Oracle Database In-Memory

A Focus On The Technology

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

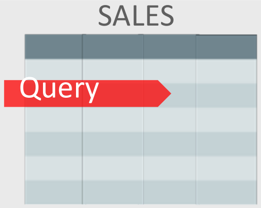
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development,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

What is Database In-Memory

ORACLE

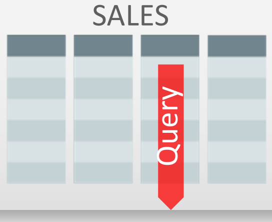
Row Format Databases vs. Column Format Databases

Rows Stored Contiguously



- **Transactions** run faster on row format
  - Example: Query or Insert a sales order
  - Fast processing few rows, many columns

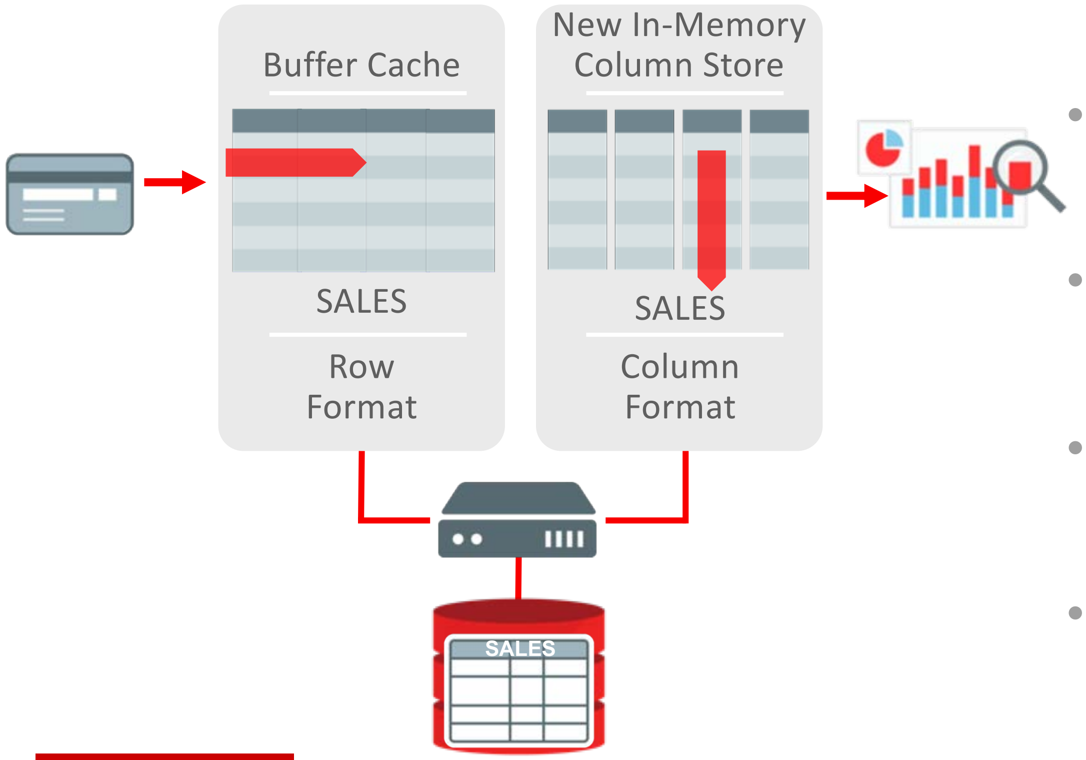
Columns Stored Contiguously



- **Analytics** run faster on column format
  - Example : Report on sales totals by region
  - Fast accessing few columns, many rows

Until Now Must Choose One Format and Suffer Tradeoffs

Breakthrough: Dual Format Database



Buffer Cache

SALES

Row Format

New In-Memory Column Store

SALES

Column Format

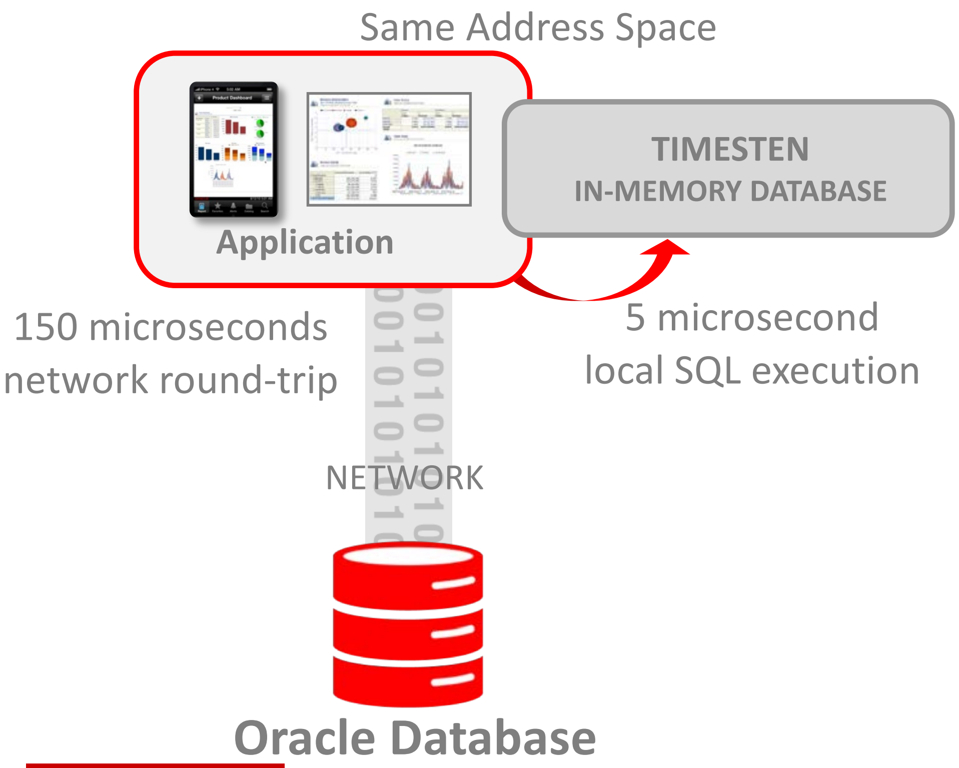
SALES

- **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

Isn't it just TimesTen

TimesTen for Latency-Critical OLTP

Complementary In-Memory Technology



Same Address Space

Application

TIMESTEN
IN-MEMORY DATABASE

150 microseconds
network round-trip

5 microsecond
local SQL execution

NETWORK

Oracle Database

- **Latency-Critical OLTP limited by network between application and database**
  - Phone call routing, stock trading
- TimesTen In-Memory Database is light-weight and ultra-fast
  - Runs in application address space: No Network
  - **30x** faster latency-critical OLTP

