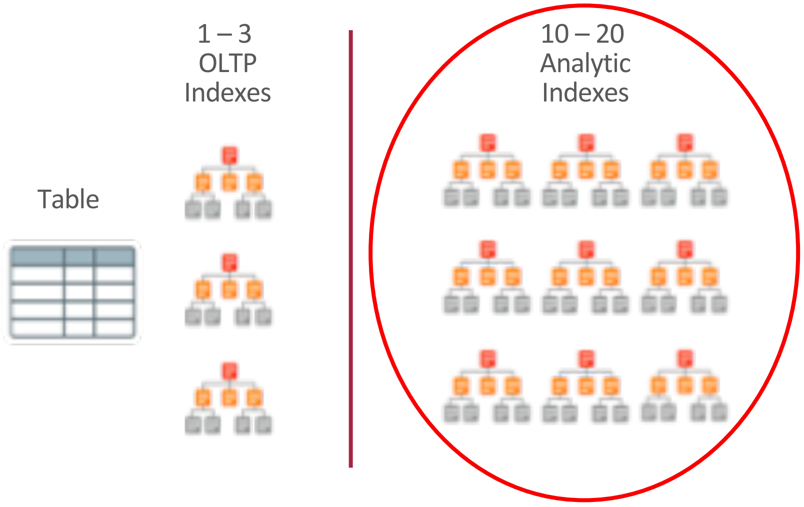
Pricerite實惠

- Using Database In-Memory resulted in:
  - **Analytic queries up to 5x faster**
  - **Real-time analytics dashboard**

Reduced Overhead

Faster Analytics -- Less Storage Overhead

- Analytic indexes can slow down the performance of transactional applications
  - Requires significantly more database storage (on costly tier 1 storage)
  - Increases overhead due to index maintenance
- Database In-Memory allows users to eliminate analytic reporting indexes – without impacting performance
- Removing the need for analytic reporting indexes greatly simplifies tuning and reduces ongoing administration



Walgreens

- Using Database In-Memory resulted in:
  - Performance Gains: 1.8X to 12X
  - Space savings and reduced contention on DML by dropping analytic indexes

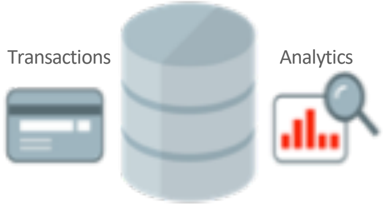
What Is Database In-Memory

Oracle Database In-Memory

Real-Time Analytics

Enable Real-Time Business Decisions

Accelerate Mixed Workload



Run analytics on Operational Systems

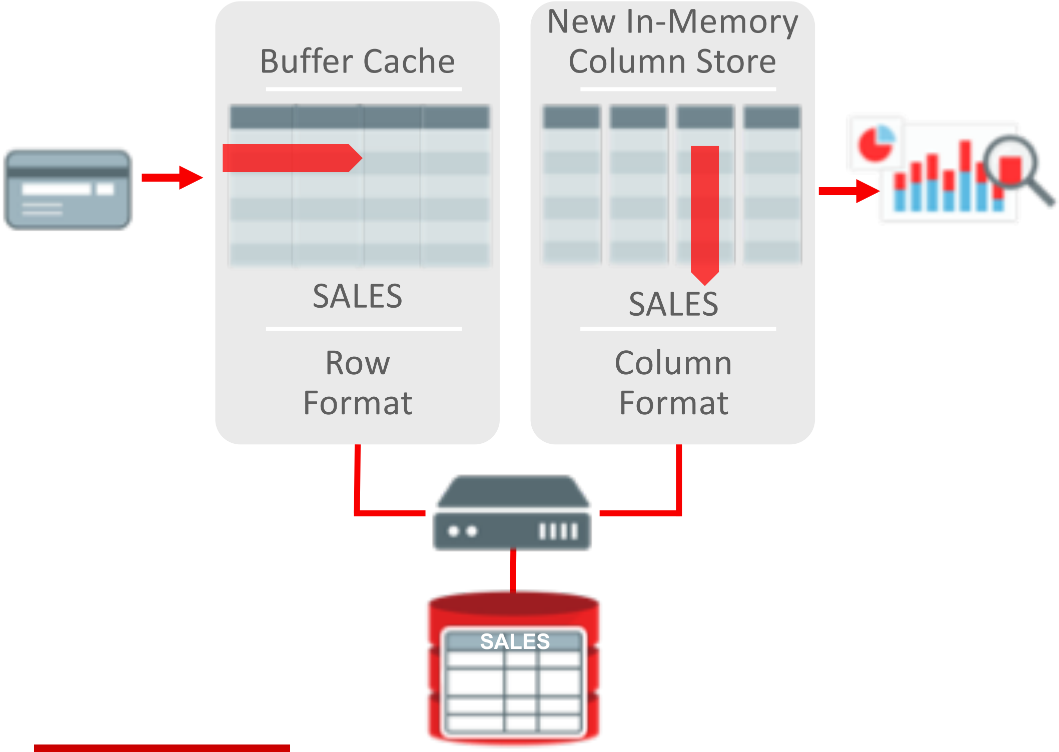
Risk-Free

Proven Scale-Out, Availability, Security

Trivial to Implement

No Application Changes
Not Limited by Memory

Breakthrough: Dual Format Database



- **BOTH** row and column formats for same table
- Simultaneously active and transactionally consistent
- Analytics & reporting use new in-memory Column format
- OLTP uses proven row format

Oracle In-Memory: Simple to Implement

1. Configure Memory Capacity
   - `inmemory_size = XXX GB`

2. Configure tables or partitions to be in memory
   - `alter table | partition … `**`inmemory;`**

3. Later drop analytic indexes to speed up OLTP

Why Columnar Formats?

- Only scan the columns involved in the query
- Columnar formats enable better compression
- Columnar data is vector oriented – takes advantage of SIMD
- Can skip portions of the data if outside value ranges - In-Memory storage indexes
- Oracle Database is capable of scanning billions of rows per second per core
- But don’t forget, it does not accelerate DML – that’s why we have both formats

Why In-Memory?

- Memory is an enabler, allowing the fastest scanning possible
  - Populating columnar data in-memory means not having to wait for I/O
- However, columnar formatted data can be placed on any storage tier:
  - DRAM – In Oracle Database SGA
  - Flash – In Exadata flash cache
  - On-disk – Engineered systems Hybrid Columnar Compression (HCC)

Where Is It Available

Database In-Memory

- Database In-Memory is an option for Oracle Database Enterprise Edition
- Database In-Memory was included in the first patchset (12.1.0.2) for 12.1 and all subsequent Oracle Database releases
- Available:
  - Database Cloud Service – Virtual Machines: **Extreme Performance**
  - Database Cloud Service – Bare Metal: **Extreme Performance**
  - Exadata Cloud Service
  - Exadata Cloud at Customer
  - Autonomous Data Warehouse (Flash only)
  - On-premises

Note: Database In-Memory is **not** enabled by default

Database In-Memory in the Oracle Public Cloud

Easiest Platform to Try or Deploy In-Memory

Database Cloud Service – Virtual Machines

- Enterprise Extreme Performance
- Up to 48 OCPUs
- Up to 640 GB RAM per VM

Database Cloud Service – Bare Metal

- Enterprise Extreme Performance
- Up to 52 OCPUs
- Up to 768 GB RAM per Database instance

Exadata Cloud Service

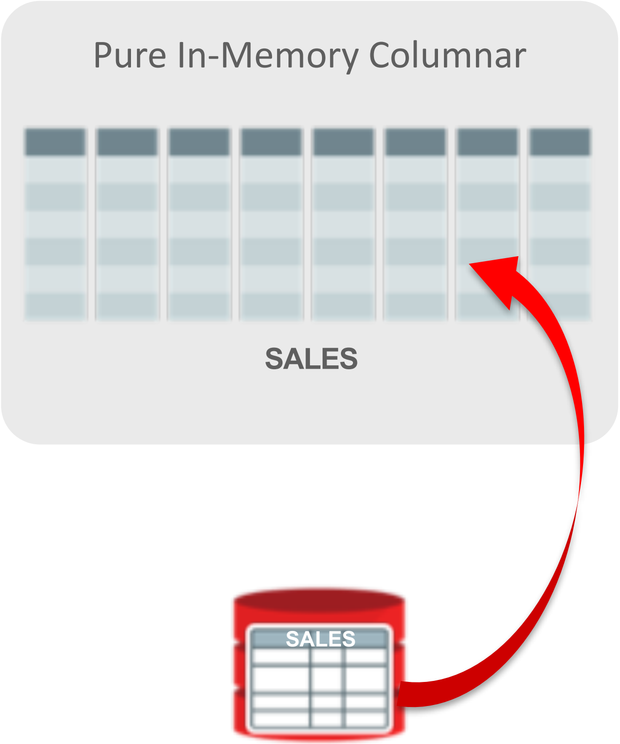
- Up to 368 Cores
- Up to 5.7 TB RAM
- Over 300 TB of Flash Cache Available

Exadata Cloud at Customer

- Up to 368 Cores
- Up to 5.7 TB RAM
- Over 300 TB of Flash Cache Available

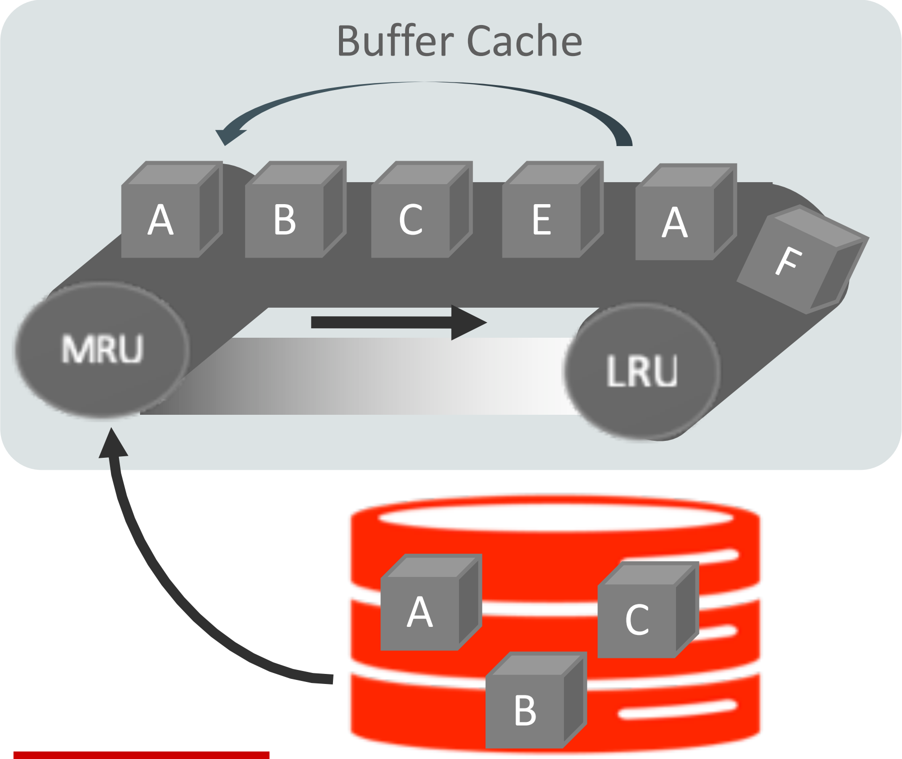
How Does Database In-Memory Work

Oracle In-Memory Columnar Technology



- Pure in-memory columnar format
  - Not persistent, so no undo or redo is generated
- Can be enabled for table, partition, subpartition or materialized view
- 2x to 20x compression typical
- Available on all hardware platforms

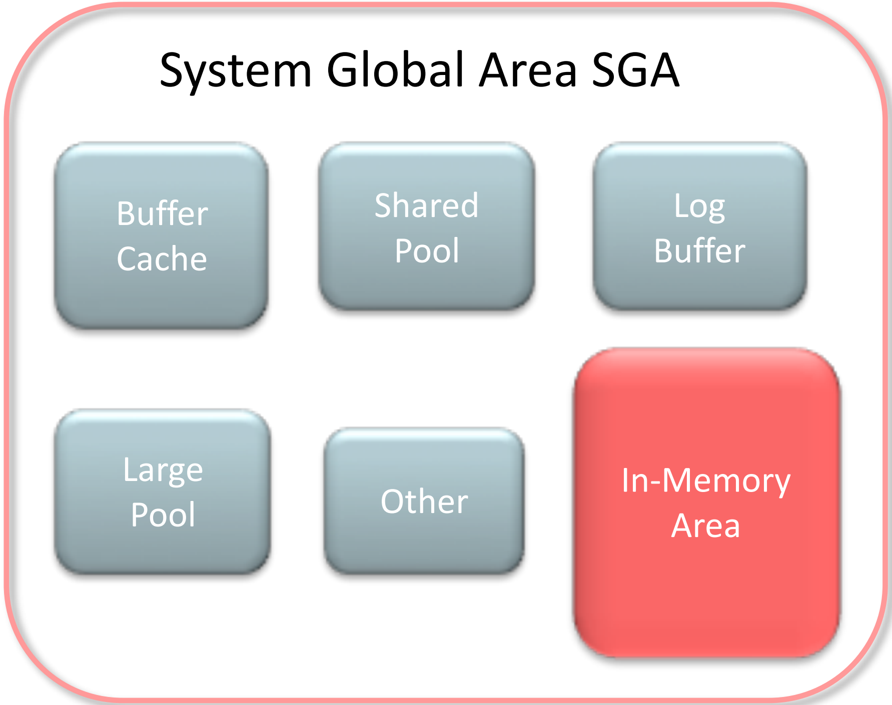
In-Memory A Store – Not A Cache



- What is a cache?
- A pool of memory
- Data automatically brought into memory based on access
- Data automatically aged out
- Good example:

