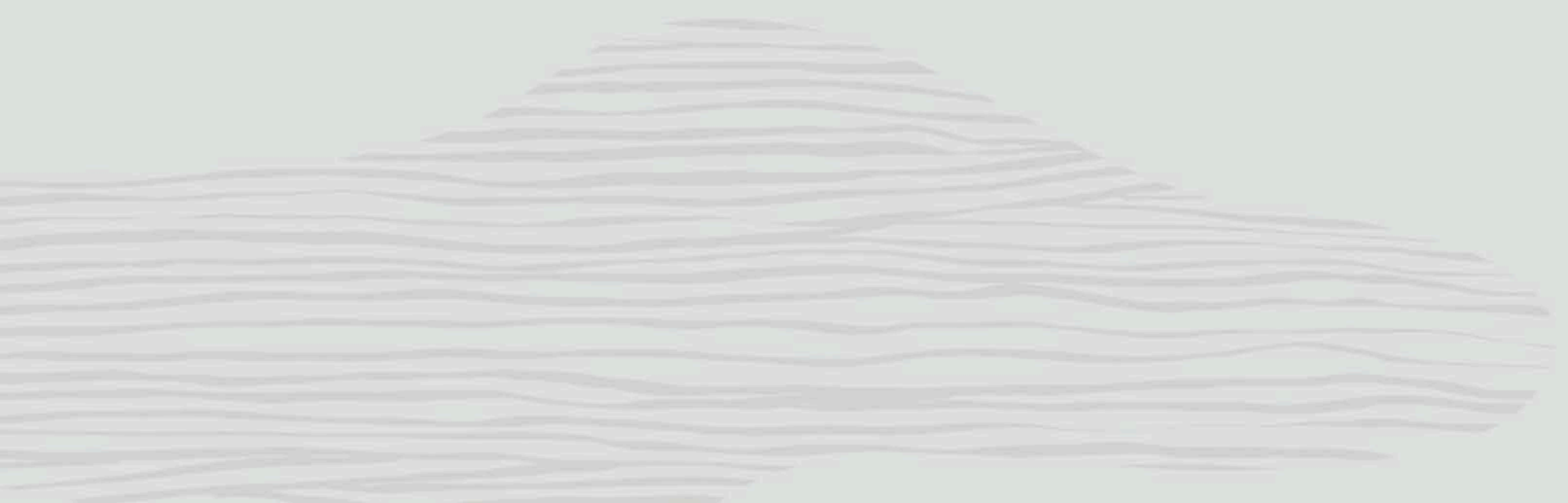


This is a subset of a presentation created by Oracle Corp. (2019)

Disclaimer: There is some academic value to the material presented. However, we are not in any way endorsing Oracle Corp. or Oracle Corp. products.

A decorative graphic consisting of several horizontal, wavy lines in a light gray color, located in the bottom left corner of the slide.