How easy is it to get started

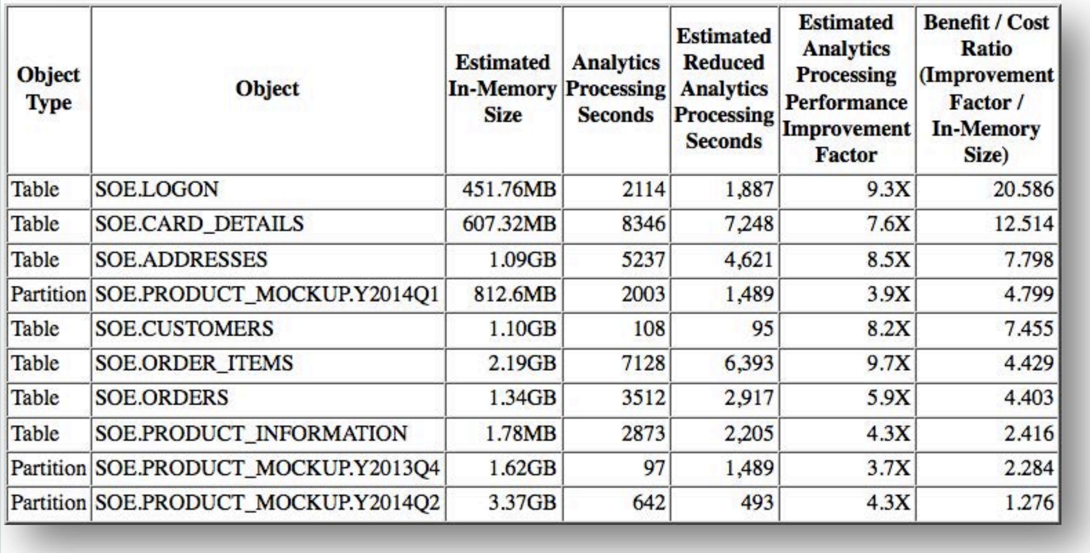
Oracle In-Memory: Simple to Implement

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

Oracle In-Memory Advisor

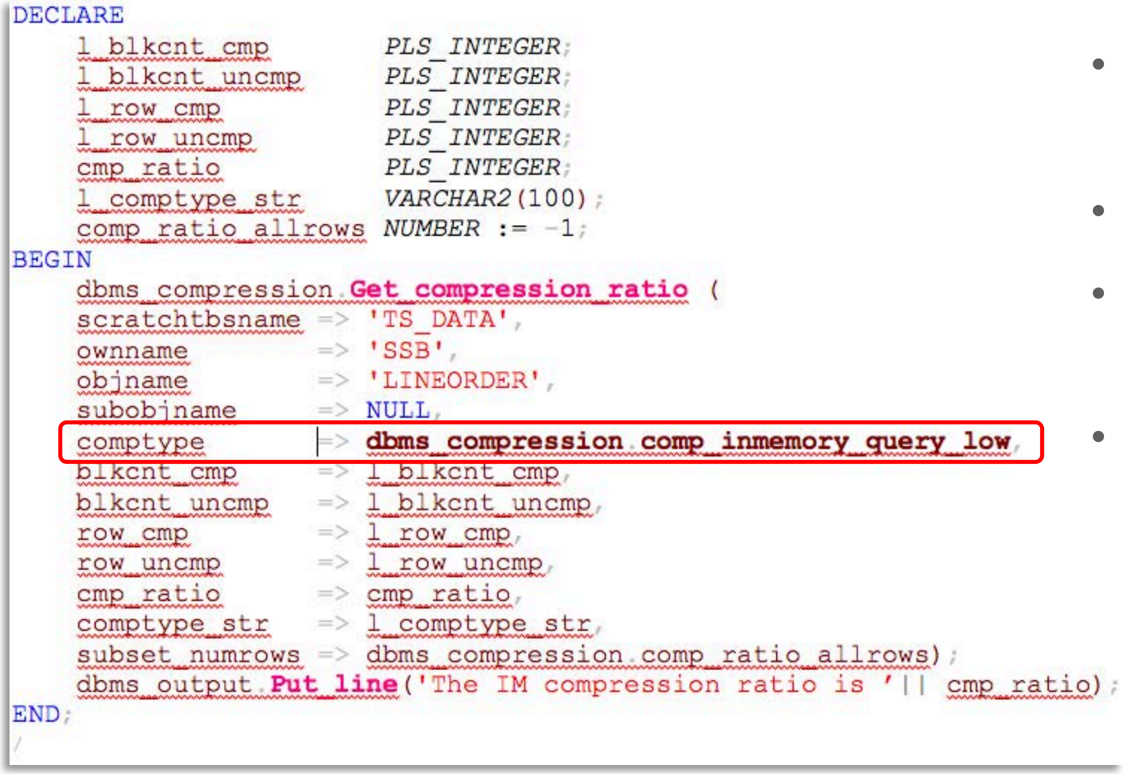


Object Type	Object	Estimated In-Memory Size	Analytics Processing Seconds	Estimated Reduced Analytics Processing Seconds	Estimated Analytics Processing Performance Improvement Factor	Benefit / Cost Ratio (Improvement Factor / In-Memory Size)
Table	SOE.LOGON	451.76MB	2114	1,887	9.3X	20.586
Table	SOE.CARD_DETAILS	607.32MB	8346	7,248	7.6X	12.514
Table	SOE.ADDRESSES	1.09GB	5237	4,621	8.5X	7.798
Partition	SOE.PRODUCT_MOCKUP.Y2014Q1	812.6MB	2003	1,489	3.9X	4.799
Table	SOE.CUSTOMERS	1.10GB	108	95	8.2X	7.455
Table	SOE.ORDER_ITEMS	2.19GB	7128	6,393	9.7X	4.429
Table	SOE.ORDERS	1.34GB	3512	2,917	5.9X	4.403
Table	SOE.PRODUCT_INFORMATION	1.78MB	2873	2,205	4.3X	2.416
Partition	SOE.PRODUCT_MOCKUP.Y2013Q4	1.62GB	97	1,489	3.7X	2.284
Partition	SOE.PRODUCT_MOCKUP.Y2014Q2	3.37GB	642	493	4.3X	1.276

- New In-Memory Advisor
- Analyzes existing DB workload via AWR & ASH repositories
- Provides list of objects that would benefit most from being populated into IM column store