  Oracle Database Buffer Cache

In-Memory Area: New Static Area within SGA



- Contains data in the new In-Memory Column Format
- Controlled by INMEMORY_SIZE parameter
  - Minimum size of 100MB
- Can dynamically grow larger (12.2)
- SGA_TARGET must be large enough to accommodate this area

Note: Don’t steal Memory from other components

Population

- **Order in which objects are populated** controlled by PRIORITY subclause:

```
ALTER TABLE sales
INMEMORY PRIORITY HIGH;
```

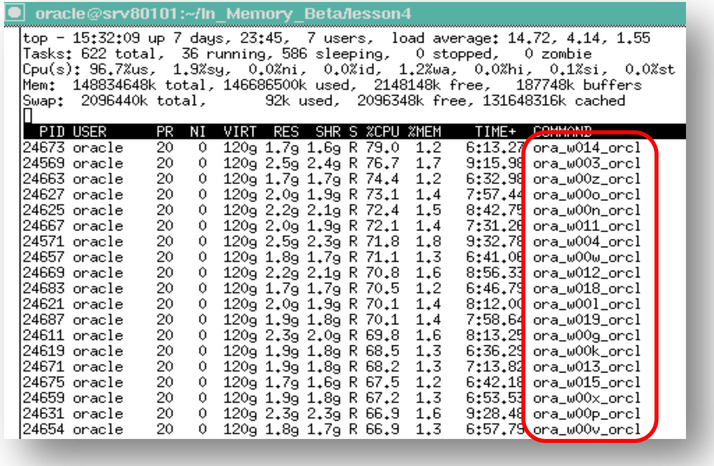
- Levels:
  - CRITICAL > HIGH > MEDIUM > LOW
  - Controls order (not speed) of populate
- Default PRIORITY is NONE
  - Populate only on first access

- Population completed by background processes

  ora_w001_orcl

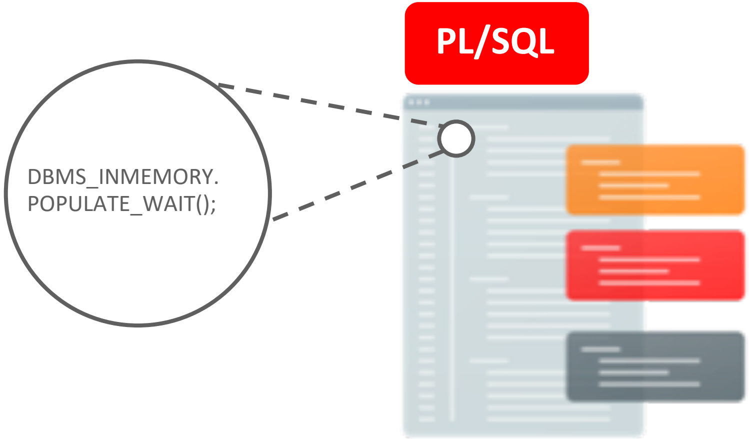
- Number of processes controlled by parameter:

  INMEMORY_MAX_POPULATE_SERVERS



```
top - 15:32:09 up 7 days, 23:45,  7 users,  load average: 14.72, 4.14, 1.55
Tasks: 622 total,  36 running, 586 sleeping,   0 stopped,   0 zombie
Cpu(s): 96.7%us,  1.9%sy,  0.0%ni,  0.0%id,  1.2%wa,  0.0%hi,  0.1%si,  0.0%st
Mem:  148834648k total, 146686500k used,  2148148k free,   187748k buffers
Swap:  2096440k total,       92k used,  2096348k free, 131648316k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
24673 oracle    20   0  120g 1.7g 1.6g R 79.0  1.2   6:13.27 ora_w014_orcl
24569 oracle    20   0  120g 2.5g 2.4g R 76.7  1.7   9:15.98 ora_w003_orcl
24663 oracle    20   0  120g 1.7g 1.7g R 74.4  1.2   6:32.98 ora_w00z_orcl
24627 oracle    20   0  120g 2.0g 1.9g R 73.1  1.4   7:57.44 ora_w00o_orcl
24625 oracle    20   0  120g 2.2g 2.1g R 72.4  1.5   8:42.75 ora_w00n_orcl
24667 oracle    20   0  120g 2.0g 1.9g R 72.1  1.4   7:31.26 ora_w011_orcl
24571 oracle    20   0  120g 2.5g 2.3g R 71.8  1.8   9:32.78 ora_w004_orcl
24657 oracle    20   0  120g 1.8g 1.7g R 71.1  1.3   6:41.06 ora_w00w_orcl
24669 oracle    20   0  120g 2.2g 2.1g R 70.8  1.6   8:56.33 ora_w012_orcl
24683 oracle    20   0  120g 1.7g 1.7g R 70.5  1.2   6:46.79 ora_w018_orcl
24621 oracle    20   0  120g 2.0g 1.9g R 70.1  1.4   8:12.00 ora_w00l_orcl
24687 oracle    20   0  120g 1.9g 1.8g R 70.1  1.4   7:58.64 ora_w019_orcl
24611 oracle    20   0  120g 2.3g 2.0g R 69.8  1.6   8:13.25 ora_w00g_orcl
24619 oracle    20   0  120g 1.9g 1.8g R 68.5  1.3   6:36.29 ora_w00k_orcl
24671 oracle    20   0  120g 1.9g 1.8g R 68.2  1.3   7:13.82 ora_w013_orcl
24675 oracle    20   0  120g 1.7g 1.6g R 67.5  1.2   6:42.18 ora_w015_orcl
24659 oracle    20   0  120g 1.9g 1.8g R 67.2  1.3   6:53.53 ora_w00x_orcl
24631 oracle    20   0  120g 2.3g 2.3g R 66.9  1.6   9:28.48 ora_w00p_orcl
24654 oracle    20   0  120g 1.8g 1.7g R 66.9  1.3   6:57.79 ora_w00v_orcl
```

Database In-Memory Wait on Populate



- New in 19c - POPULATE_WAIT function in DBMS_INMEMORY package
- Based on population priority setting
- Provides an application API to ensure that objects are populated before being accessed
  - Can be used to ensure application SLAs are met

Database In-Memory Technology

Scanning and filtering data more efficiently

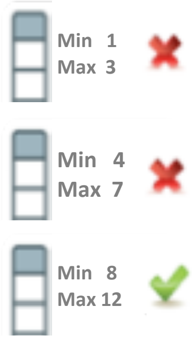
Columnar Format

Access only the columns you need

Compression

Scan & filter data in compressed format

Storage Indexes



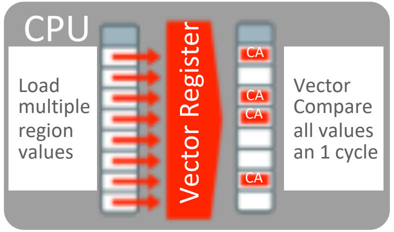
Min 1 Max 3 ✗

Min 4 Max 7 ✗

Min 8 Max 12 ✓

Prune out any unnecessary data from the column

SIMD Vector Processing



CPU

Load multiple region values

Vector Register

CA CA CA CA

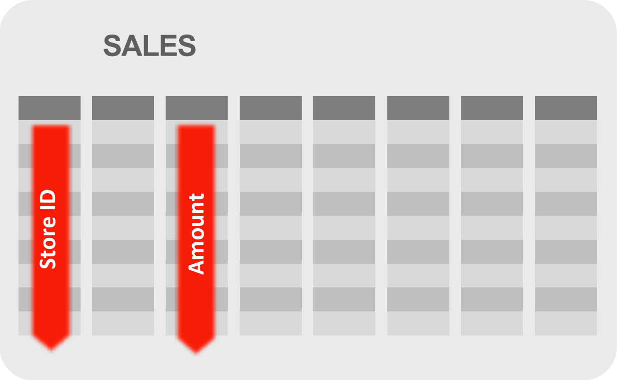
Vector Compare all values an 1 cycle

Process multiple column values in a single CPU instruction

Technology: Columnar Format

Access Only The Columns In The Query

- Scan only the needed columns
  - No need to read each “row” and traverse each column to find values
- All columns accessed for the table(s) in the query must be populated
  - If excluded columns are accessed the query will run against the row-store

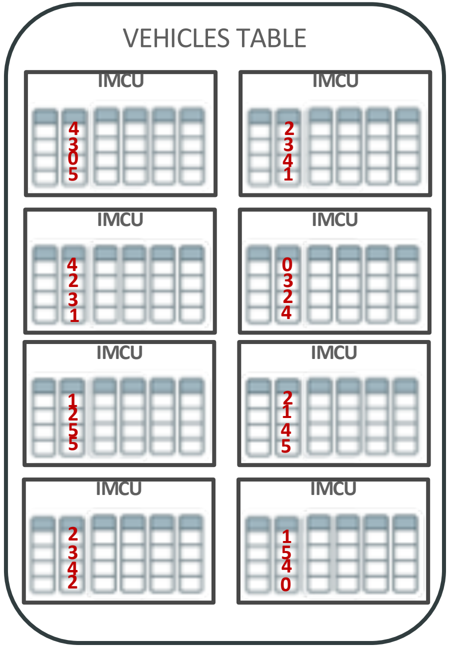


Technology: Compression

- Multiple levels of compression
  - FOR DML
  - FOR QUERY LOW/HIGH
  - FOR CAPACITY LOW/HIGH
- Query Low and High use dictionary encoding and run length encoding – evaluated directly against compressed data
- Capacity Low and High add additional “zip-like” compression

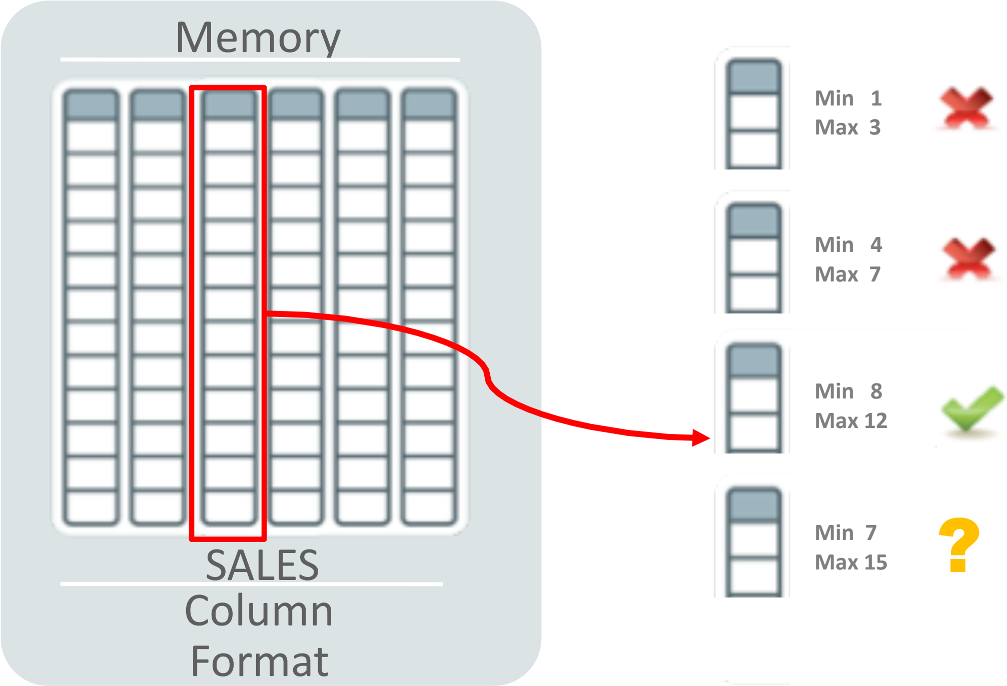
Common Dictionary

NAME	ID
AUDI	0
BMW	1
CADILLAC	2
PORSCHE	3
TESLA	4
VW	5



Technology: In-Memory Storage Indexes

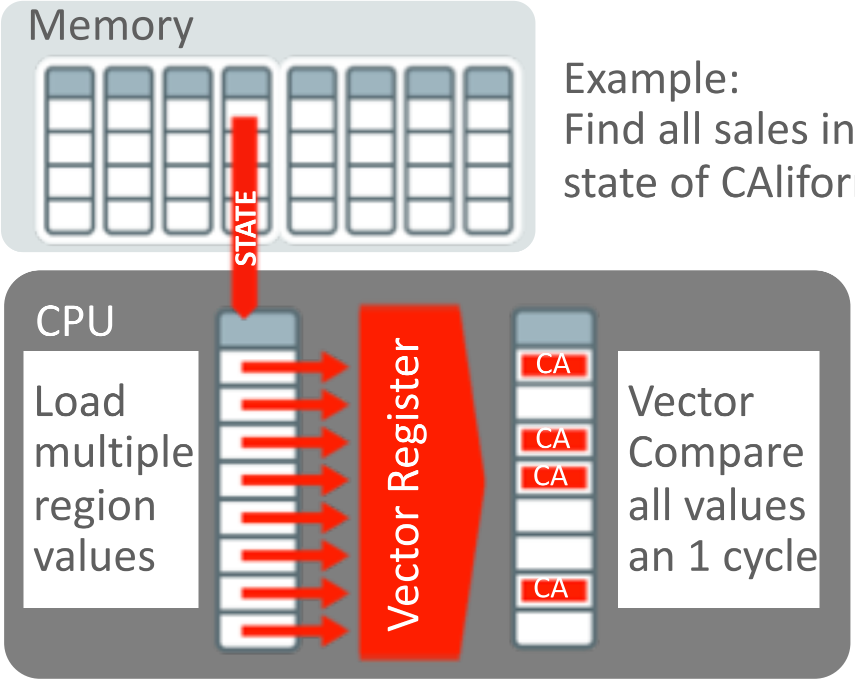
Only look at the data you need!