ORACLE

# Top-5 Innovations of Oracle's Database In-Memory

CMU, 2019

Shasank Chavan

Vice President, In-Memory Database Technologies





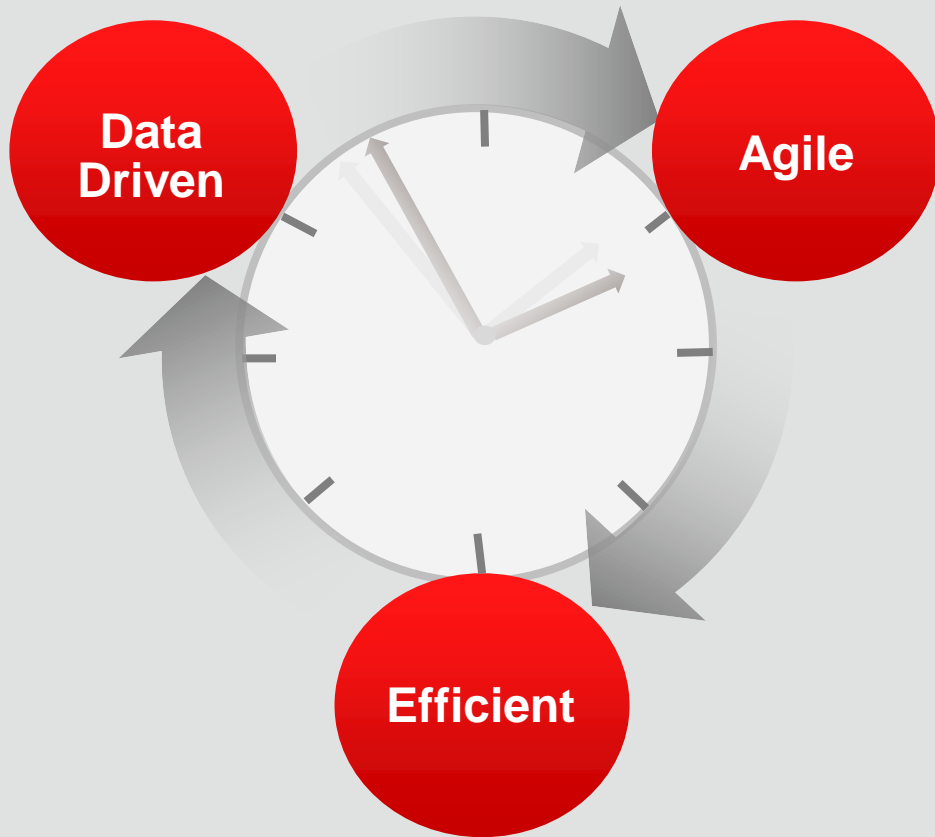
## Safe Harbor

---

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, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Statements in this presentation relating to Oracle's future plans, expectations, beliefs, intentions and prospects are "forward-looking statements" and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle's Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading "Risk Factors." These filings are available on the SEC's website or on Oracle's website at <http://www.oracle.com/investor>. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.