Note: Database Tuning Pack license required

Oracle Compression Advisor And In-Memory

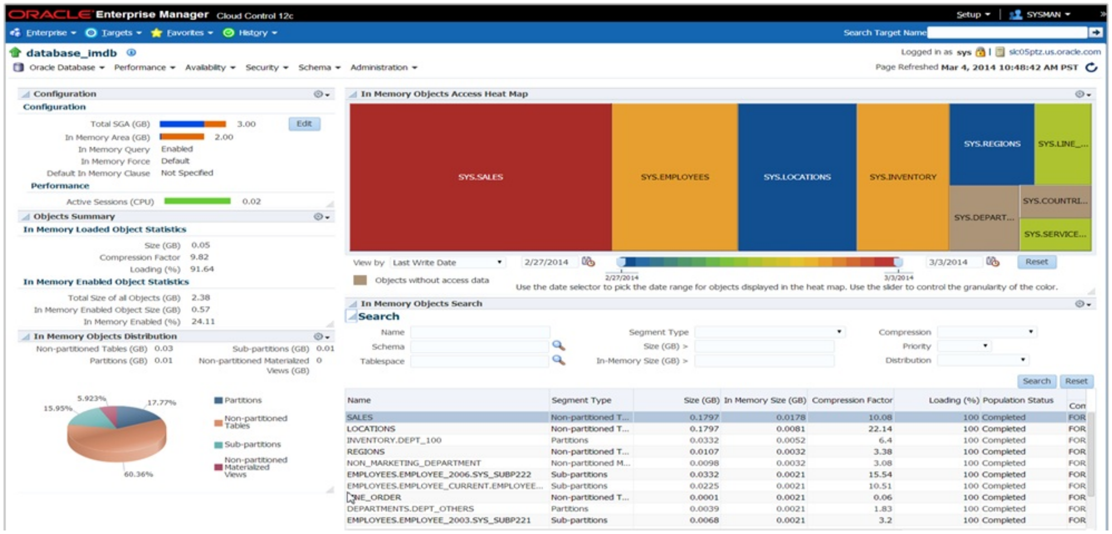


```
DECLARE
    l_blkcnt_cmp        PLS_INTEGER;
    l_blkcnt_uncmp      PLS_INTEGER;
    l_row_cmp           PLS_INTEGER;
    l_row_uncmp         PLS_INTEGER;
    cmp_ratio           PLS_INTEGER;
    l_comptype_str      VARCHAR2(100);
    comp_ratio_allrows  NUMBER := -1;
BEGIN
    dbms_compression.Get_compression_ratio (
    scratchtbsname  => 'TS_DATA',
    ownname         => 'SSB',
    objname         => 'LINEORDER',
    subobjname      => NULL,
    comptype        => dbms_compression.comp_inmemory_query_low,
    blkcnt_cmp      => l_blkcnt_cmp,
    blkcnt_uncmp    => l_blkcnt_uncmp,
    row_cmp         => l_row_cmp,
    row_uncmp       => l_row_uncmp,
    cmp_ratio       => cmp_ratio,
    comptype_str    => l_comptype_str,
    subset_numrows  => dbms_compression.comp_ratio_allrows);
    dbms_output.Put_line('The IM compression ratio is '|| 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

Oracle Enterprise Manager: In-Memory Central

In-Memory Central



Logged in as sys | slc05ptz.us.oracle.com

Page Refreshed Mar 4, 2014 10:48:42 AM PST

Configuration

Configuration

Total SGA (GB) 3.00

In Memory Area (GB) 2.00

In Memory Query Enabled

In Memory Force Default

Default In Memory Clause Not Specified

Performance

Active Sessions (CPU) 0.02

Objects Summary

In Memory Loaded Object Statistics

Size (GB) 0.05

Compression Factor 9.82

Loading (%) 91.64

In Memory Enabled Object Statistics

Total Size of all Objects (GB) 2.38

In Memory Enabled Object Size (GB) 0.57

In Memory Enabled (%) 24.11

In Memory Objects Distribution

Non-partitioned Tables (GB) 0.03

Sub-partitions (GB) 0.01

Partitions (GB) 0.01

Non-partitioned Materialized Views (GB) 0

In Memory Objects Access Heat Map

View by Last Write Date ▾ 2/27/2014 3/3/2014

Objects without access data

Use the date selector to pick the date range for objects displayed in the heat map. Use the slider to control the granularity of the color.

In Memory Objects Search

Search

Name

Schema

Tablespace

Segment Type ▾

Size (GB) >

In-Memory Size (GB) >

Compression ▾

Priority ▾

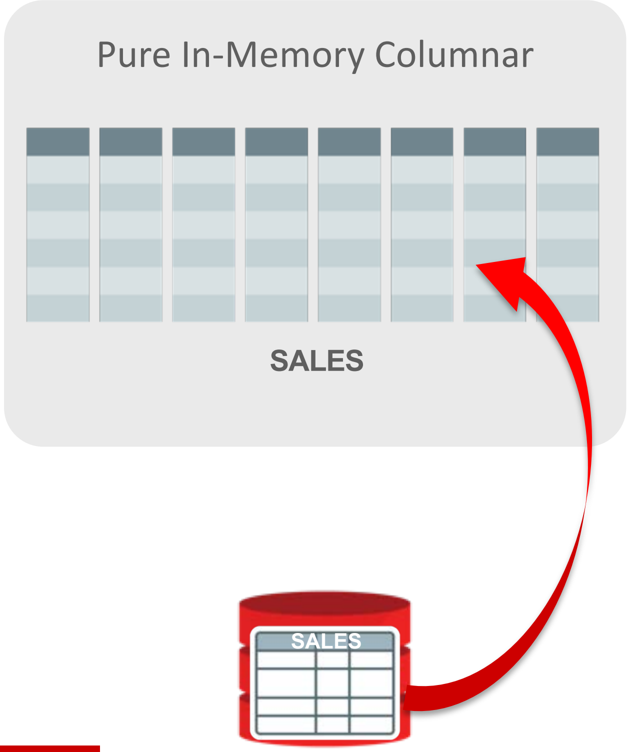
Distribution ▾

Name	Segment Type	Size (GB)	In Memory Size (GB)	Compression Factor	Loading (%)	Population Status	Con
SALES	Non-partitioned T...	0.1797	0.0178	10.08	100	Completed	FOR
LOCATIONS	Non-partitioned T...	0.1797	0.0081	22.14	100	Completed	FOR
INVENTORY.DEPT_100	Partitions	0.0332	0.0052	6.4	100	Completed	FOR
REGIONS	Non-partitioned T...	0.0107	0.0032	3.38	100	Completed	FOR
NON_MARKETING_DEPARTMENT	Non-partitioned M...	0.0098	0.0032	3.08	100	Completed	FOR
EMPLOYEES.EMPLOYEE_2006.SYS_SUBP222	Sub-partitions	0.0332	0.0021	15.54	100	Completed	FOR
EMPLOYEES.EMPLOYEE_CURRENT.EMPLOYEE...	Sub-partitions	0.0225	0.0021	10.51	100	Completed	FOR
NE_ORDER	Non-partitioned T...	0.0001	0.0021	0.06	100	Completed	FOR
DEPARTMENTS.DEPT_OTHERS	Partitions	0.0039	0.0021	1.83	100	Completed	FOR
EMPLOYEES.EMPLOYEE_2003.SYS_SUBP221	Sub-partitions	0.0068	0.0021	3.2	100	Completed	FOR

- OEM supports Database In-Memory
- In-Memory Central page gives a dashboard look to the IM column store