- **Example:** Find all sales from stores with a store_id of 8
  - Each column is the made up of multiple column units
  - Min / max value is recorded for each column unit in a storage index
  - Storage index provides partition pruning like performance for ALL queries

Technology: SIMD Vector Processing

Orders of Magnitude Faster Analytic Data Scans


Memory
STATE
CPU
Load multiple region values
Vector Register
CA
CA
CA
CA
Vector Compare all values an 1 cycle

Example:
Find all sales in
state of CAlifornia

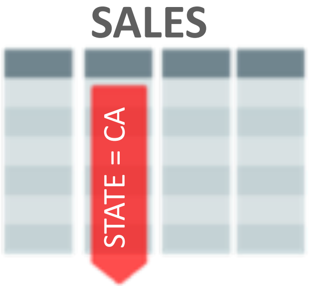
> 100x Faster

- Each CPU core scans local in-memory columns
- SIMD vector instructions used to process multiple values in each instruction
  - Originally designed for graphics & science
- **Billions of rows/sec** scan rate per CPU core
  - Row format is millions/sec

Optimizer Enhancements

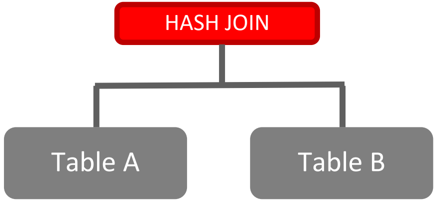
Improves all aspects of analytic queries

Data Scans



- Speed of memory
- Scan and Filter only the needed Columns
- Vector Instructions

Joins



- Convert Star Joins into 10X Faster Column Scans
- Search large table for values that match small table

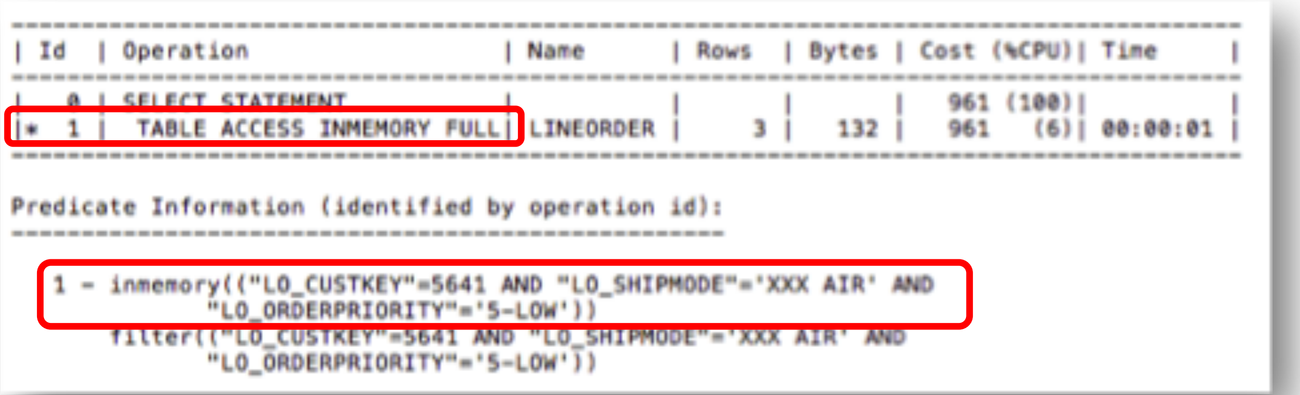
In-Memory Aggregation

- Create In-Memory Report Outline that is Populated during Fast Scan
- Runs Reports Instantly

Optimizer: Data Scans

Pushing down filter predicates

- Many types of aggregations and filter predicates can be more efficiently evaluated **during** the In-Memory scan rather than **after**
  - Evaluate predicates directly against compressed columnar data
  - Use SIMD to evaluate predicates on multiple column values concurrently

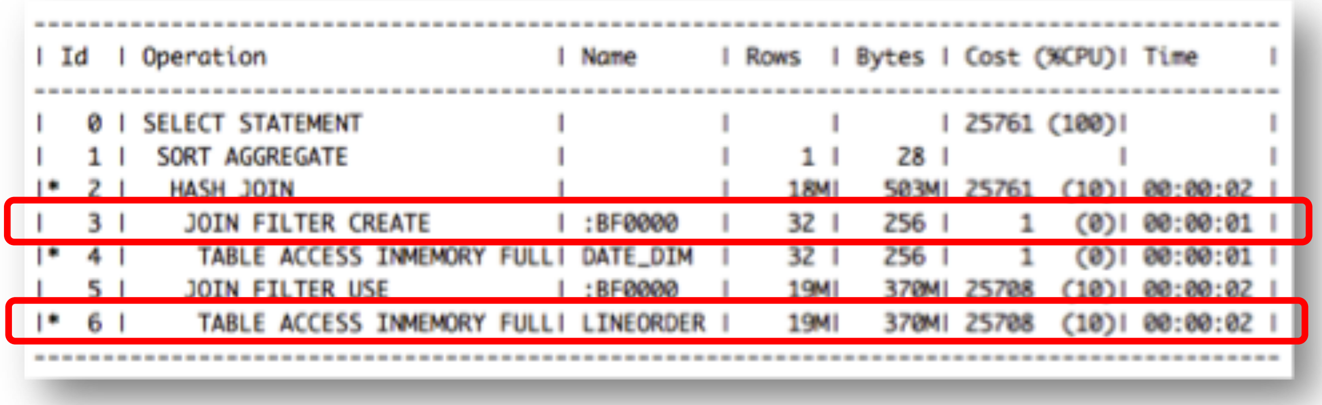


```
-----------------------------------------------------------------------------------------
| Id  | Operation                  | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |           |       |       |   961 (100)|          |
|*  1 |  TABLE ACCESS INMEMORY FULL| LINEORDER |     3 |   132 |   961   (6)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - inmemory(("LO_CUSTKEY"=5641 AND "LO_SHIPMODE"='XXX AIR' AND
              "LO_ORDERPRIORITY"='5-LOW'))
       filter(("LO_CUSTKEY"=5641 AND "LO_SHIPMODE"='XXX AIR' AND
              "LO_ORDERPRIORITY"='5-LOW'))
```

Optimizer: Hash Joins

In-Memory Execution Plan with Bloom Filter

- Bloom filters enable joins to be converted into fast column scans
- Can see the Bloom filter create and use – No guessing
- Same technique used to offload joins on Exadata

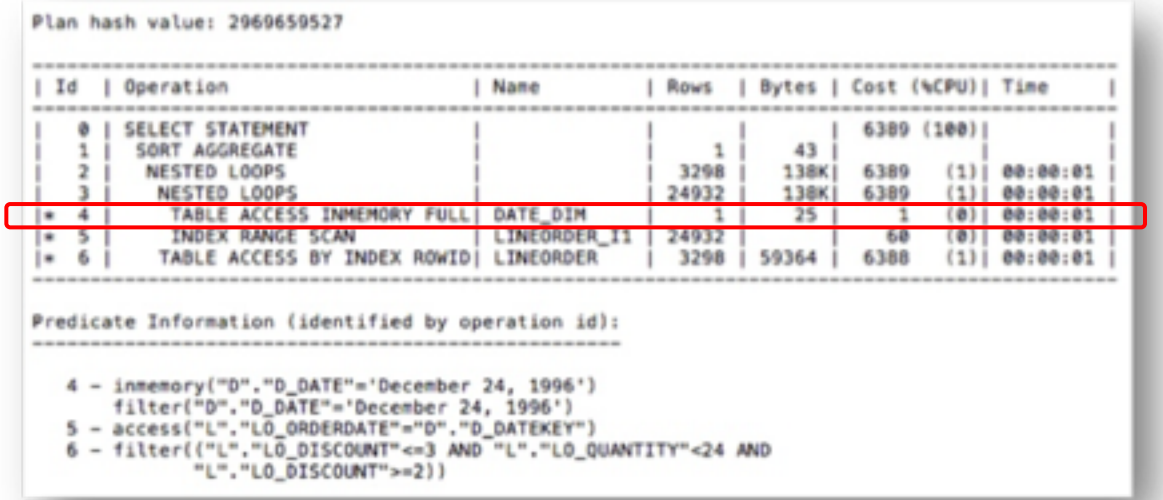


```
---------------------------------------------------------------------------------------------
| Id  | Operation                       | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                |           |       |       | 25761 (100)|          |
|   1 |  SORT AGGREGATE                 |           |     1 |    28 |            |          |
|*  2 |   HASH JOIN                     |           |   18M|   503M| 25761  (10)| 00:00:02 |
|   3 |    JOIN FILTER CREATE           | :BF0000   |    32 |   256 |     1   (0)| 00:00:01 |
|*  4 |     TABLE ACCESS INMEMORY FULL| DATE_DIM  |    32 |   256 |     1   (0)| 00:00:01 |
|   5 |    JOIN FILTER USE              | :BF0000   |   19M|   370M| 25708  (10)| 00:00:02 |
|*  6 |     TABLE ACCESS INMEMORY FULL| LINEORDER |   19M|   370M| 25708  (10)| 00:00:02 |
---------------------------------------------------------------------------------------------
```

Optimizer: Nested Loops Joins

In-Memory Execution Plan with Nested Loops Join

- Database In-Memory can work **with** indexes (but doesn't use)
- Optimizer makes a cost based decision

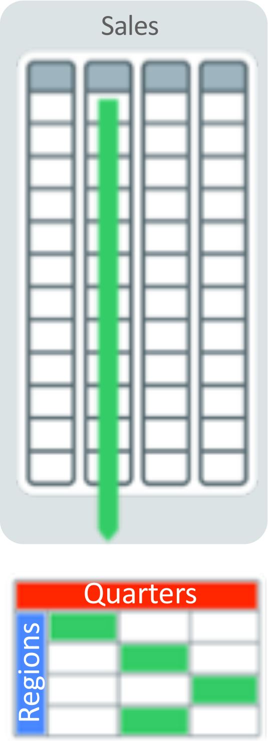


```
Plan hash value: 2969659527

-------------------------------------------------------------------------------------------
| Id  | Operation                      | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |              |       |       |  6389 (100)|          |
|   1 |  SORT AGGREGATE                |              |     1 |    43 |            |          |
|   2 |   NESTED LOOPS                 |              |  3298 |  138K|  6389   (1)| 00:00:01 |
|   3 |    NESTED LOOPS                |              | 24932 |  138K|  6389   (1)| 00:00:01 |
|*  4 |     TABLE ACCESS INMEMORY FULL| DATE_DIM     |     1 |    25 |     1   (0)| 00:00:01 |
|*  5 |     INDEX RANGE SCAN           | LINEORDER_I1 | 24932 |       |    60   (0)| 00:00:01 |
|*  6 |    TABLE ACCESS BY INDEX ROWID| LINEORDER    |  3298 | 59364 |  6388   (1)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - inmemory("D"."D_DATE"='December 24, 1996')
       filter("D"."D_DATE"='December 24, 1996')
   5 - access("L"."LO_ORDERDATE"="D"."D_DATEKEY")
   6 - filter(("L"."LO_DISCOUNT"<=3 AND "L"."LO_QUANTITY"<24 AND
              "L"."LO_DISCOUNT">=2))
```