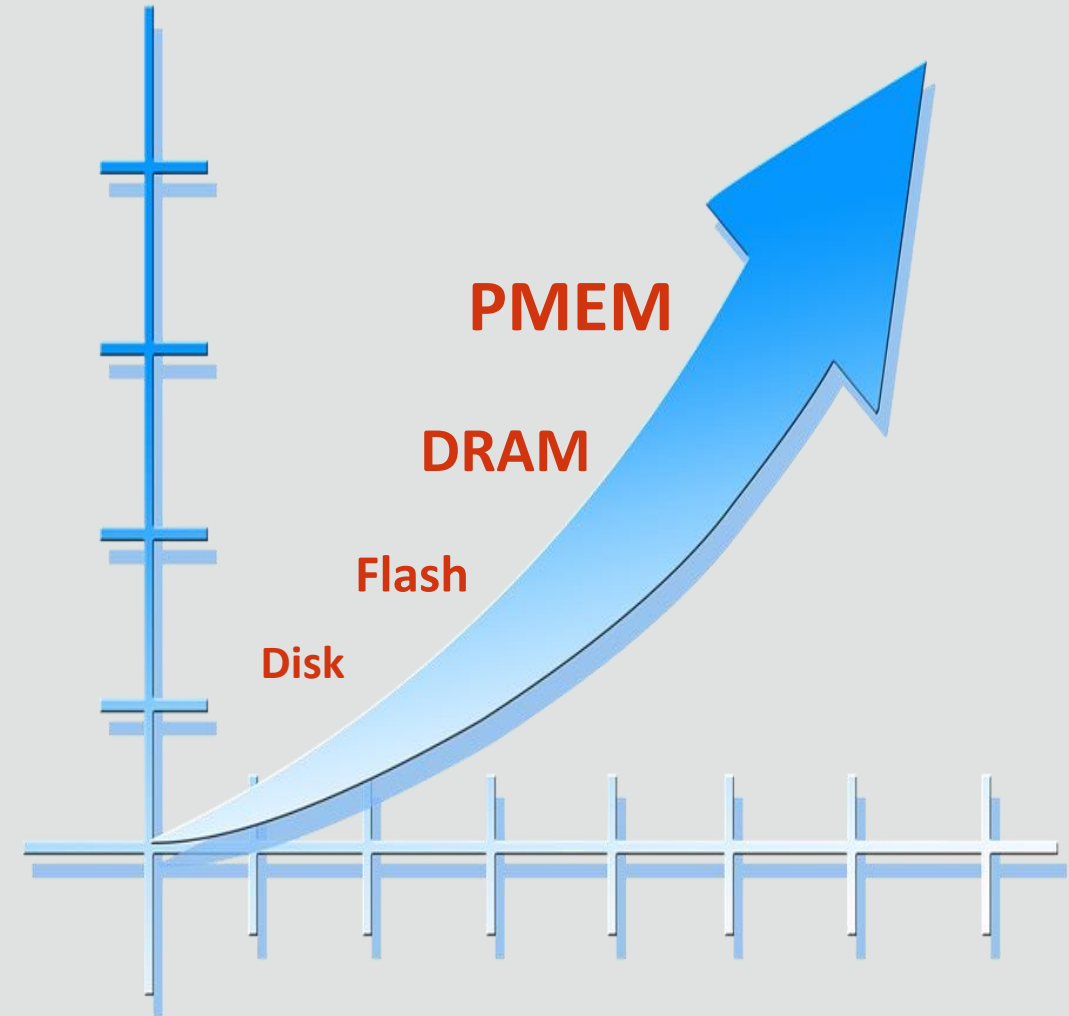
# In-Memory Now | Real-Time Enterprises

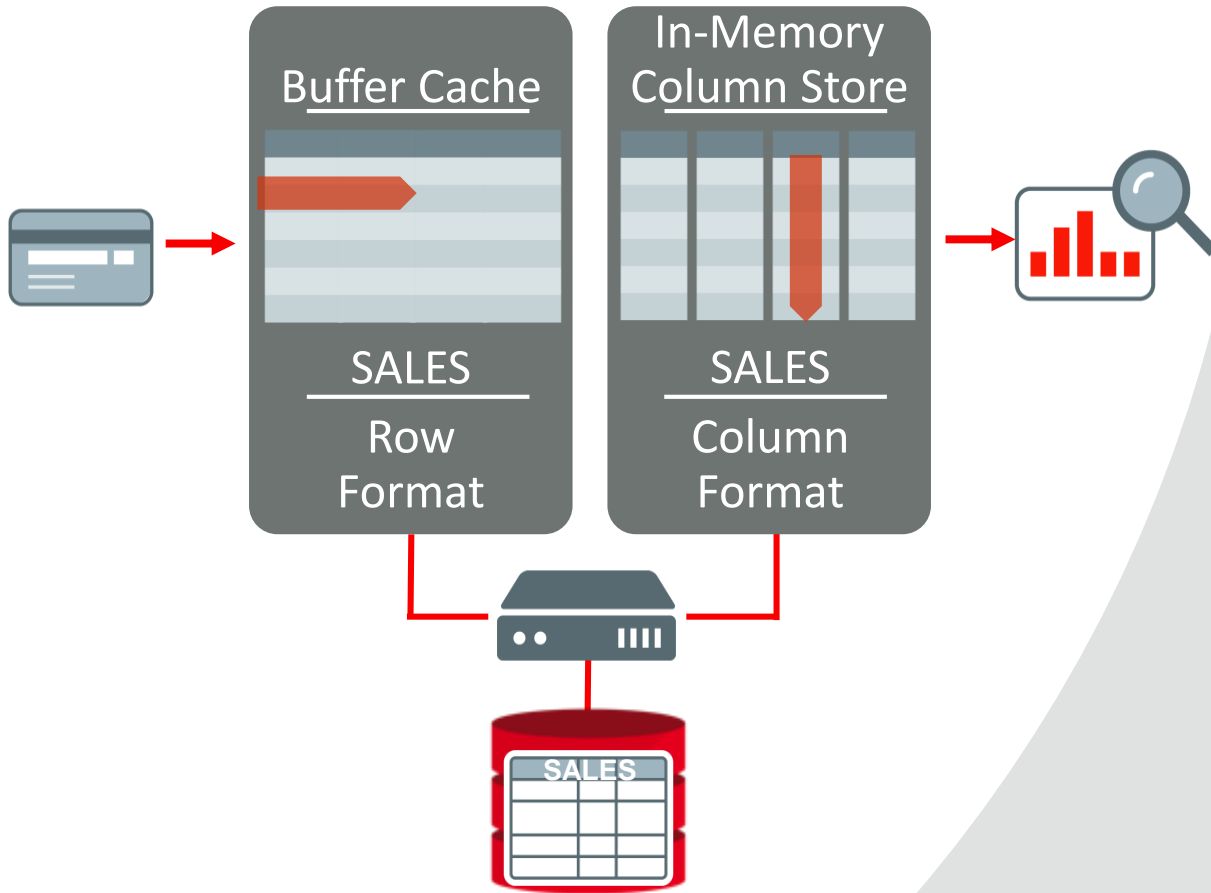


- **Insurance companies** improve portfolios and reduce cost with real-time analytics for pricing
- **Retailers** use location-based analytics to automate sending personalized mobile coupons to customers
- **Manufacturing Processes** use real-time analytics to monitor production quality and adjust assembly parameters
- **Financial Services** perform risk/fraud analysis across channels in real-time, not after the event occurs
- **Telecom and Broadband** vendors use real-time congestion metrics to optimize their networks

# In-Memory Now | Hardware Trends

- **Larger, Cheaper Memory** (DRAM, PMEM)
- **Larger CPU Caches** (e.g. 32MB Shared L3 Cache)
- **Larger Multi-Core Processors** (24 cores w/ Intel)
- **Larger SIMD Vector Processing Units** (e.g. AVX-512)
- **Faster Networks** (100Gb/s RoCE vs 40Gb/s Infiniband)
- **NUMA Architectures** (Local Memory vs Remote)
- **Persistent Memory** (Availability, Capacity, Speed)