- Provides list of objects populated in the IM column store

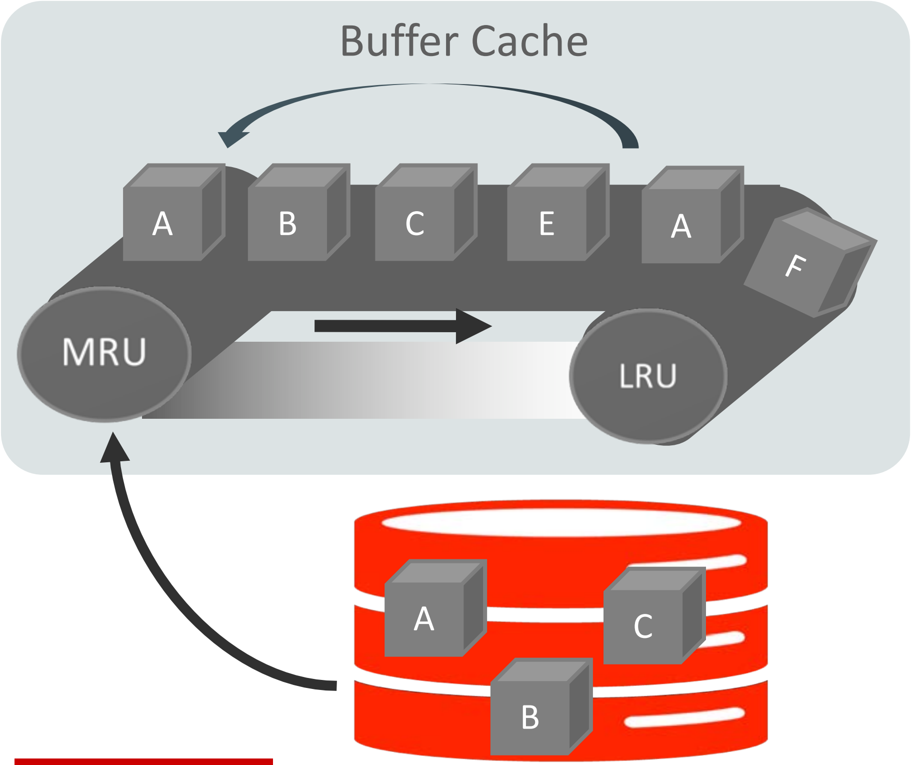
How does it work

Oracle In-Memory Columnar Technology



- Pure in-memory columnar format
  - Not persistent, and no logging
  - Quick to change data: fast OLTP
- Enabled at table or partition
  - Only active data in-memory
- 2x to 20x compression typical
- Available on all hardware platforms

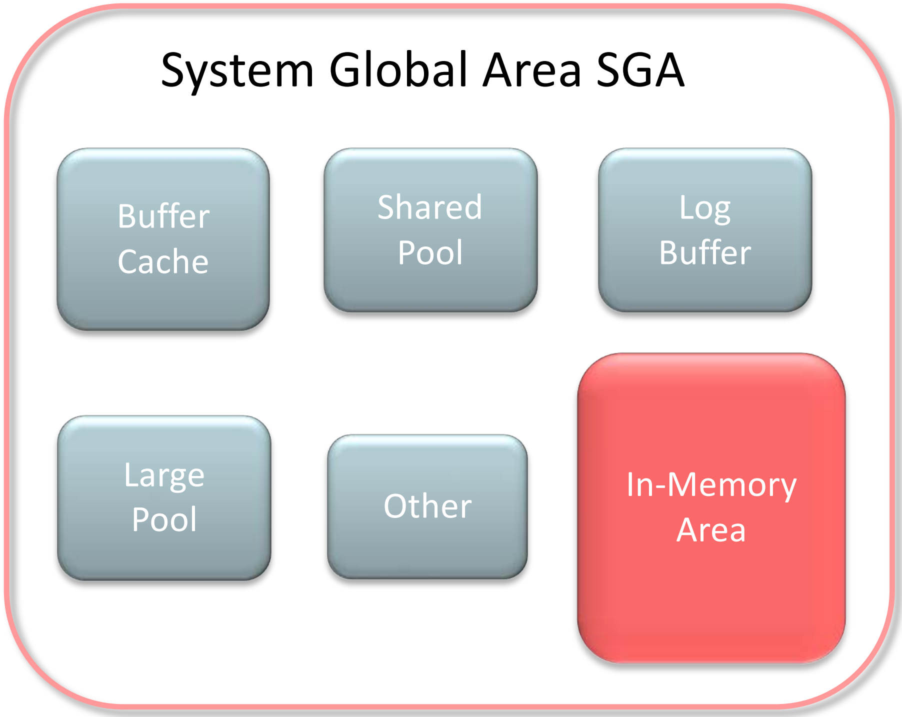
In-Memory A Store – Not A Cache



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

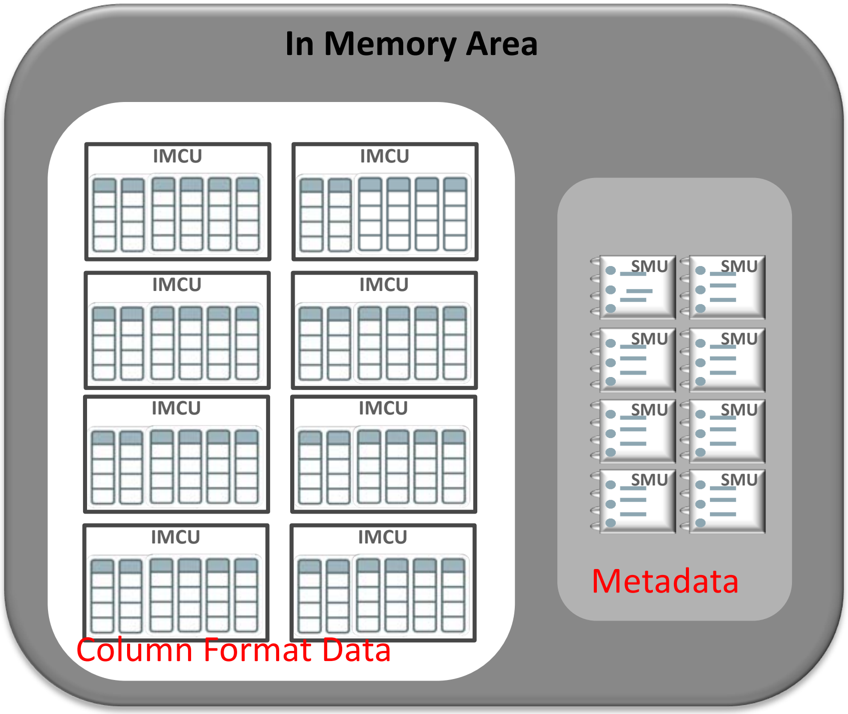
  Oracle Database Buffer Cache

In-Memory Area: Static Area within SGA



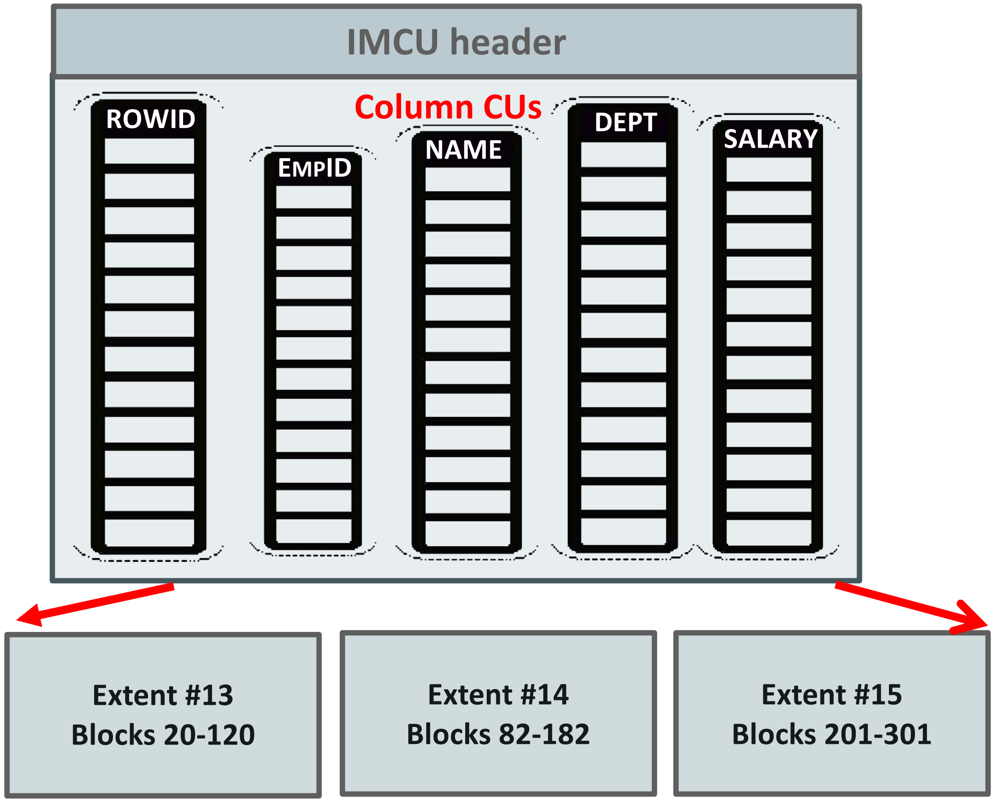
- Contains data in the new In-Memory Column Format
- Controlled by INMEMORY_SIZE parameter
  - Minimum size of 100MB
- SGA_TARGET must be large enough to accommodate this area

Composition of In-Memory Area



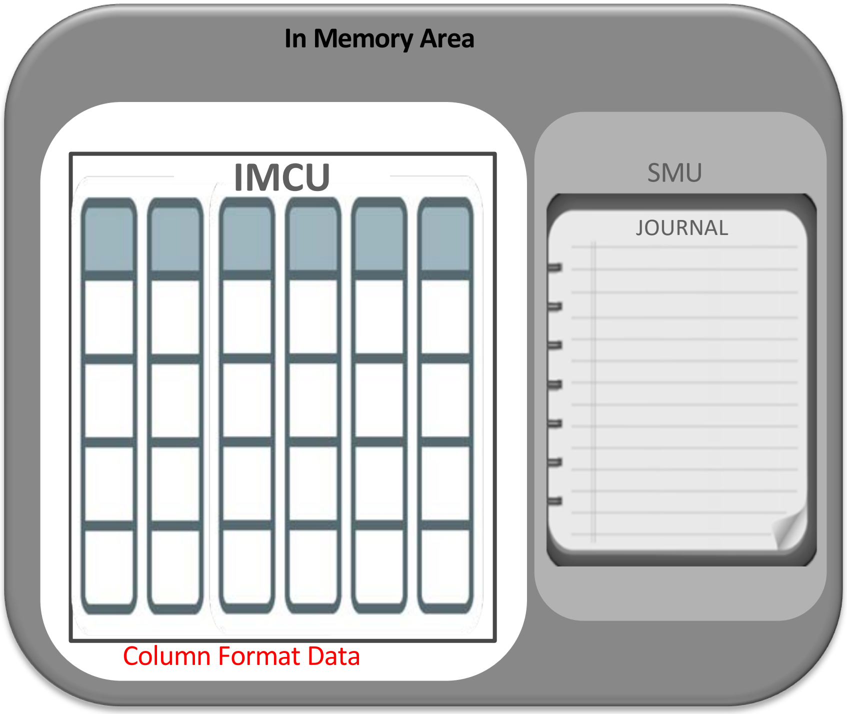
- Contains two subpools:
  - IMCU pool: Stores In Memory Compression Units (IMCUs)
  - SMU pool: Stores Snapshot Metadata Units (SMUs)
- IMCUs contain column formatted data
- SMUs contain metadata and transactional information

Composition of In-Memory Compression Unit (IMCU)



- Unit of column store allocation
  - Columnar representation of a large number of rows from an object
  - Rows from one or more table extents
- Actual size depends on size of rows, compression factor, etc.
- Each column stored as a separate contiguous **Column Compression Unit** (column CU)
  - Rowids also stored as a Column CU

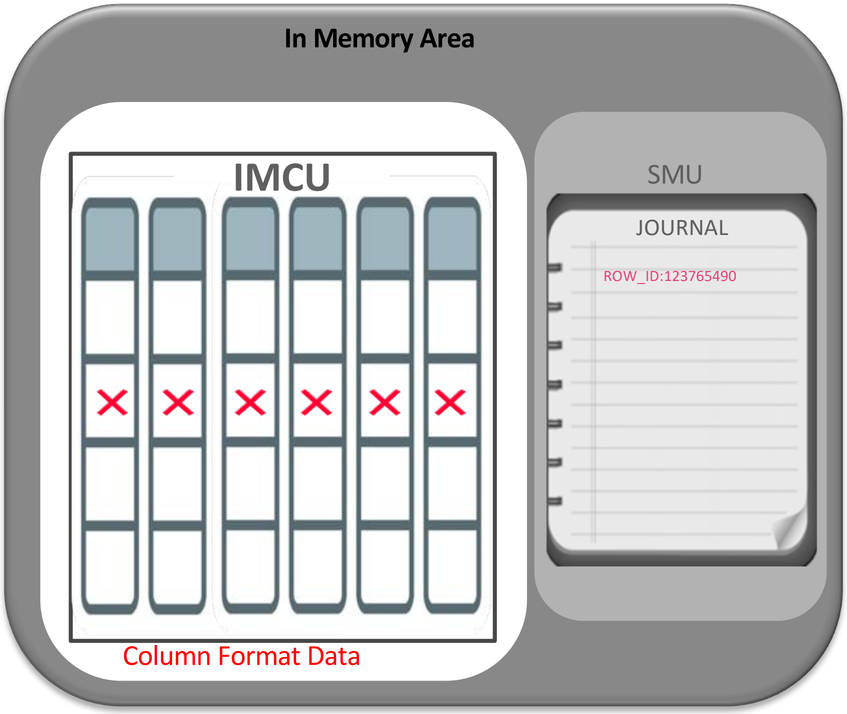
DML and the In-Memory Column Store



- Each IMCU contains the column entries for a subset of rows in the object

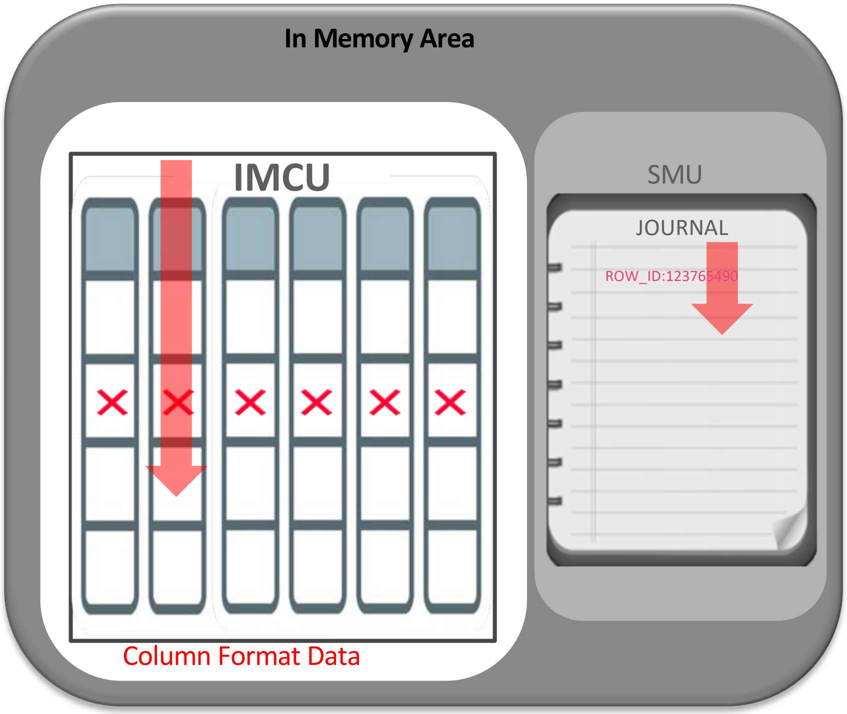
- The SMU, associated with the IMCU, has a transaction journal that is used to keep the column store transactionally consistent

DML and the In-Memory Column Store



- DML operations processed in row store just as they are today
- Corresponding entry in column store marked stale as of SCN
- ROWID of row stored in the Transaction Journal

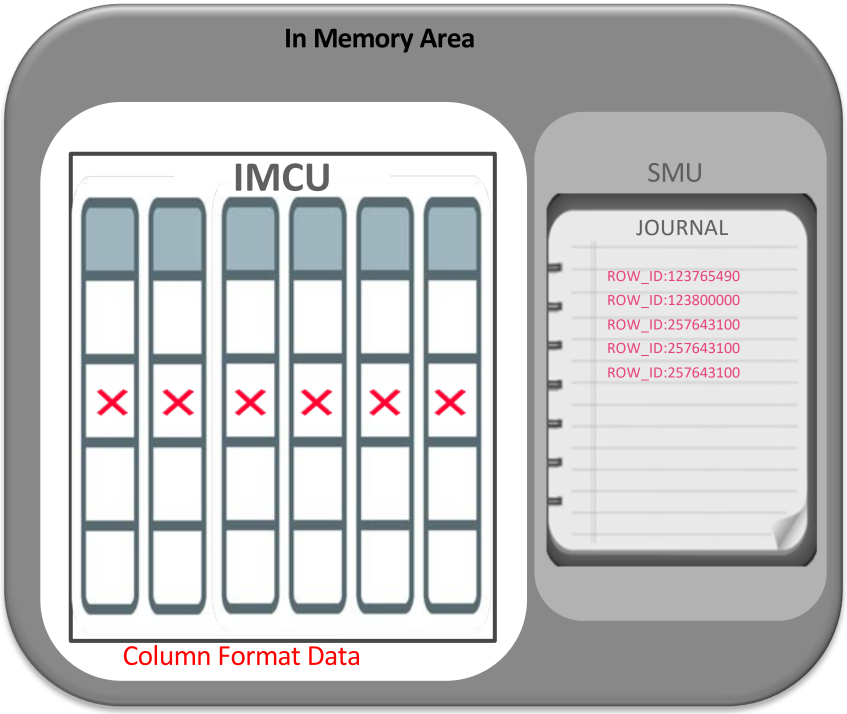
DML and the In-Memory Column Store



- In-Memory Column Store is never out of date

- Read-consistency is achieved by merging contents of column, the transaction journal and buffer cache

DML and the In-Memory Column Store



- When number of entries in transaction journal hits an internal threshold, IMCU is automatically repopulated
- This is an online operation
  - IM column store always available
  - DMLs are not blocked

Factors That Impact Repopulation Performance

- The rate of change
- In-Memory compression level chosen
  - Tables with higher compression levels will incur more overhead
- Location of the changed rows
  - Changes co-located in same database block or partition have less impact then changes that are distributed across table
- Type of operations being performed
  - Inserts cheaper than deletes and updates
- Number of active worker processes
  - INMEMORY_REPOPULATE_SERVERS
  - INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT

How Do I Get Data In And Out Of The In-Memory Column Store?

Populating : Enable Objects for In-Memory