Optimizer: In-Memory Aggregation

Key Vector Use & Vector Group By

```
----------------------------------------------------------------------------
| Id  | Operation                                   | Name                        |
----------------------------------------------------------------------------
|   0 | SELECT STATEMENT                            |                             |
|   1 |  TEMP TABLE TRANSFORMATION                  |                             |
|   2 |   LOAD AS SELECT(CURSOR DURATION MEMORY)| SYS_TEMP_0FD9DADAD_9873DD |
|   3 |    VECTOR GROUP BY                          |                             |
|   4 |     KEY VECTOR CREATE BUFFERED              | :KV0000                     |
|   5 |      PARTITION RANGE ALL                    |                             |
|   6 |       TABLE ACCESS INMEMORY FULL            | TIME_DIM                    |
|   7 |   LOAD AS SELECT(CURSOR DURATION MEMORY)| SYS_TEMP_0FD9DADAE_9873DD |
|   8 |    VECTOR GROUP BY                          |                             |
|   9 |     KEY VECTOR CREATE BUFFERED              | :KV0001                     |
|  10 |      TABLE ACCESS INMEMORY FULL             | CUSTOMER_DIM                |
|  11 |   HASH GROUP BY                             |                             |
|  12 |    HASH JOIN                                |                             |
|  13 |     HASH JOIN                               |                             |
|  14 |      TABLE ACCESS FULL                      | SYS_TEMP_0FD9DADAE_9873DD |
|  15 |      VIEW                                   | VW_VT_AF278325              |
|  16 |       VECTOR GROUP BY                       |                             |
|  17 |        HASH GROUP BY                        |                             |
|  18 |         KEY VECTOR USE                      | :KV0001                     |
|  19 |          KEY VECTOR USE                     | :KV0000                     |
|  20 |           PARTITION RANGE SUBQUERY          |                             |
|  21 |            TABLE ACCESS INMEMORY FULL       | SALES_FACT                  |
|  22 |     TABLE ACCESS FULL                       | SYS_TEMP_0FD9DADAD_9873DD |
----------------------------------------------------------------------------
```

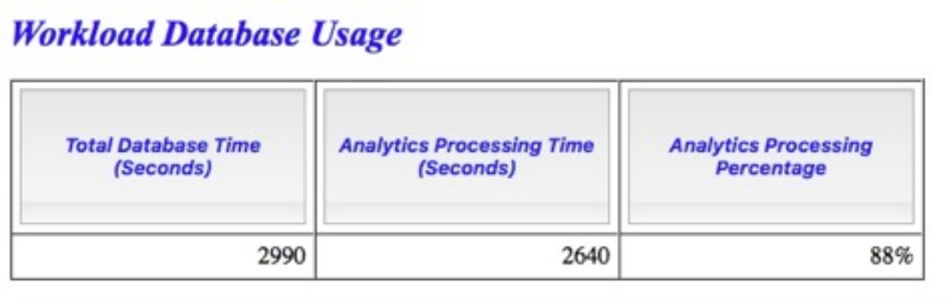


Scan, filter and aggregate

How Much Memory Do You Need

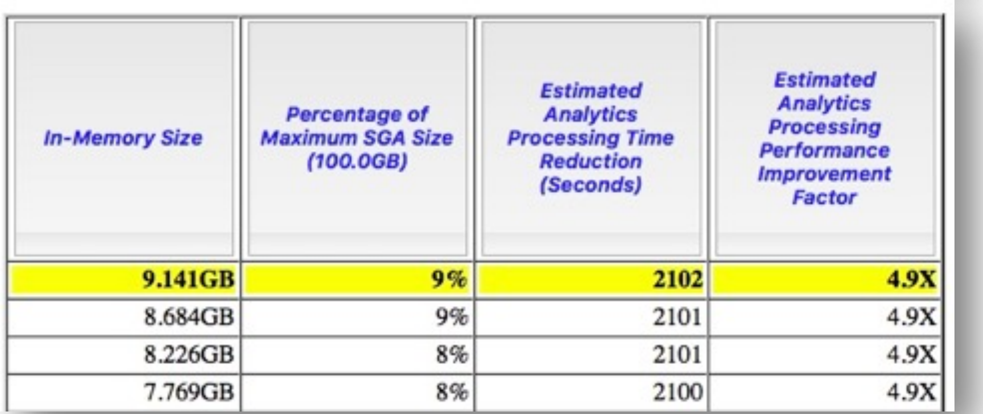
ORACLE

Oracle In-Memory Advisor



Workload Database Usage

Total Database Time (Seconds)	Analytics Processing Time (Seconds)	Analytics Processing Percentage
2990	2640	88%

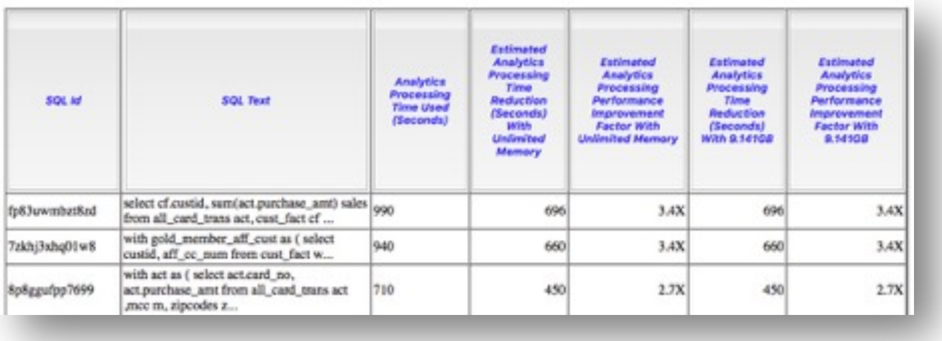


In-Memory Size	Percentage of Maximum SGA Size (100.0GB)	Estimated Analytics Processing Time Reduction (Seconds)	Estimated Analytics Processing Performance Improvement Factor
9.141GB	9%	2102	4.9X
8.684GB	9%	2101	4.9X
8.226GB	8%	2101	4.9X
7.769GB	8%	2100	4.9X

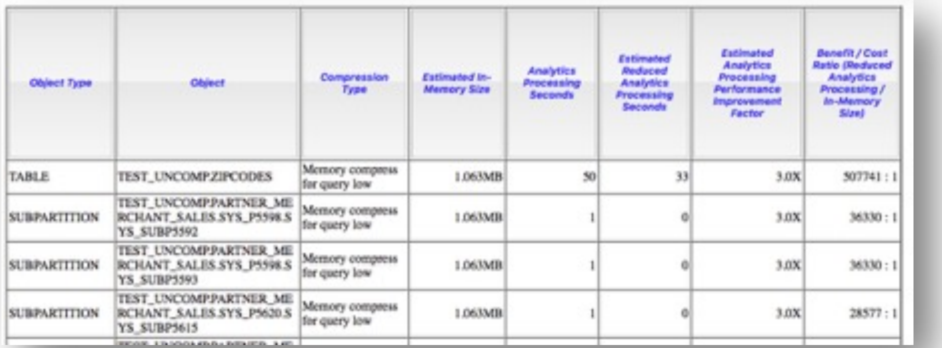
- In-Memory Advisor – free download available on OTN for 11.2.0.3+ DBs
- Analyzes existing DB workload via AWR & ASH repositories
- Provides list of objects that would benefit most from being populated into IM column

Note: Database Tuning Pack license required

Oracle In-Memory Advisor



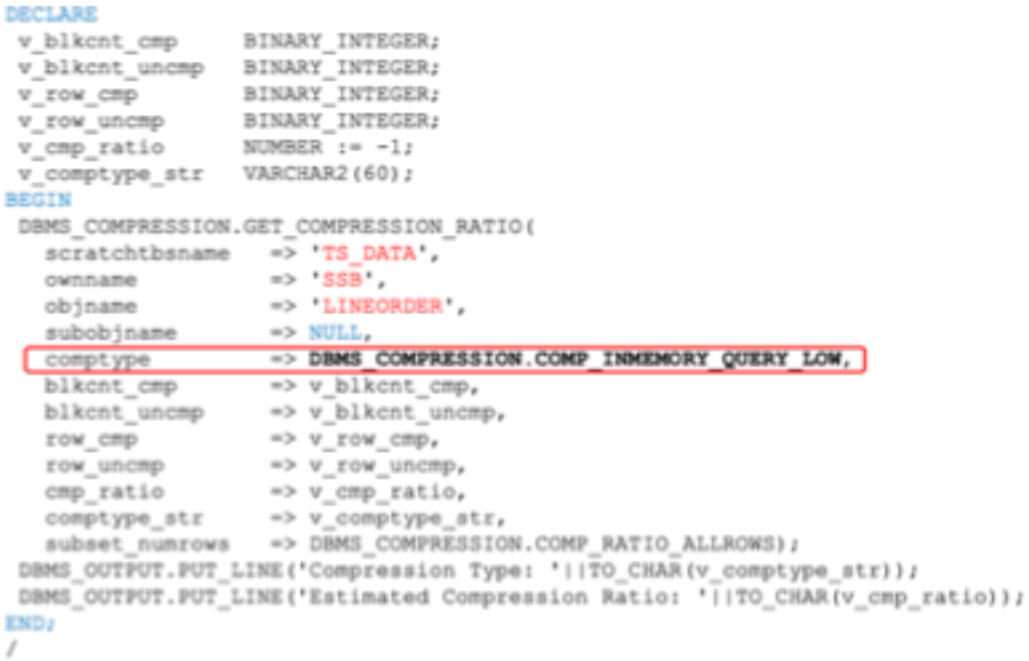
SQL Id	SQL Text	Analytics Processing Time Used (Seconds)	Estimated Analytics Processing Time Reduction (Seconds) With Unlimited Memory	Estimated Analytics Processing Performance Improvement Factor With Unlimited Memory	Estimated Analytics Processing Time Reduction (Seconds) With 9.141GB	Estimated Analytics Processing Performance Improvement Factor With 9.141GB
fp83uwmbzt8zd	select cf.custid, sum(act.purchase_amt) sales from all_card_trans act, cust_fact cf ...	990	696	3.4X	696	3.4X
7zkhj3xhq01w8	with gold_member_aff_cust as (select custid, aff_cc_num from cust_fact w...	940	660	3.4X	660	3.4X
8p8ggufpp7699	with act as (select act.card_no, act.purchase_amt from all_card_trans act ,mcc m, zipcodes z...	710	450	2.7X	450	2.7X



Object Type	Object	Compression Type	Estimated In-Memory Size	Analytics Processing Seconds	Estimated Reduced Analytics Processing Seconds	Estimated Analytics Processing Performance Improvement Factor	Benefit / Cost Ratio (Reduced Analytics Processing / In-Memory Size)
TABLE	TEST_UNCOMP.ZIPCODES	Memory compress for query low	1.063MB	50	33	3.0X	507741 : 1
SUBPARTITION	TEST_UNCOMP.PARTNER_MERCHANT_SALES.SYS_P5598.SYS_SUBP5592	Memory compress for query low	1.063MB	1	0	3.0X	36330 : 1
SUBPARTITION	TEST_UNCOMP.PARTNER_MERCHANT_SALES.SYS_P5598.SYS_SUBP5593	Memory compress for query low	1.063MB	1	0	3.0X	36330 : 1
SUBPARTITION	TEST_UNCOMP.PARTNER_MERCHANT_SALES.SYS_P5620.SYS_SUBP5615	Memory compress for query low	1.063MB	1	0	3.0X	28577 : 1

- Multiple sections available
  - In-Memory Size
  - SQL Statements with Analytic Benefit
  - Top object recommendations
  - All object based on memory size
  - Recommendation Rationale
  - Implementation SQL

Oracle Compression Advisor And In-Memory



```
DECLARE
 v_blkcnt_cmp      BINARY_INTEGER;
 v_blkcnt_uncmp    BINARY_INTEGER;
 v_row_cmp         BINARY_INTEGER;
 v_row_uncmp       BINARY_INTEGER;
 v_cmp_ratio       NUMBER := -1;
 v_comptype_str    VARCHAR2(60);
BEGIN
 DBMS_COMPRESSION.GET_COMPRESSION_RATIO(
   scratchtbsname    => 'TS_DATA',
   ownname           => 'SSB',
   objname           => 'LINEORDER',
   subobjname        => NULL,
   comptype          => DBMS_COMPRESSION.COMP_INMEMORY_QUERY_LOW,
   blkcnt_cmp        => v_blkcnt_cmp,
   blkcnt_uncmp      => v_blkcnt_uncmp,
   row_cmp           => v_row_cmp,
   row_uncmp         => v_row_uncmp,
   cmp_ratio         => v_cmp_ratio,
   comptype_str      => v_comptype_str,
   subset_numrows    => DBMS_COMPRESSION.COMP_RATIO_ALLROWS);
 DBMS_OUTPUT.PUT_LINE('Compression Type: '||TO_CHAR(v_comptype_str));
 DBMS_OUTPUT.PUT_LINE('Estimated Compression Ratio: '||TO_CHAR(v_cmp_ratio));
END;
/
```

- Easy way to determine memory requirements
- Use DBMS_COMPRESSION
- Applies MEMCOMPRESS to sample set of data from a table
- Returns estimated compression ratio