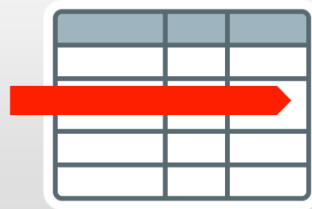
# Oracle Database In-Memory

*Background*

# In-Memory Row Format: **Slower for Analytics**

Row

SALES




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

Buffer Cache

COL1	COL2	COL3	COL4
X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X

Row Format

SELECT **COL4** FROM MYTABLE;



RESULT

X X X X X

Needs to skip over unneeded data

# In-Memory Columnar Format: **Faster for Analytics**

Column



▪ **Analytics** run faster on column format

- Example : Report on sales totals by region
- Fast accessing few columns, many rows

IM Column Store

COL1	COL2	COL3	COL4
X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X

Column Format

```
SELECT COL4 FROM MYTABLE;
```



RESULT



# In-Memory Columnar Format: **Faster for Analytics**

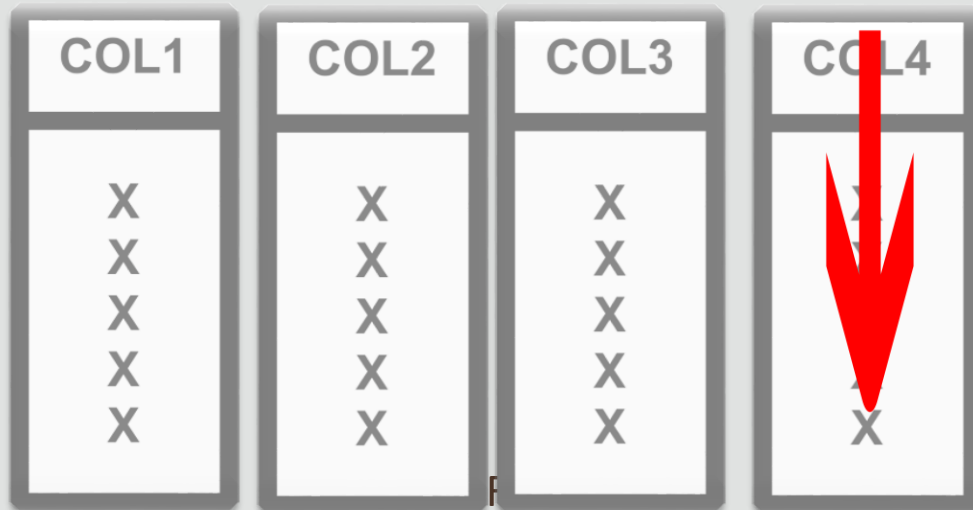
Column



▪ **Analytics** run faster on column format

- Example : Report on sales totals by region
- Fast accessing few columns, many rows

IM Column Store



SELECT **COL4** FROM MYTABLE;