```
ALTER TABLE sales INMEMORY;

ALTER TABLE sales NO INMEMORY;


CREATE TABLE customers ……
PARTITION BY LIST
   (PARTITION p1 …… INMEMORY,
   (PARTITION p2 …… NO INMEMORY);
```

- New INMEMORY ATTRIBUTE
- Eligible segment types are
  - Tables
  - Partitions
  - Subpartitions
  - Materialized views
- Following types not eligible
  - IOTs
  - Hash clusters
  - Out of line LOBs

Pure OLTP Features

ORACLE

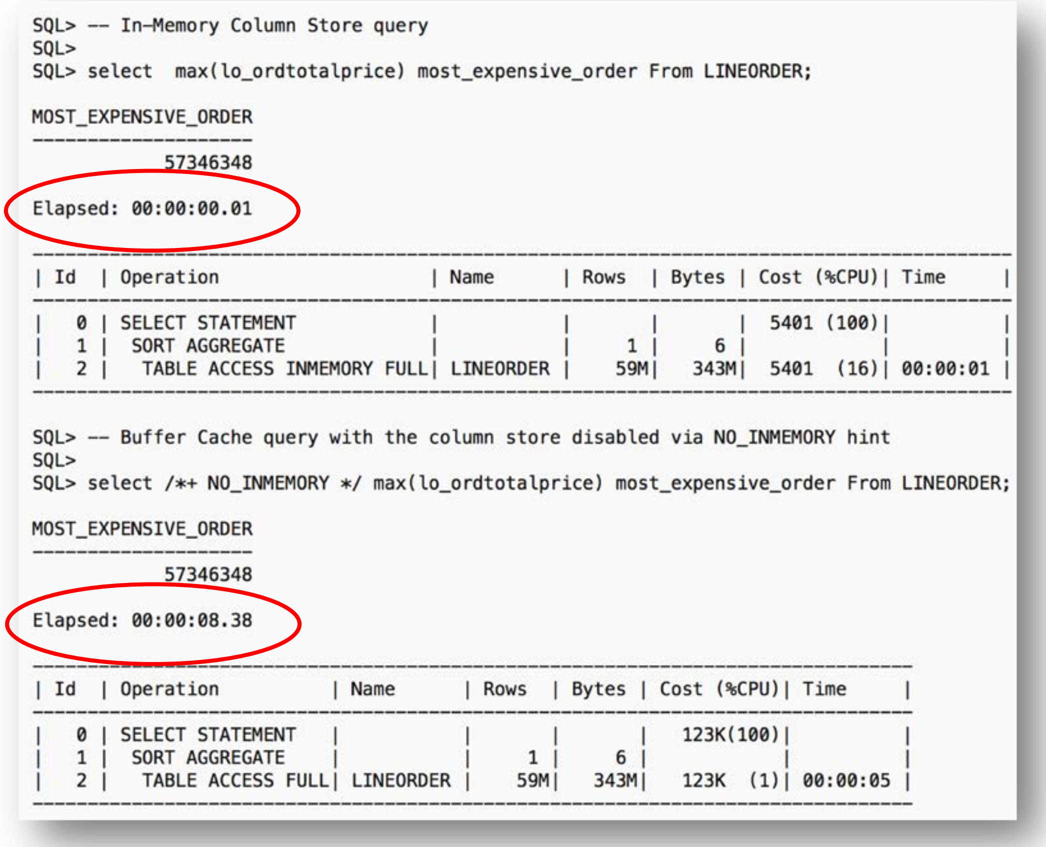
Populating : Columns Can Be Excluded

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

- You don't have to populate all columns
- It is possible to populate only certain columns
- 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

Why not just ”cache” the table in the row store

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 |
--------------------------------------------------------------------------------
```

Why Are Analytic Queries Faster In The In-Memory Column Store?