What If You Don’t Have Enough Memory

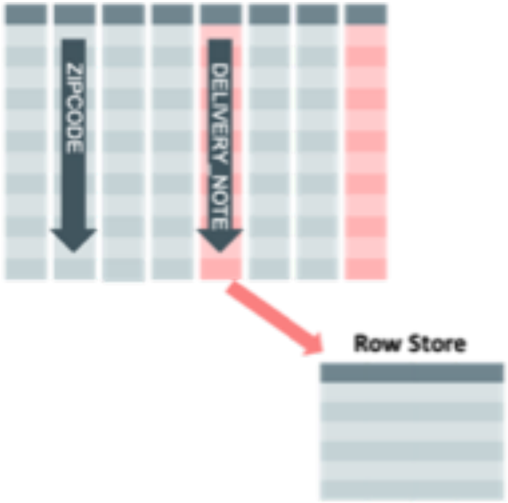
Compression

```
ALTER MATERIALIZED VIEW mv1
INMEMORY
MEMCOMPRESS FOR QUERY LOW;


CREATE TABLE trades
   (Name varchar(20),
    Desc varchar(200))
INMEMORY
MEMCOMPRESS FOR DML(desc);
```

- Objects compressed during population
- New compression techniques
  - Focused on scan performance
- 2x to 20x compression typical
- Multiple levels of compression
  - FOR DML
  - FOR QUERY LOW/HIGH
  - FOR CAPACITY LOW/HIGH
- Possible to use a different level for different partitions in a table

Columns Can Be Excluded

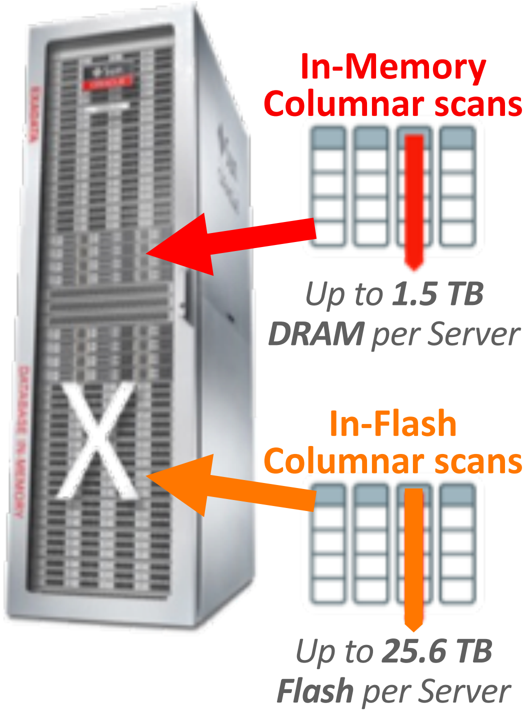
```
ALTER TABLE sales INMEMORY
NO INMEMORY (delivery_note);
```



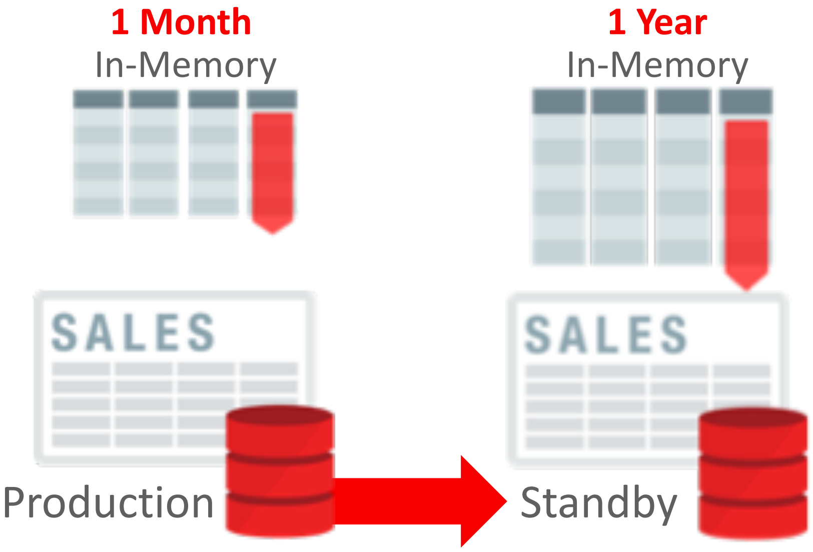
- You don't have to populate all columns
  - But, if excluded columns are accessed then the query will run against the row-store
- Two phase approach
  1. INMEMORY attribute on table automatically inherited by columns
  2. Need to remove attribute from the columns you don’t want populated

Extend In-Memory Analytics into Storage

- Exadata automatically transforms table data into In-Memory DB columnar formats in Exadata Flash Cache
  - Enables fast vector processing for storage server queries
- Additional compression for OLTP compressed or uncompressed tables in flash – new in Exadata System Software 18.1
- Enables dictionary lookup and avoids processing unnecessary rows
- Smart Scan results sent back to database in In-Memory Columnar format
  - Reduces Database node CPU utilization
- Uniquely optimizes next generation Flash as memory


In-Memory Columnar scans
Up to **1.5 TB** *DRAM* per Server
In-Flash Columnar scans
Up to **25.6 TB** *Flash* per Server
X

Mixed Workload: In-Memory on Active Data Guard

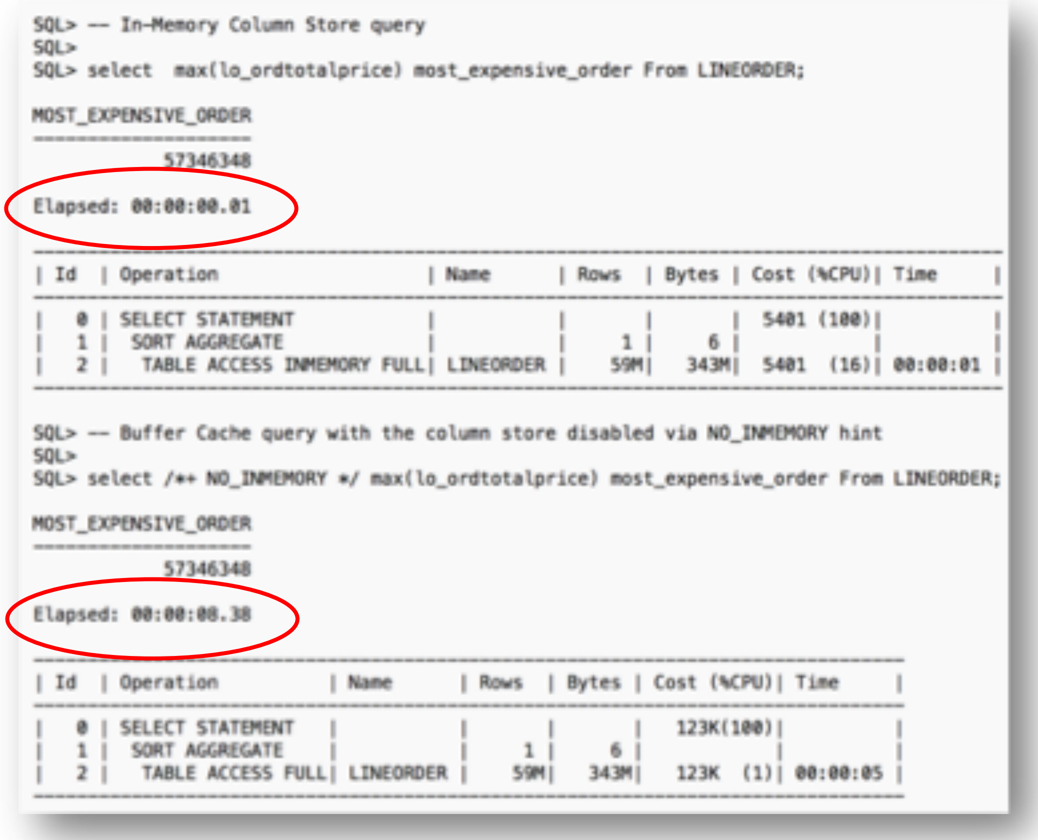


- Real-time analytics with no impact on primary database
- Makes full use of memory on standby system
- Standby can populate different data than production database
- Available on Exadata and PaaS Cloud Services

Why Not Just Cache The Table In The Buffer Cache

ORACLE

Compare Column-store to Row-store



```
SQL> -- In-Memory Column Store query
SQL>
SQL> select  max(lo_ordtotalprice) most_expensive_order From LINEORDER;

MOST_EXPENSIVE_ORDER
--------------------
            57346348

Elapsed: 00:00:00.01

-------------------------------------------------------------------------------------------
| Id  | Operation                  | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |           |       |       |  5401 (100)|          |
|   1 |  SORT AGGREGATE            |           |     1 |     6 |            |          |
|   2 |   TABLE ACCESS INMEMORY FULL| LINEORDER |   59M|   343M|  5401  (16)| 00:00:01 |
-------------------------------------------------------------------------------------------

SQL> -- Buffer Cache query with the column store disabled via NO_INMEMORY hint
SQL>
SQL> select /*+ NO_INMEMORY */ max(lo_ordtotalprice) most_expensive_order From LINEORDER;

MOST_EXPENSIVE_ORDER
--------------------
            57346348

Elapsed: 00:00:08.38

----------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
----------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |       |       |  123K(100)|          |
|   1 |  SORT AGGREGATE    |           |     1 |     6 |            |          |
|   2 |   TABLE ACCESS FULL| LINEORDER |   59M|   343M|  123K  (1)| 00:00:05 |
----------------------------------------------------------------------------------
```

ORACLE

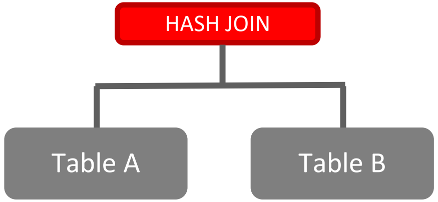
How Do You Tell If It’s Working

Which Queries Benefit From Database In-Memory?

For a non-trivial amount of rows and execution time, when a significant amount of time …

is spent accessing data

is spent joining data

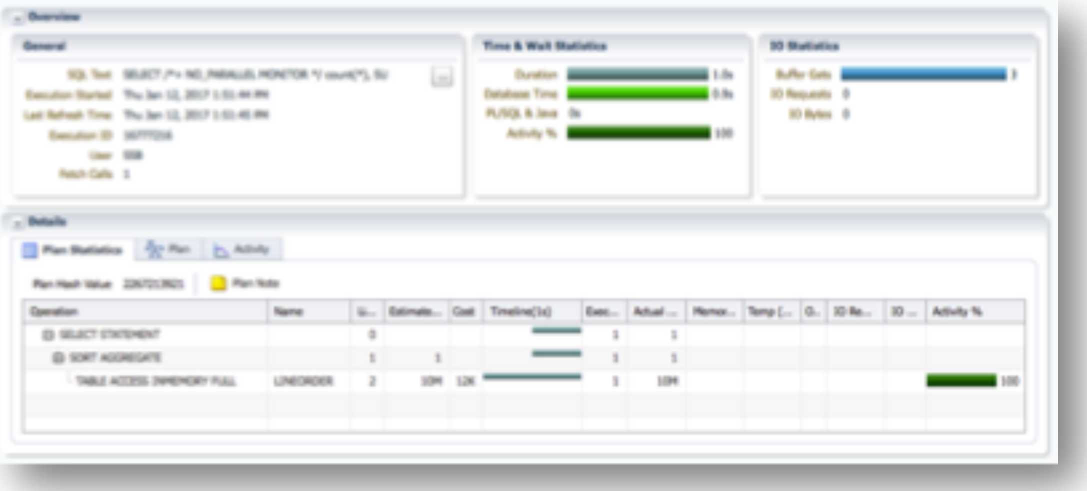


is spent aggregating data

Use Time Based Analysis Techniques To Evaluate Benefit

SQL Monitor Active Reports

- Shows how SQL was executed and where **time was spent**
- See blogs.oracle.com/In-Memory for a technical brief on creating SQL Monitor active reports



SQL Text SELECT /*+ NO_PARALLEL MONITOR */ count(*), SU
Execution Started Thu Jan 12, 2017 1:51:44 PM
Last Refresh Time Thu Jan 12, 2017 1:51:45 PM
Execution ID 16777216
User SSB
Fetch Calls 1