RESULT

X X X X

Scans only the data required by the query

# Background | Row vs. Column Databases

## Row



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

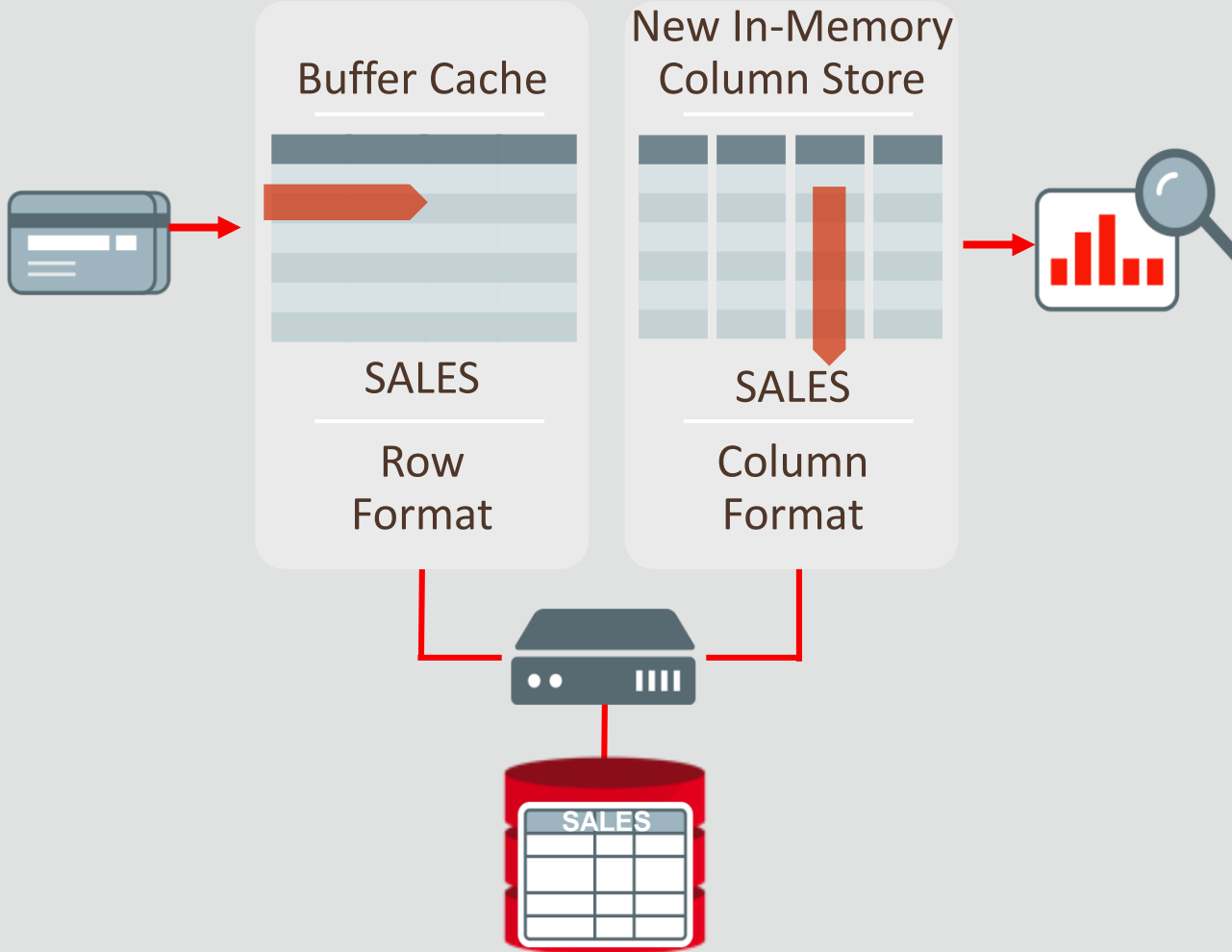
## Column



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

Choose One Format and Suffer the Consequences / Tradeoffs

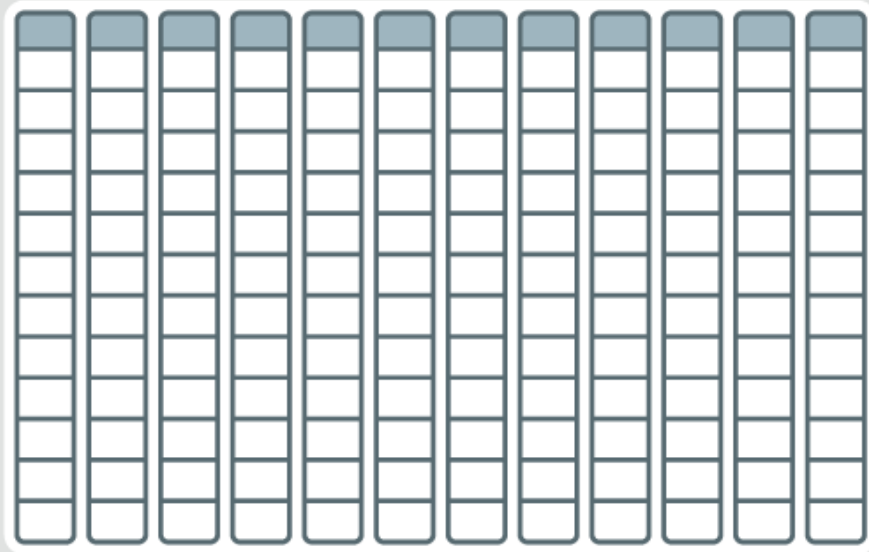
# Database In-Memory | Architecture



- **Both** row and column format for same table
  - Simultaneously active and consistent
- OLTP uses existing row format
- Analytics uses In-Memory Column format
  - **Seamlessly** built into Oracle Database
  - All enterprise features work
    - RAC, Dataguard, Flashback, etc.

# In-Memory Columnar Format

## Pure In-Memory Columnar



SALES

SALES		



- Pure in-memory column format
- Fast In-Memory Maintenance with OLTP
- No Changes to Disk Format
- Available on All Platforms
- Enabled at tablespace, table, partition, sub-partition, and even column level
- Total memory area controlled by **inmemory\_size** parameter



# In-Memory Columnar Format | Deep Dive

## In-Memory Compression Unit

### Column CUs

ROWID	EmpID	Name	Dept	Salary

TABLE

## In-Memory Compression Unit (IMCU)

- Unit of column store allocation  
Spans large number of rows (e.g. 0.5 million) on one or more table extents
- Each column stored as **Column Compression Unit** (column CU)