Database In-Memory Technology

Scanning and filtering data more efficiently

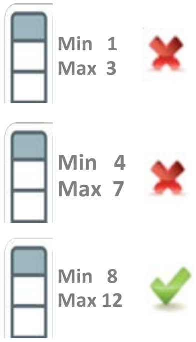
Columnar Format

Access only the columns you need

Compression

Scan & filter data in compressed format

Storage Indexes



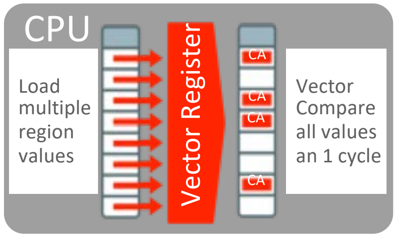
Min 1
Max 3 ✗

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

ORACLE

What does the Optimizer know about Database In-Memory

Be Afraid ….Optimizer gets to Decide

- The Optimizer gets to pick the execution plan for queries
  - It decides if the In-Memory column store will be used or not
- Traditional cost model assumes all scan operations will read data from disk
- The cost model was expanded to account for In-Memory scans too
- Cost is now computed based on statistics maintained on
  - Objects: tables, columns, indexes, partitions etc.
  - System: CPU speed, IO throughput, etc.
  - In-Memory tables: In-Memory specific statistics

You are still on the hook to get these statistics correct

Optimizer : New In-Memory Statistics

- Automatically computed on the fly during hard parse
- Computed at the segment level – table or partition/subpartition
  - **# IMCUs**
  - **# IM Blocks**
  - **IM Quotient**
    - Fraction of table populated in In-Memory column store
    - Value between 0 and 1
  - **# IM Rows**
  - **# IM Transaction Journal Rows**
- In-Memory statistics are RAC-aware (DUPLICATE and DISTRIBUTE)
- In 12.2 IM stats are available in the ALL_TAB_STATISTICS view

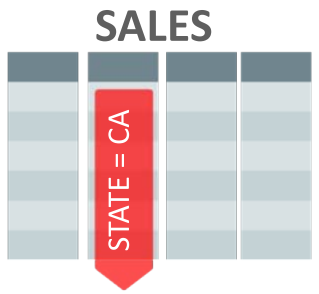
In-Memory Aware Cost Model

- Cost of In-Memory Scan
  - **IO cost:** Includes the cost of reading:
    - Invalid rows from disk
    - Extent map
  - **CPU cost:** Includes
    - Traversing IMCUs
    - IMCU pruning using storage indexes
    - Decompressing IMCUs
    - Predicate evaluation
    - Stitching rows
    - Scanning transaction journal rows

Optimizer Enhancements

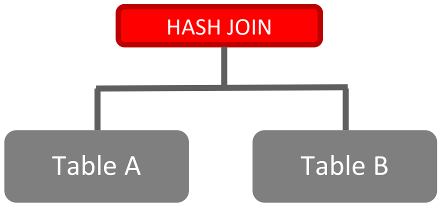
Improves all aspects of analytic queries

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

Pushing Predicates to In-Memory Scans

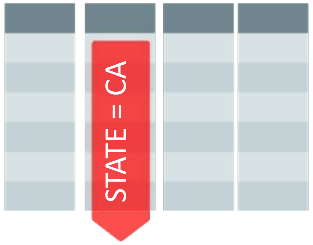
- Many types of filter predicates can be more efficiently evaluated **during** the In-Memory scan rather than **after**
  - Only scan the columns needed for the query
  - Prune IMCUs using storage indexes and dictionary-based compression metadata
  - Evaluate predicates directly against compressed columnar data
  - Use SIMD to evaluate predicates on multiple column values concurrently
- What predicates can be pushed to be evaluated in memory?
  - Predicates that are in the WHERE clause
  - Predicates **implied** by the WHERE clause

Database In-Memory: Real-time Analytics

Improves key aspects of analytic queries

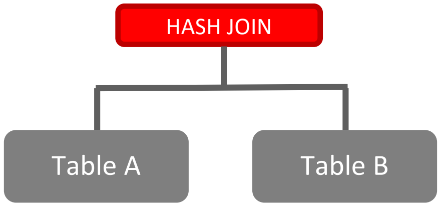
Data Scans

SALES



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

Joins

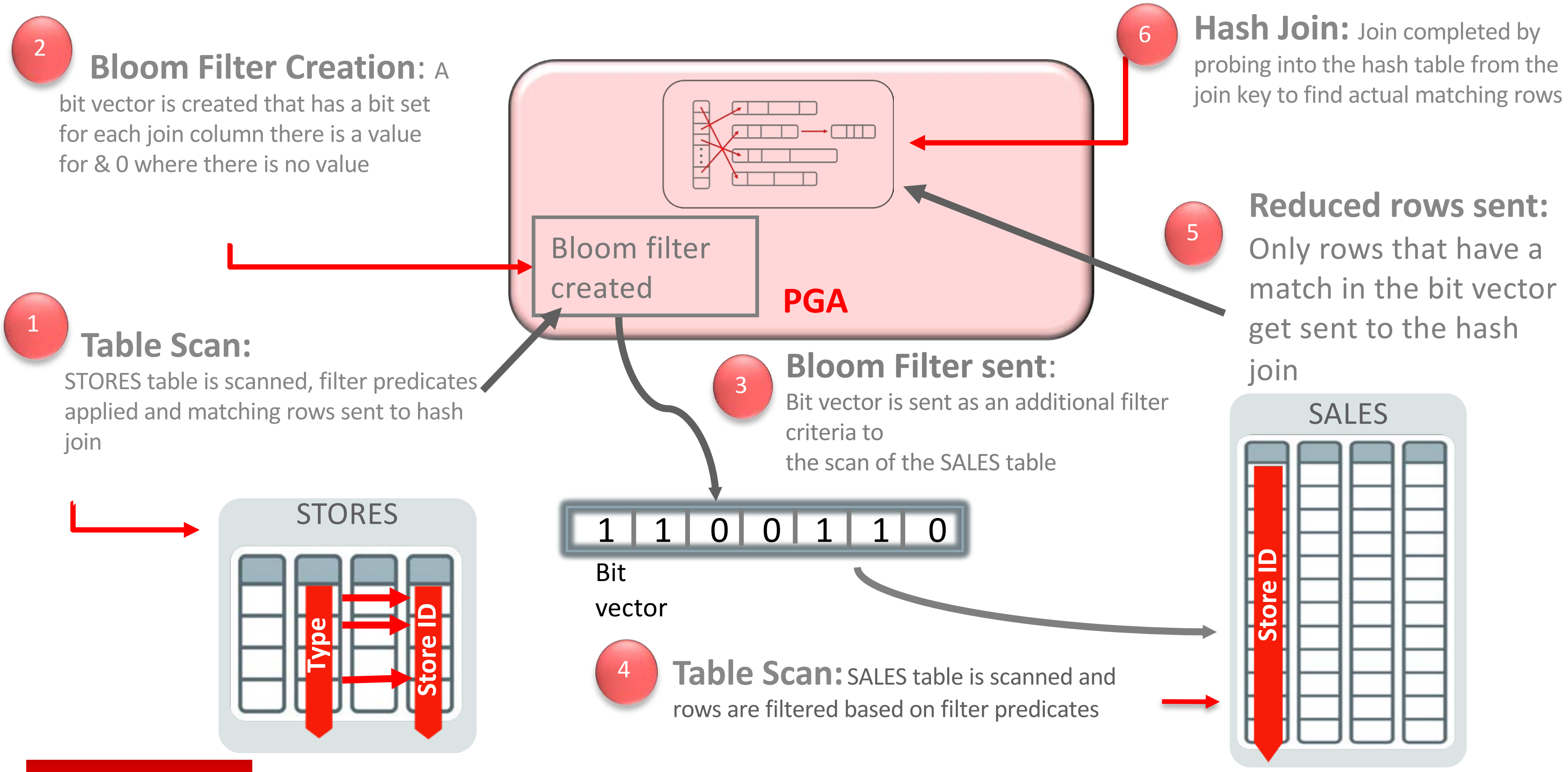


- Convert Hash Joins into 10X Faster Column Scans
- Search large table for values that match small table

In-Memory Aggregation

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

Database In-Memory Hash Join With Bloom Filters



1 **Table Scan:** STORES table is scanned, filter predicates applied and matching rows sent to hash join

2 **Bloom Filter Creation:** A bit vector is created that has a bit set for each join column there is a value for & 0 where there is no value

3 **Bloom Filter sent:** Bit vector is sent as an additional filter criteria to the scan of the SALES table

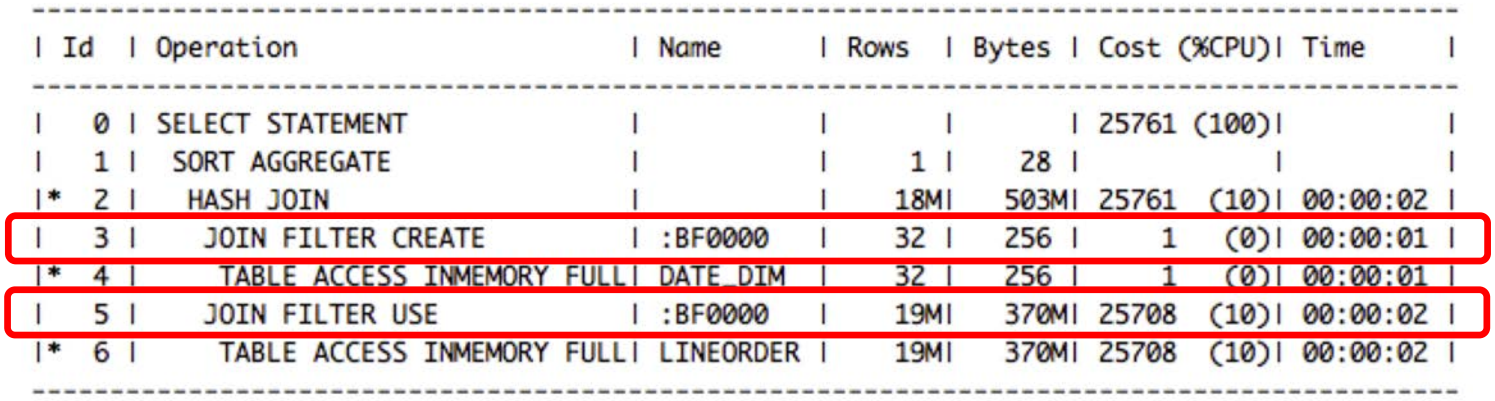
4 **Table Scan:** SALES table is scanned and rows are filtered based on filter predicates

5 **Reduced rows sent:** Only rows that have a match in the bit vector get sent to the hash join

6 **Hash Join:** Join completed by probing into the hash table from the join key to find actual matching rows

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



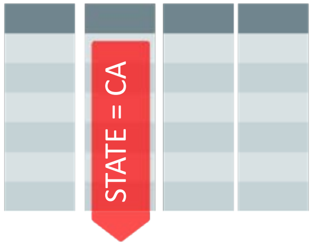
```
-----------------------------------------------------------------------------------------
| Id  | Operation                      | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |           |       |       | 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 |
-----------------------------------------------------------------------------------------
```