Duration 1.0s
Database Time 0.9s
PL/SQL & Java 0s
Activity % 100

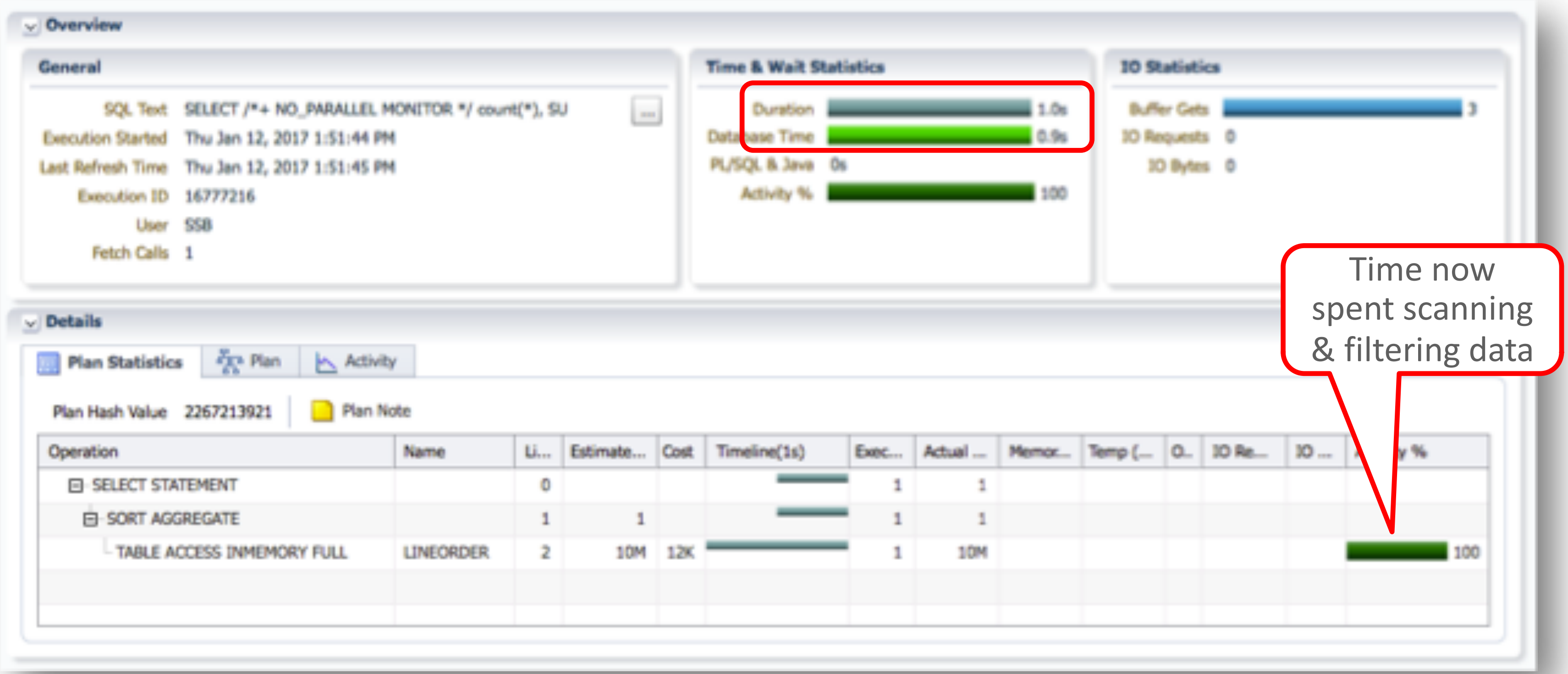
Buffer Gets 3
IO Requests 0
IO Bytes 0

Plan Hash Value 2267213921

Operation	Name	Li...	Estimate...	Cost	Timeline(1s)	Exec...	Actual ...	Memor...	Temp (...	O..	IO Re...	IO ...	Activity %
SELECT STATEMENT		0				1	1						
SORT AGGREGATE		1	1			1	1						
TABLE ACCESS INMEMORY FULL	LINEORDER	2	10M	12K		1	10M						100

ORACLE

Accessing Data: Scan & filter data in-memory



Overview

General

SQL Text	SELECT /*+ NO_PARALLEL MONITOR */ count(*), SU
Execution Started	Thu Jan 12, 2017 1:51:44 PM
Last Refresh Time	Thu Jan 12, 2017 1:51:45 PM
Execution ID	16777216
User	SSB
Fetch Calls	1

Time & Wait Statistics

Duration	1.0s
Database Time	0.9s
PL/SQL & Java	0s
Activity %	100

IO Statistics

Buffer Gets	3
IO Requests	0
IO Bytes	0

Details

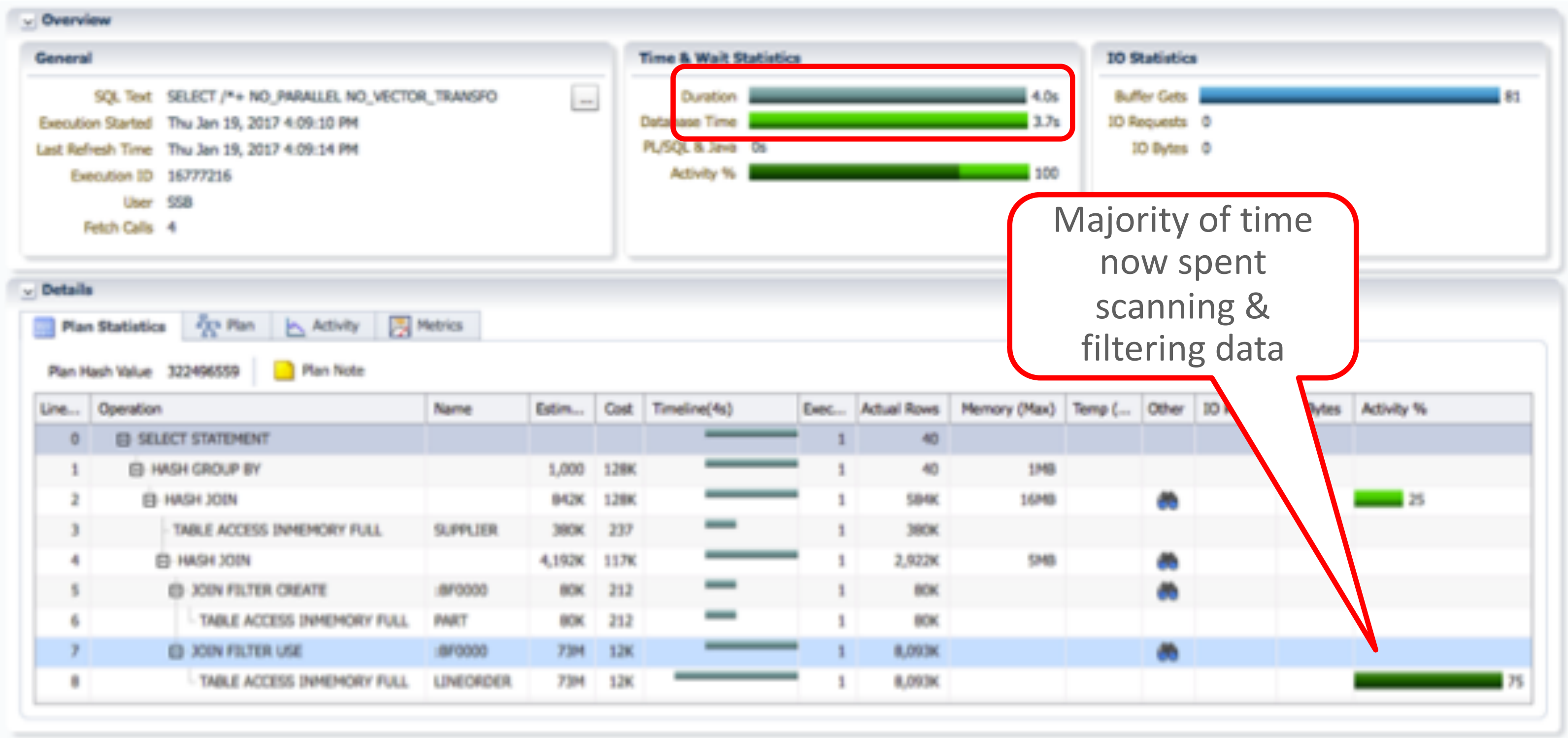
Plan Hash Value 2267213921

Plan Note

Operation	Name	Li...	Estimate...	Cost	Timeline(1s)	Exec...	Actual ...	Memor...	Temp (...	O...	IO Re...	IO ...	y %
⊟ SELECT STATEMENT		0				1	1						
⊟ SORT AGGREGATE		1	1			1	1						
TABLE ACCESS INMEMORY FULL	LINEORDER	2	10M	12K		1	10M						100

Time now spent scanning & filtering data

Joining Data: Hash join with Bloom filters in-memory



SQL Text SELECT /*+ NO_PARALLEL NO_VECTOR_TRANSFO

Execution Started Thu Jan 19, 2017 4:09:10 PM

Last Refresh Time Thu Jan 19, 2017 4:09:14 PM

Execution ID 16777216

User SSB

Fetch Calls 4

Duration 4.0s

Database Time 3.7s

PL/SQL & Java 0s

Activity % 100

Buffer Gets 81

IO Requests 0

IO Bytes 0

Plan Hash Value 322496559

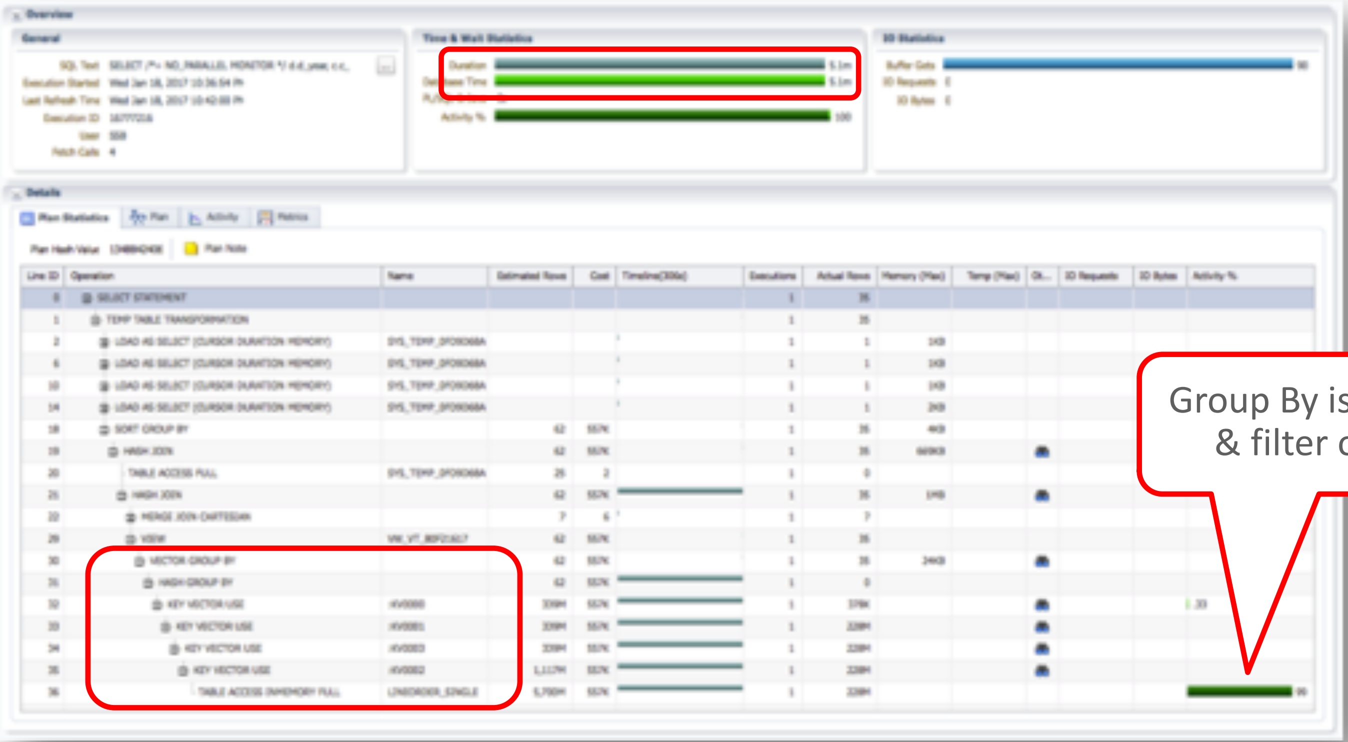
Plan Note

Line...	Operation	Name	Estim...	Cost	Timeline(4s)	Exec...	Actual Rows	Memory (Max)	Temp (...	Other	IO	Bytes	Activity %
0	SELECT STATEMENT					1	40						
1	HASH GROUP BY		1,000	128K		1	40	1MB					
2	HASH JOIN		842K	128K		1	584K	16MB		🔍			25
3	TABLE ACCESS INMEMORY FULL	SUPPLIER	380K	237		1	380K						
4	HASH JOIN		4,192K	117K		1	2,922K	5MB		🔍			
5	JOIN FILTER CREATE	:BF0000	80K	212		1	80K			🔍			
6	TABLE ACCESS INMEMORY FULL	PART	80K	212		1	80K						
7	JOIN FILTER USE	:BF0000	73M	12K		1	8,093K			🔍			
8	TABLE ACCESS INMEMORY FULL	LINEORDER	73M	12K		1	8,093K						75