Database In-Memory: Real-time Analytics

Improves key aspects of analytic queries

Data Scans

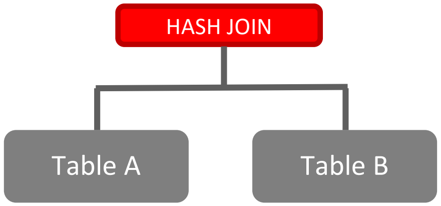
SALES



STATE = CA

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

Joins



HASH JOIN

Table A

Table B

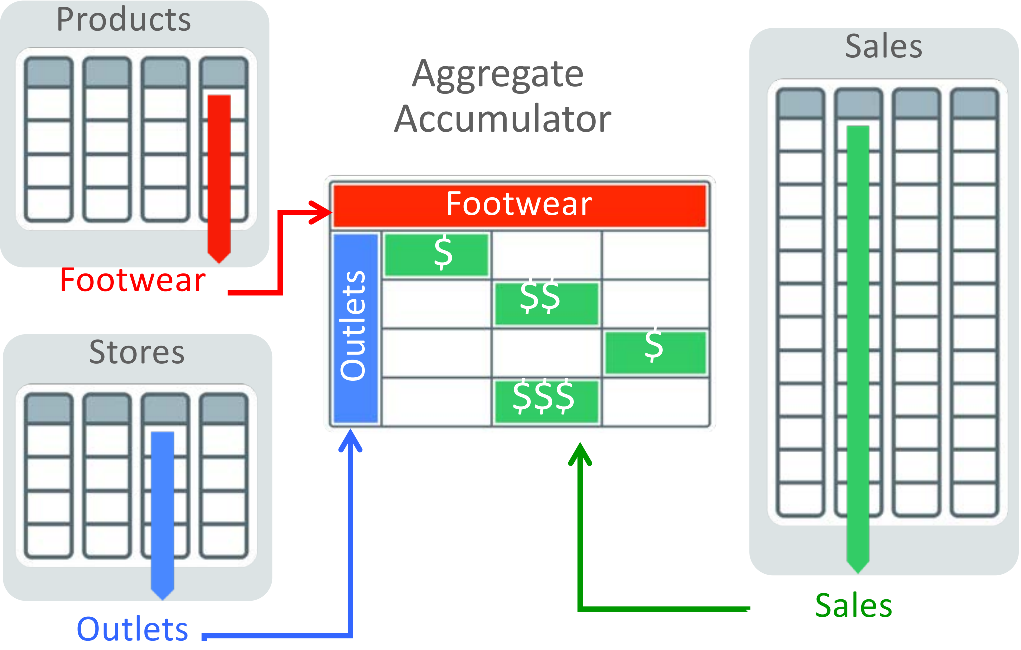
- Convert Hash Joins into 10X Faster Column Scans
- Search large table for values that match small table

In-Memory Aggregation

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

In-Memory Aggregation

Example: Report sales of footwear in outlet stores



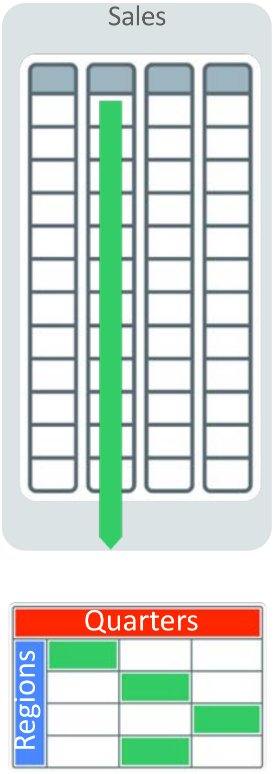
- Dynamically creates in-memory report outline (aggregate accumulator)
- Aggregation performed in-memory during fast fact scan
- Key vectors are used instead of Bloom filters
- Key vectors use dense grouping keys to map all key combinations

In-Memory Aggregation Advantages

- Also known by its execution plan operation: VECTOR GROUP BY
- Improved GROUP BY query performance with fewer CPU resources
  - 3-8 times query performance typical
- Fully dynamic aggregation, no need for indexes, summary tables or materialized views
- Fully leverages Database In-Memory features
- IMA will scale better than non-IMA plans for query complexity and concurrency

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 |
-----------------------------------------------------------------------------
```

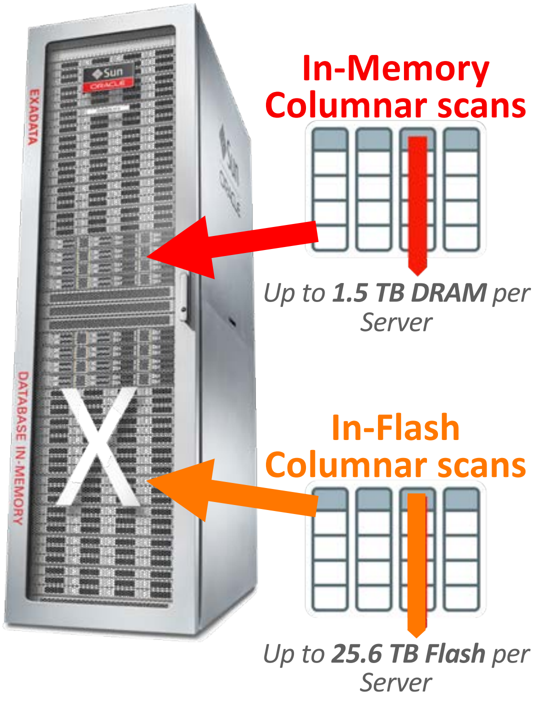


Scan, filter and aggregate

Improvements in Database In-Memory

In-Memory Analytics Extended into Exadata Flash 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 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*

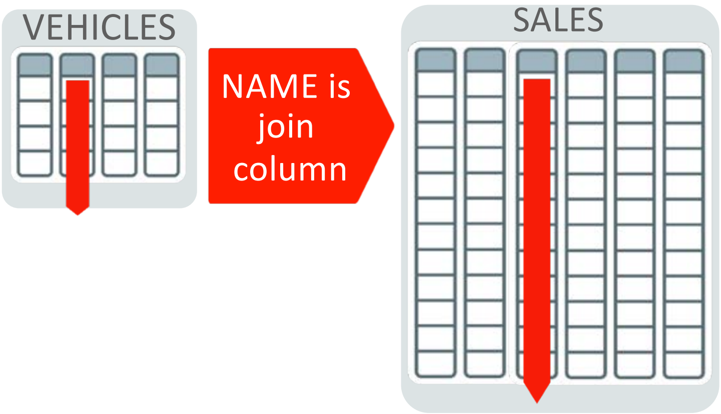
In-Flash Columnar scans

*Up to **25.6 TB Flash** per Server*

Join Groups: Faster Hash Joins

- Join columns in both tables are compressed using the same dictionary
- Joins occur on dictionary values rather than on data
  - Saves on decompression of data
  - Save on hashing the data

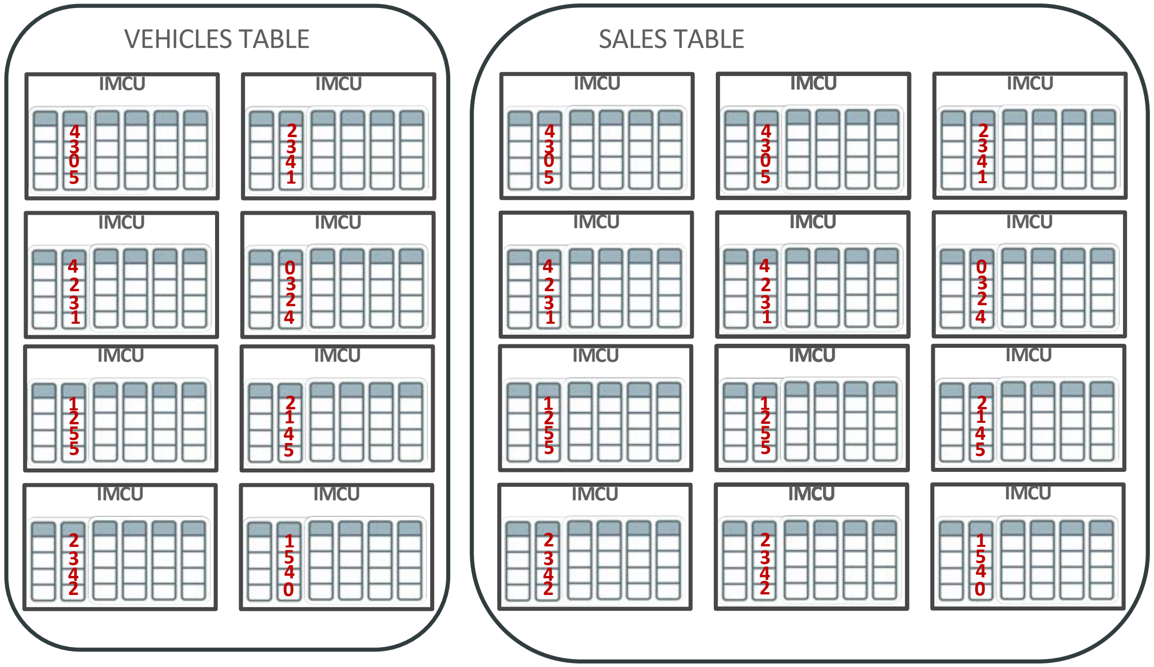
Example: Find sales price of each Vehicle



Join Group: Same Common Dictionary Used By Both Tables

Common Dictionary

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

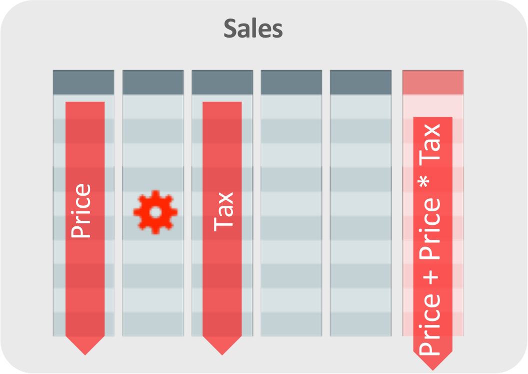


Common Dictionary created when first table is populated & used for join column in both tables – Must be defined with CREATE INMEMORY JOIN GROUP syntax

Analytics Performance: In-Memory Expressions

Example: Compute total sales price

Net = Price + Price * Tax



- Analytic queries contain complex expressions
  - Originally evaluated for every row
- In-Memory Expression
  - SQL expression computed & stored as additional inmemory columns
  - All In-Memory optimizations apply to expression columns (e.g. Vector processing, storage indexes)
- Reduce repeated evaluations
  - Save CPU by only calculating once
- **3-5x** faster complex queries

In-Memory Expressions : Manually Declaration

```
CREATE TABLE SALES (
      PRICE NUMBER,
      TAX NUMBER,
      …,
      NET AS (PRICE+PRICE*TAX)
      )
INMEMORY;