Majority of time now spent scanning & filtering data

ORACLE

Aggregating Data: Vector Group By with in-memory



SQL Text: SELECT /*+ NO_PARALLEL MONITOR */ d.d_year, c.c_...
Execution Started: Wed Jan 18, 2017 10:36:54 PM
Last Refresh Time: Wed Jan 18, 2017 10:42:00 PM
Execution ID: 16777216
User: SSB
Fetch Calls: 4

Duration: 5.1m
Database Time: 5.1m
PL/SQL & Java: [illegible]
Activity %: 100

Buffer Gets: [illegible]
IO Requests: 0
IO Bytes: 0

Plan Hash Value: [illegible]

Line ID	Operation	Name	Estimated Rows	Cost	Timeline(306s)	Executions	Actual Rows	Memory (Max)	Temp (Max)	Ot...	IO Requests	IO Bytes	Activity %
0	SELECT STATEMENT					1	35						
1	TEMP TABLE TRANSFORMATION					1	35						
2	LOAD AS SELECT (CURSOR DURATION MEMORY)	SYS_TEMP_0FD9D68A				1	1	1KB					
6	LOAD AS SELECT (CURSOR DURATION MEMORY)	SYS_TEMP_0FD9D68A				1	1	1KB					
10	LOAD AS SELECT (CURSOR DURATION MEMORY)	SYS_TEMP_0FD9D68A				1	1	1KB					
14	LOAD AS SELECT (CURSOR DURATION MEMORY)	SYS_TEMP_0FD9D68A				1	1	2KB					
18	SORT GROUP BY		42	557K		1	35	4KB					
19	HASH JOIN		42	557K		1	35	669KB		●			
20	TABLE ACCESS FULL	SYS_TEMP_0FD9D68A	25	2		1	0						
21	HASH JOIN		42	557K		1	35	1MB		●			
22	MERGE JOIN CARTESIAN		7	6		1	7						
29	VIEW	VW_VT_80F21617	42	557K		1	35						
30	VECTOR GROUP BY		42	557K		1	35	24KB		●			
31	HASH GROUP BY		42	557K		1	0						
32	KEY VECTOR USE	:KV0000	339M	557K		1	378K			●			33
33	KEY VECTOR USE	:KV0001	339M	557K		1	228M			●			
34	KEY VECTOR USE	:KV0003	339M	557K		1	228M			●			
35	KEY VECTOR USE	:KV0002	1,117M	557K		1	228M			●			
36	TABLE ACCESS INMEMORY FULL	LINEORDER_SINGLE	5,700M	557K		1	228M						66

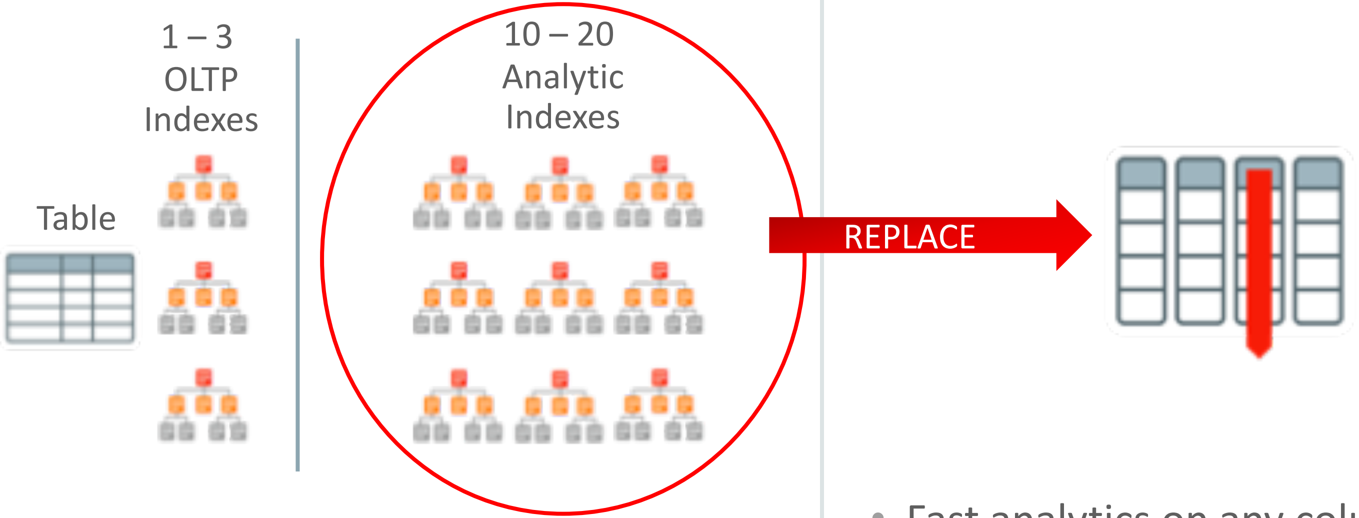
Group By is now a scan & filter operation

What about indexes

Database In-Memory Accelerates Mixed Workloads

- Complex OLTP is Slowed by Analytic Indexes

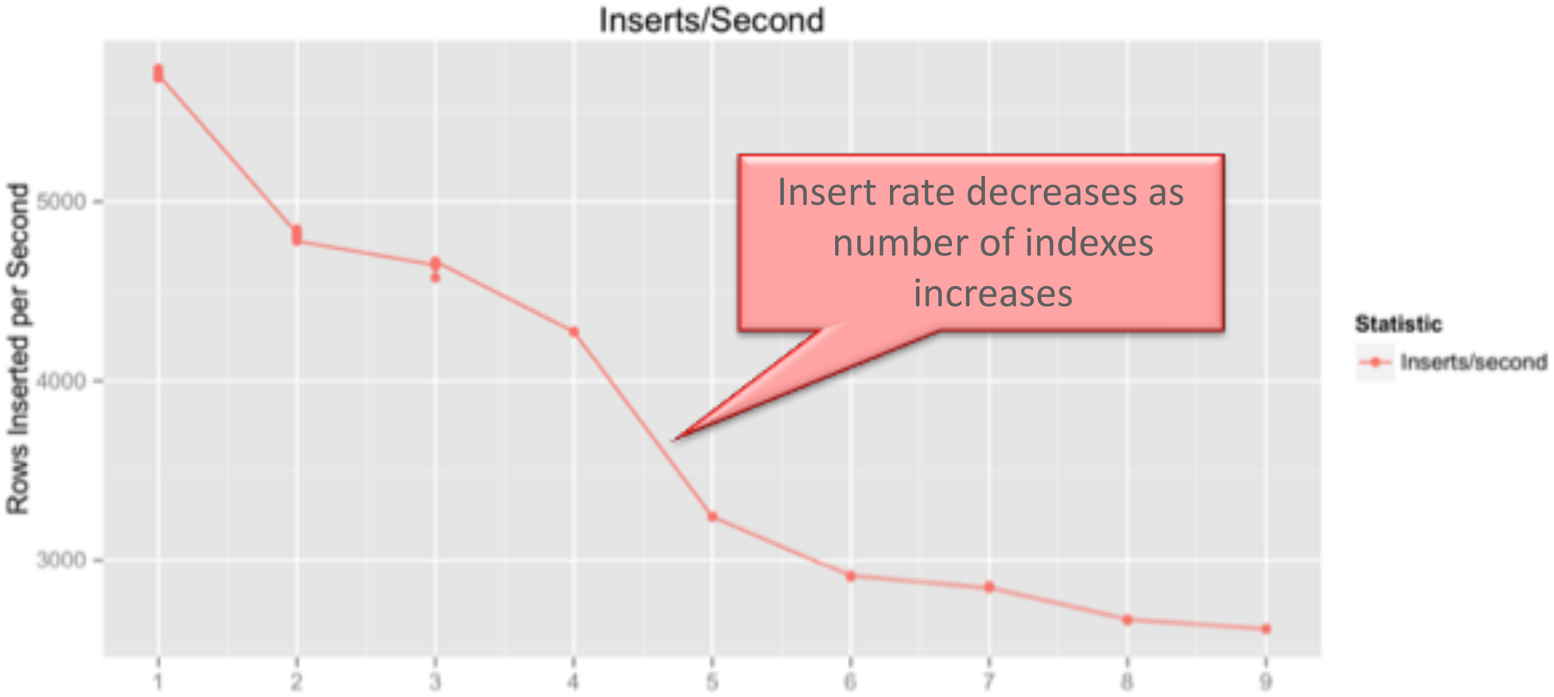
- Column Store Replaces Analytic Indexes



- Inserting one row into a table requires updating 10-20 analytic indexes: Slow!

- Fast analytics on <u>any</u> column
- Column Store not persistent so update cost is much lower

OLTP is Slowed Down by Analytic Indexes



Insert rate decreases as number of indexes increases

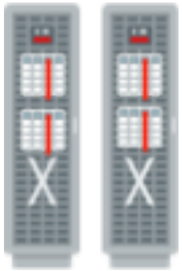
# of Fully Cached Indexes (Disk Indexes are much slower)

Does It Work With Other Oracle Database Features

ORACLE

Database In-Memory: Other Features

Scale-Out

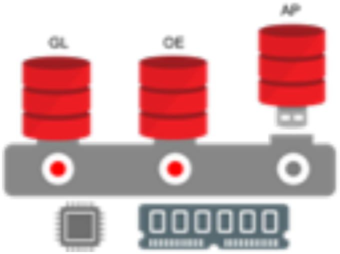


- Scale-Out Across Servers to Grow Memory and CPUs
- In-Memory Queries Parallelized Across Servers

Scale-Up

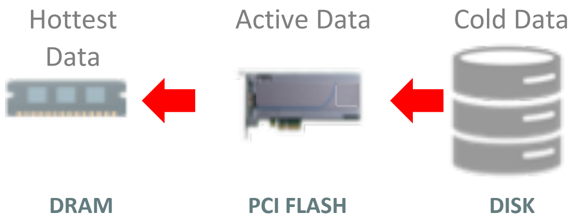
- Scale-Up on large SMPs
- NUMA Optimized

Consolidation



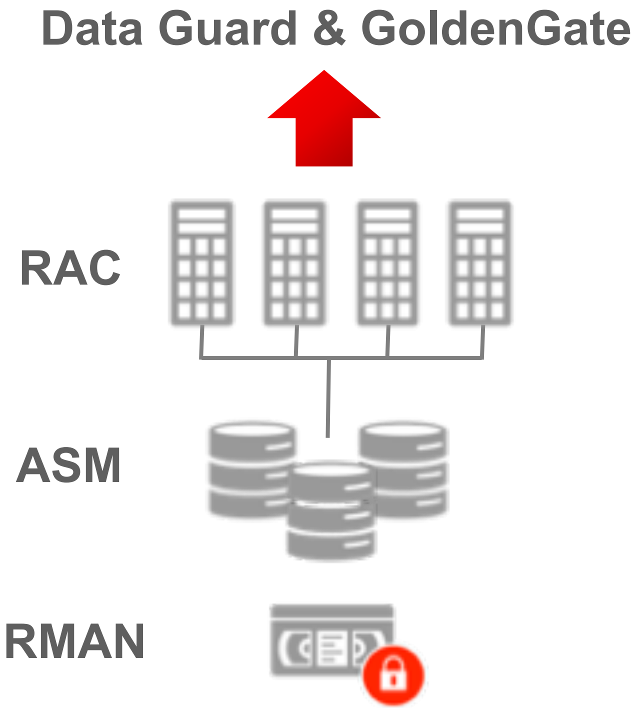
- Frees up memory and CPU
- Shares memory and background processes
- Column store defined at CDB level

Combine with Flash and Disk



- Easily place data on most cost effective tier
- Simultaneously Achieve:
  - Speed of DRAM
  - I/Os of Flash
  - Cost of Disk

Database In-Memory: Industrial Strength Availability



- Pure In-Memory format does not change Oracle’s storage format, logging, backup, recovery, etc.

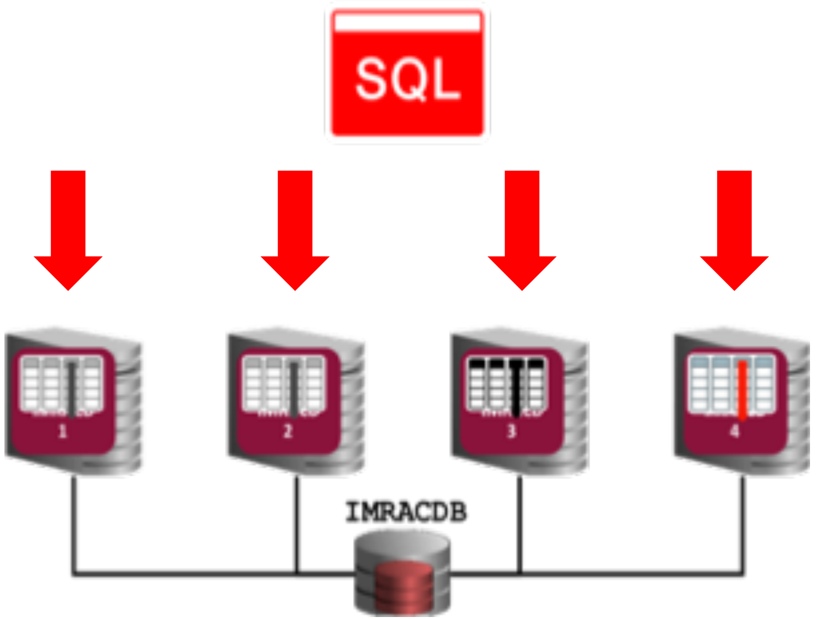
- All Oracle’s proven availability technologies work transparently

- Protection from all failures
  - Node, site, corruption, human error, etc.