ALTER SYSTEM SET
inmemory_virtual_columns=ENABLE
;
```

1. Create virtual columns for desired in-memory expressions

1. Check value of the parameter `INMEMORY_VIRTUAL_COLUMNS`
   - Default is Manual
   - Consider switching to Enable

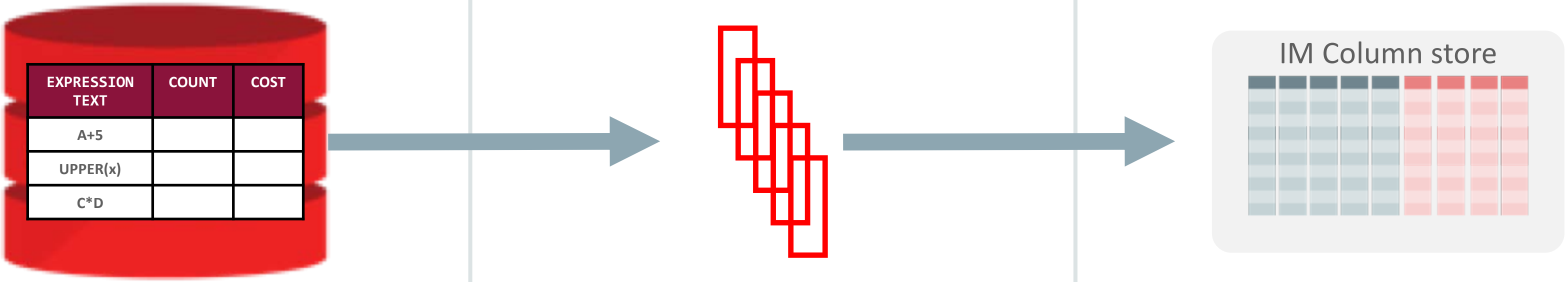
2. Populate tables into IM column store

In-Memory Expressions : Automatic Capture

Expression Statistic Store

Capture Top N Expressions

In-Memory Expressions automatically added to IM Column Store



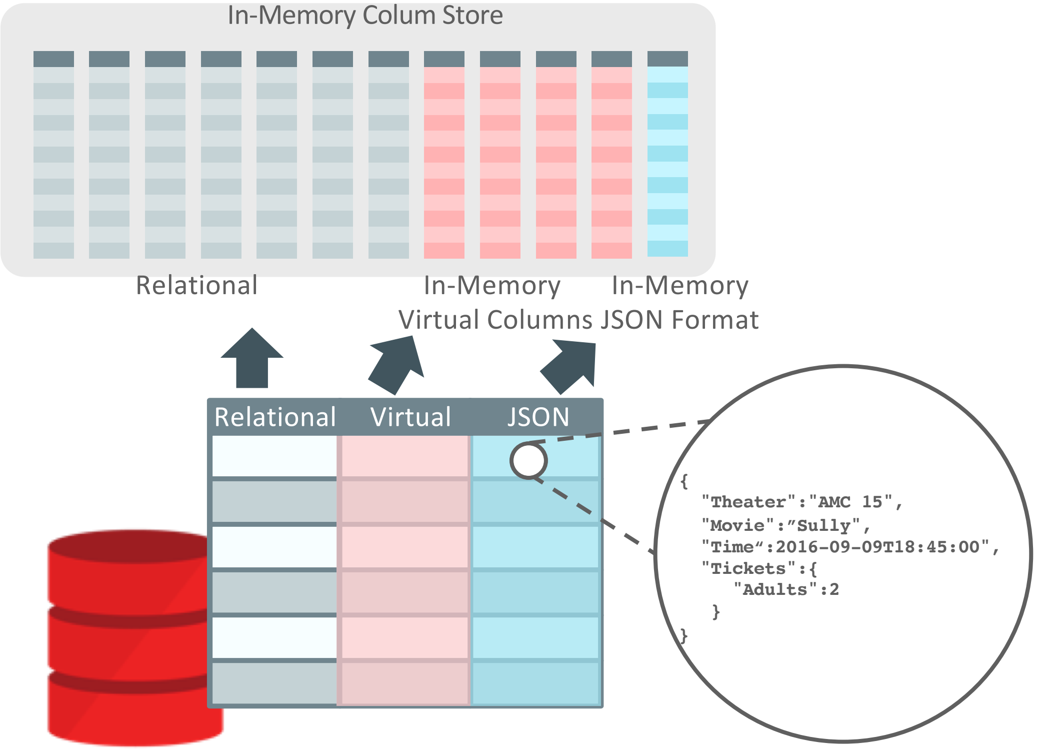
EXPRESSION TEXT	COUNT	COST
A+5		
UPPER(x)		
C*D		

- Expressions Statistics Store (ESS) constantly monitors workload
  - Records “hot” expressions based on frequency and cost during query parse

- Capture Top N Expressions Using new procedure IME_CAPTURE_EXPRESSIONS
  - Part of the DBMS_INMEMORY_ADMIN package

- Create hidden virtual columns and populates them into the IM Column store using new procedures IME_POPULATE_EXPRESSIONS
  - Part of the DBMS_INMEMORY_ADMIN package

Superfast / Multi-Model Analytics: In-Memory JSON



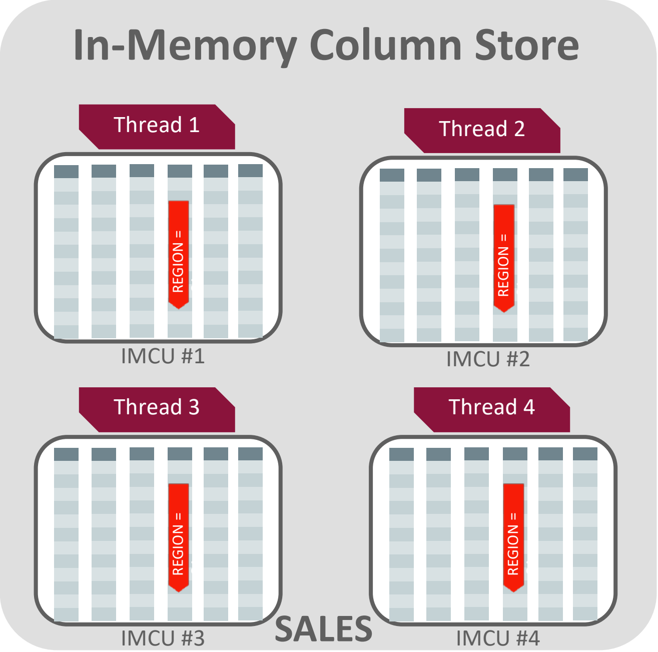
```
{
  "Theater":"AMC 15",
  "Movie":”Sully",
  "Time“:2016-09-09T18:45:00",
  "Tickets":{
     "Adults":2
   }
}
```

- Full JSON documents populated using an optimized binary format
- Additional expressions can be created on JSON columns (e.g. JSON_VALUE) & stored in column store
- Queries on JSON content or expressions automatically directed to In-Memory format
  - E.g. Find movies where movie.name contains “Jurassic”
- **20 - 60x** performance gains observed

In-Memory Dynamic Scans

Better Columnar Scan Performance

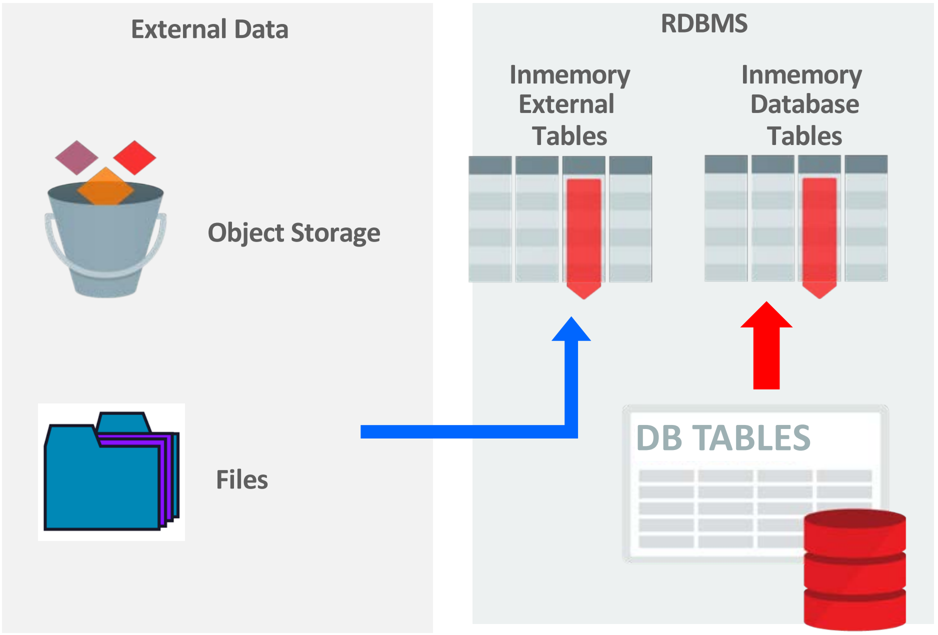
```
select SUM(total) from SALES where region = 'CA'
group by store_id
```



- Dynamic multi-threaded scan mechanism
  - Multiple threads per scan process
  - Each thread scans 1 IMCU at a time
  - Scan Multiple IMCUs in parallel in a single scan process
- No user intervention needed
  - #threads controlled by Resource Manager
  - Goal: Fully utilize available CPU cores
- Supplements Parallel Query
  - Each Parallel Query slave can have multiple threads
  - Up to 2x performance gains observed

In-Memory For External Tables

Fast Analytics on External Data

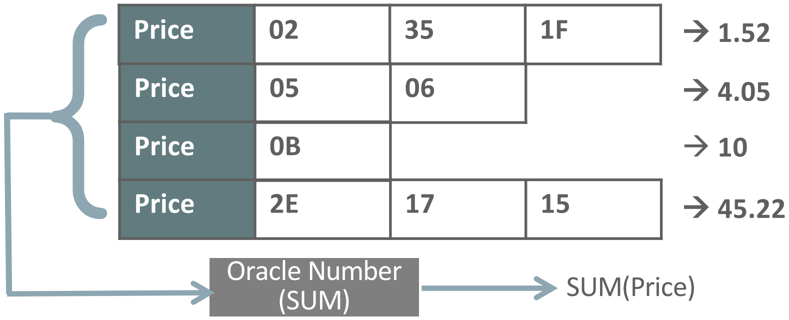


- External Tables allow transparent access to external file data
  - In-Memory For External Tables will allow fast analytics on external data without having to import it into the database
- All In-Memory Optimizations apply
  - e.g. vector processing, JSON
- Up to 100X faster

In-Memory Optimized Arithmetic

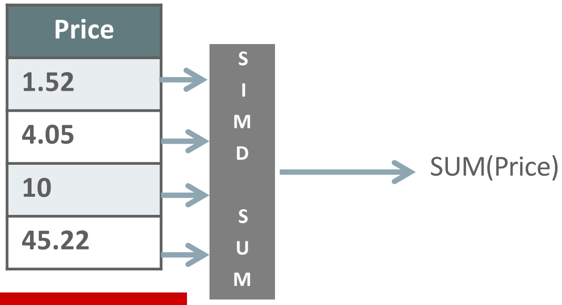
Blazing Fast Numeric operation

Slower Row-by-Row Oracle Number Processing



Price	02	35	1F	→ 1.52
Price	05	06		→ 4.05
Price	0B			→ 10
Price	2E	17	15	→ 45.22

FASTER SIMD Vector Processing of In-Memory Numbers

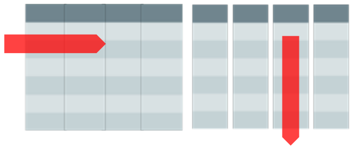


Price
1.52
4.05
10
45.22

- **New** In-Memory optimized format for NUMBER columns
  - Instead of software-implemented, variable-width ORACLE NUMBERs
  - Enabled using new parameter **inmemory_optimized_arithmetic**
- SIMD Vector Processing on optimized inmemory number format
- Aggregation and Arithmetic operators can improve **up to 40X**

Further Improvements in Database In-Memory

Mixed Workload

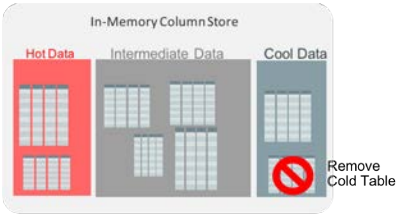


- Active Data Guard Support

Automation

- Policy-based Movement Between Storage & Memory
- IM FastStart
- IM Column Store Re-sizing

Automatic In-Memory



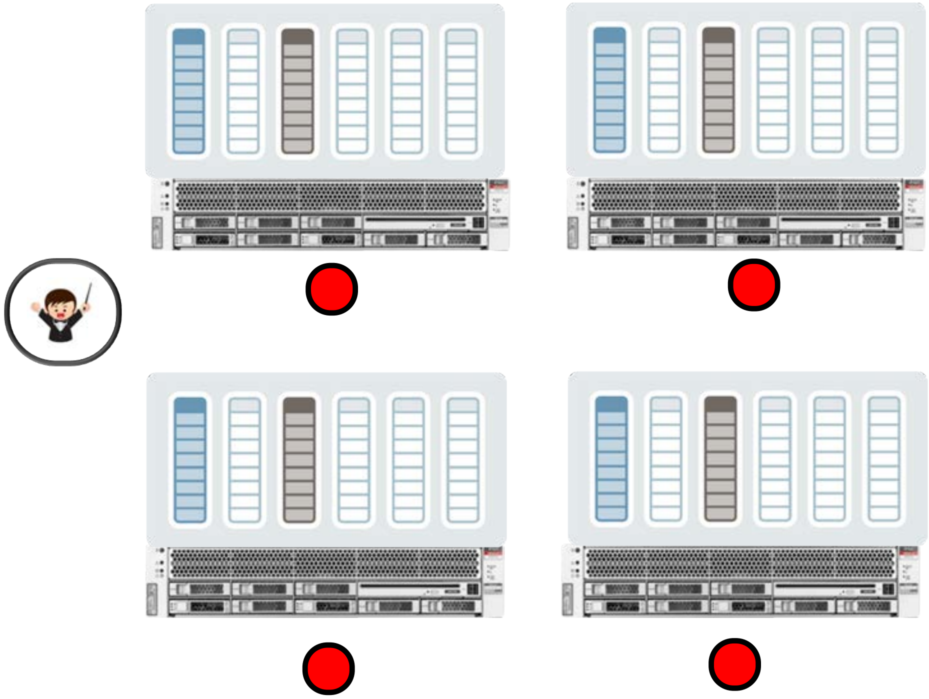
Automatic Data Movement Between Storage & Memory

Database In-Memory Scale Out

Database In-Memory: Scale Out

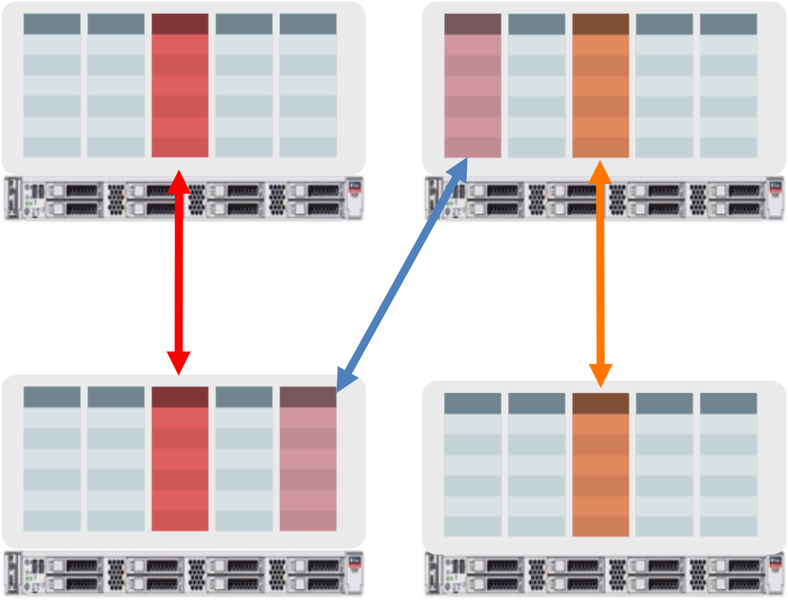
- Shared nothing architecture
- IMCUs not shipped across interconnect
- Serial queries will only access data from the IM column store on its node
- Rest of the data will come from row-store

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)

Database In-Memory: Unique Fault Tolerance

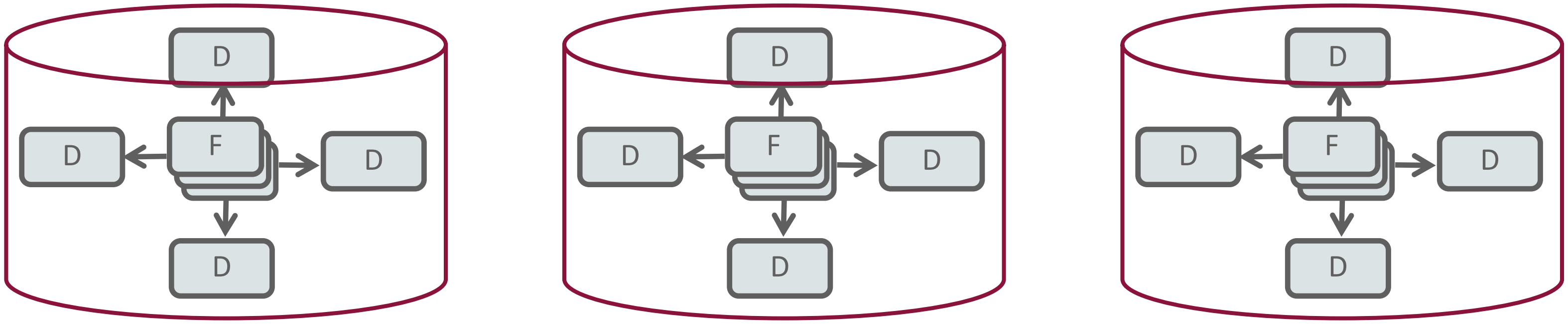


Only Available on Engineered Systems

- Similar to storage mirroring
- Duplicate in-memory columns on another node
  - Enabled per table/partition
    - e.g. only recent data
  - Application transparent
- Downtime eliminated by using duplicate after failure

Example: Duplicate Strategy For a Star Schema

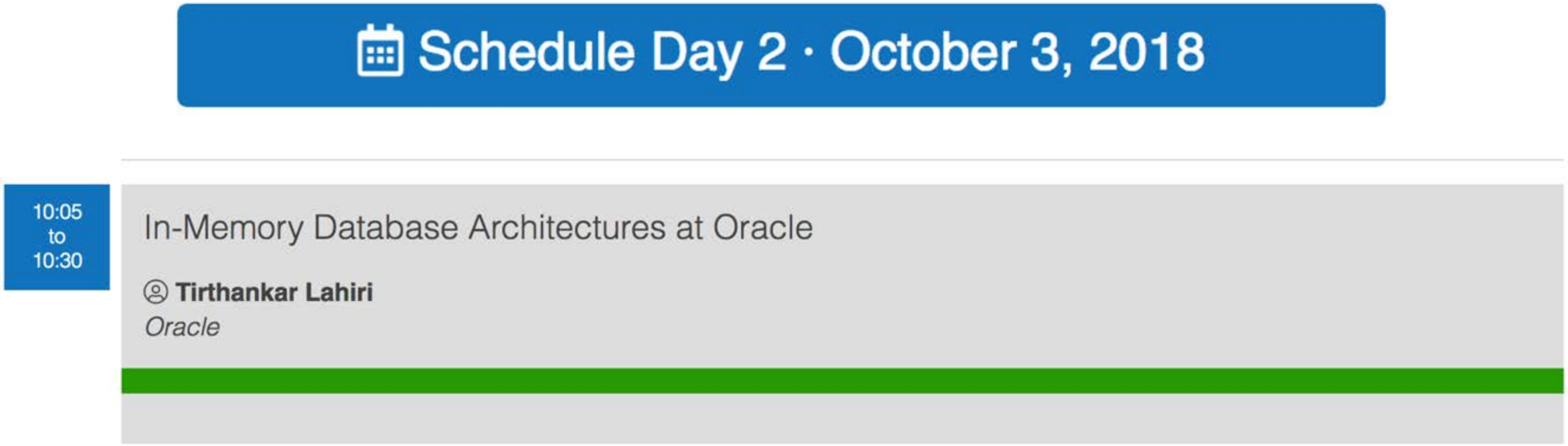
- Fact tables are distributed by partition
- Dimension tables are duplicated (DUPLICATE ALL)
- Co-locates joins between the distributed fact table partitions and the dimension tables



Only Available on Engineered Systems

Where can I get more information

In-Memory Computing SUMMIT | NORTH AMERICA 2018



Schedule Day 2 · October 3, 2018

10:05 to 10:30

In-Memory Database Architectures at Oracle

Tirthankar Lahiri

Oracle

Additional Resources

Join the Conversation

- https://twitter.com/db_inmemory
- https://twitter.com/TheInMemoryGuy
- https://blogs.oracle.com/In-Memory/
- https://www.facebook.com/OracleDatabase
- http://www.oracle.com/goto/dbim.html

White Papers (otn.com)

- Oracle Database In-Memory White Paper
- Oracle Database Implementation and Usage White Paper
- Oracle Database In-Memory Aggregation Paper
- When to use Oracle Database In-Memory
- Oracle Database In-Memory Advisor

Videos

- Oracle Database In-Memory YouTube Channel
- oracle.com

  Powering the Real-Time Enterprise
  oracle.com/us/corporate/events/dbim/index.html
  Real-Time Analytics Demo
- YouTube - Juan Loaiza: DBIM: What's new in 12.2

Additional Questions

- In-Memory blog: blogs.oracle.com/In-Memory
- My email: andy.rivenes@oracle.com