How does Database In-Memory Work With RAC

RAC : Scale-Out In-Memory Database to Any Size

- Scale-Out across servers to grow memory and CPUs
- Shared nothing architecture
- IMCUs not shipped across interconnect – cache fusion is not in play!
- In-Memory **queries are parallelized** across servers to access local column data



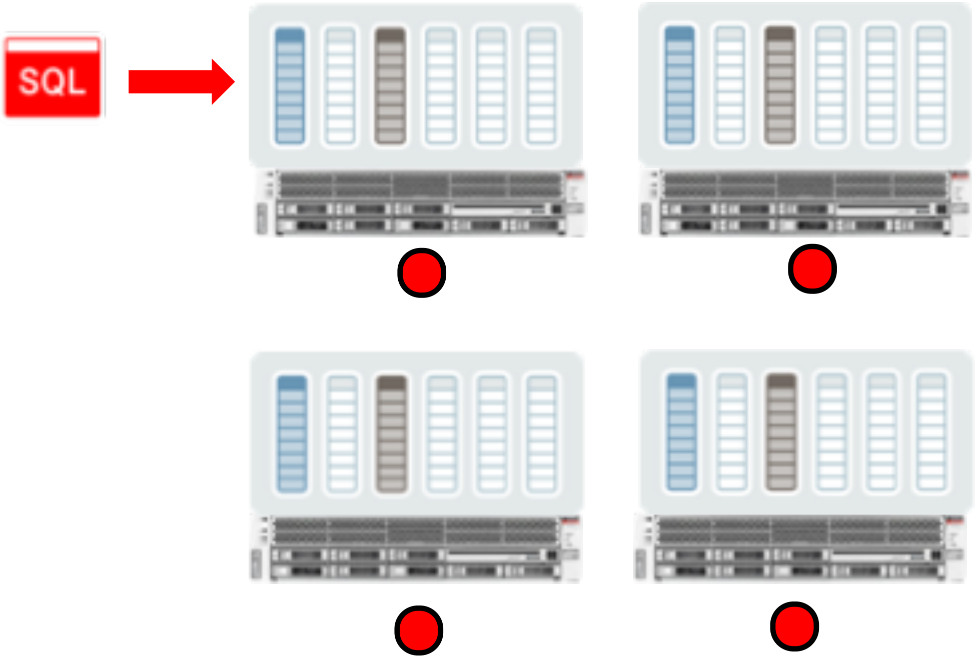
RAC : In-Memory and Distribution of Data

```
ALTER TABLE sales INMEMORY;

ALTER TABLE sales INMEMORY
DISTRIBUTE BY PARTITION;

ALTER TABLE sales INMEMORY
DISTRIBUTE ROWID RANGE;
```

- Distribution allows in memory segments larger than individual instance memory
- Policy is automatic (*Distribute AUTO*) or user-specifiable
- Controlled by DISTRIBUTE subclause
  - Distribute by rowid range
  - Distribute by partition
  - Distribute by subpartition
- Goal: Ensure Even Distribution

ORACLE

RAC : Database In-Memory Queries in a RAC Environment

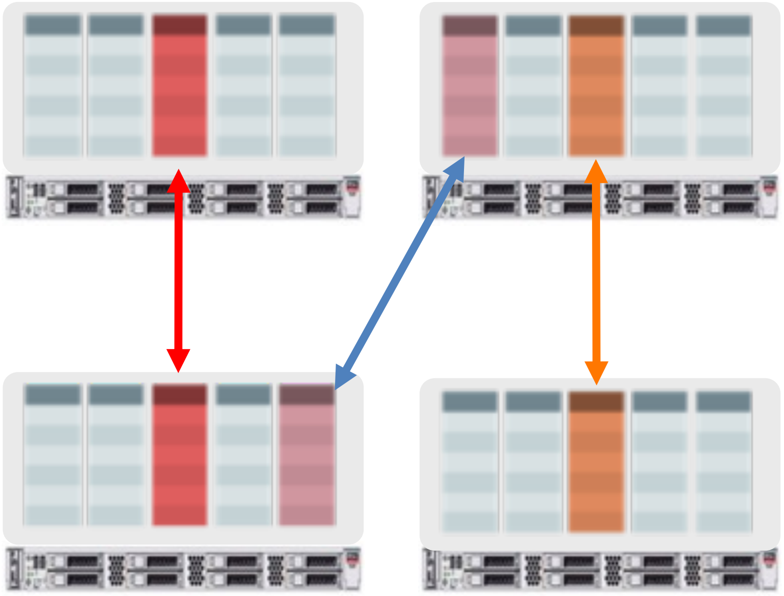


- Shared nothing architecture means **Parallel Query** must be used to access data

- Must have a DOP greater than or equal to the number of column stores

- Query coordinator automatically starts parallel server processes on the correct nodes (Requires Auto DOP in 12.1.0.2)

Engineered Systems: Unique Fault Tolerance



Only Available on Engineered Systems

- Similar to storage mirroring
- Duplicate in-memory columns on another node
  - Enabled per table/partition
  - Application transparent
- Performance preserved by using duplicate during a node failure
- Performance can be improved by performing joins within each node (partial partition wise joins)

What’s New

Database In-Memory New Features

Performance

- In-Memory Expressions
- Join Groups
- In-Memory Dynamic Scans
- In-Memory Optimized Arithmetic

In-Memory Column Store
Hot Data
Intermediate Data
Cool Data
Remove Cold Table

Managability

- Automatic Data Optimization
- Automatic In-Memory

Expanded Capacity

- Exadata Flash
- Active Data Guard
- External Tables

How Have Customers Benefited From In-Memory

How Customers Use Database In-Memory

AT&T WiFi – Data Warehouse

- Business Objects reports 100X faster
- ETL processes improved by 50% faster
- No changes to SAP Business Objects reports

Villeroy & Boch – SAP BW

- SAP BW COPA queries 30 – 33X faster
- SAP Transaction list queries 4 – 4,800X faster
- Avoided expensive & risky upgrade to S4/Hana

BOSCH – SAP CRM

- Dropped all custom indexes
- Analytic queries 2-20X faster, DML 2-3X faster
- No changes to application required

Die Mobiliar – Mixed Workload

- Analytic queries 50-200X faster
- Database size reduced considerably
- Phase out of Netezza and mainframe systems

How Customers Use Database In-Memory

Mankind Pharma – Mixed Workload

- Analytical reports 11x faster
- Dropping indexes improved OLTP
- 90% reduction in database size

Shanghai Customs – Mixed Workload

- Processes Clearance 43x Faster
- Improves Declaration-Services Efficiency
- Reduced Costs

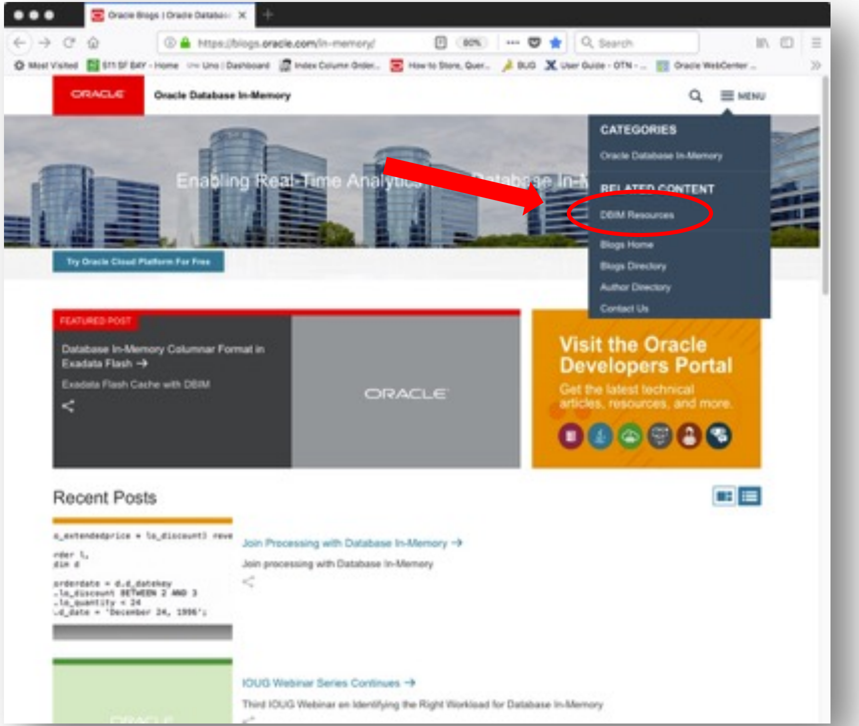
- **LION** – SAP ERP
- Analytic queries 4X faster
- Transactions 2X faster
- Analytic queries now possible on 100s of Millions of Point-of-Sales Transactions

Lufthansa – Reporting Application

- Analytic queries up to 100x faster
- Improved data ingest performance
- Reduction in database size

Where Can You Get More Information

https://blogs.oracle.com/in-memory/dbim-resources



Database In-Memory Columnar Format in Exadata Flash →

Exadata Flash Cache with DBIM

Visit the Oracle Developers Portal

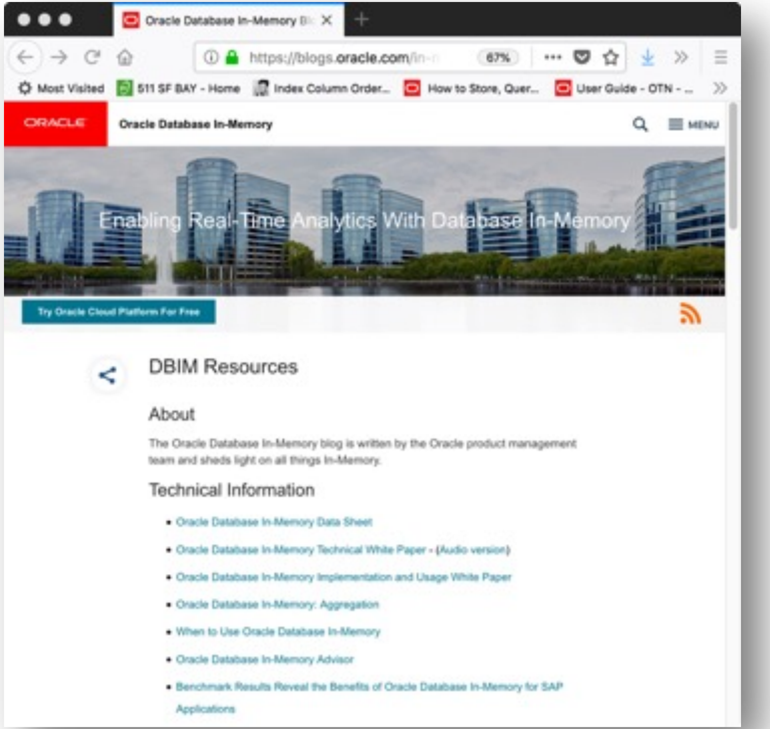
Get the latest technical articles, resources, and more.

Join Processing with Database In-Memory →

Join processing with Database In-Memory

IOUG Webinar Series Continues →

Third IOUG Webinar on Identifying the Right Workload for Database In-Memory


Oracle Database In-Memory B
https://blogs.oracle.com/in-
67%
Most Visited
511 SF BAY - Home
Index Column Order...
How to Store, Quer...
User Guide - OTN - ...
ORACLE
Oracle Database In-Memory
MENU
Enabling Real-Time Analytics With Database In-Memory
Try Oracle Cloud Platform For Free

DBIM Resources

About

The Oracle Database In-Memory blog is written by the Oracle product management team and sheds light on all things In-Memory.

Technical Information

- Oracle Database In-Memory Data Sheet
- Oracle Database In-Memory Technical White Paper - (Audio version)
- Oracle Database In-Memory Implementation and Usage White Paper
- Oracle Database In-Memory: Aggregation
- When to Use Oracle Database In-Memory
- Oracle Database In-Memory Advisor
- Benchmark Results Reveal the Benefits of Oracle Database In-Memory for SAP Applications

Additional Resources

Join the Conversation

- https://twitter.com/db_inmemory
- https://twitter.com/TheInMemoryGuy
- http://www.oracle.com/goto/dbim.html

Database In-Memory Information

[Database In-Memory Blog]

[oracle.com – Database In-Memory]

[Database In-Memory YouTube Channel]

[Ask TOM Database In-Memory Office Hours]

[Database In-Memory Guide (Documentation)]

ORACLE