## Multiple MEMCOMPRESS levels:

FOR QUERY – fastest queries

FOR CAPACITY – best compression

Extent #13  
Blocks 20 to 120

Extent #14  
Blocks 82 to 182

Extent #15  
Blocks 201 to 301

# In-Memory Columnar Format | Compression

## Uncompressed Data

CAT  
CAT  
FISH  
FISH  
HORSE  
HORSE  
HORSE  
DOG  
DOG  
CAT  
CAT  
FISH  
HORSE  
HORSE  
DOG  
DOG

# In-Memory Columnar Format | Compression

## Uncompressed Data

CAT  
CAT  
FISH  
FISH  
HORSE  
HORSE  
HORSE  
DOG  
DOG  
CAT  
CAT  
FISH  
HORSE  
HORSE  
DOG  
DOG



## Dictionary Compressed

CAT     0  
DOG     1  
FISH    2  
HORSE   3  
  
00, 00, 10, 10  
11, 11, 11, 01  
01, 00, 00, 10  
11, 11, 01, 01



# In-Memory Columnar Format | Compression

## Uncompressed Data

CAT  
CAT  
FISH  
FISH  
HORSE  
HORSE  
HORSE  
DOG  
DOG  
CAT  
CAT  
FISH  
HORSE  
HORSE  
DOG  
DOG



## Dictionary Compressed

CAT     0  
DOG     1  
FISH    2  
HORSE   3  
  
00, 00, 10, 10  
11, 11, 11, 01  
01, 00, 00, 10  
11, 11, 01, 01



## Dict + RLE Compressed

CAT     0  
DOG     1  
FISH    2  
HORSE   3

00, 10, 11, 01  
01, 00, 10, 11  
01

Runs:  
2,2,3,2,2,1,2,2





# In-Memory Columnar Format | Compression

## Uncompressed Data

CAT  
CAT  
FISH  
FISH  
HORSE  
HORSE  
HORSE  
DOG  
DOG  
CAT  
CAT  
FISH  
HORSE  
HORSE  
DOG  
DOG

## Dictionary Compressed

CAT 0  
DOG 1  
FISH 2  
HORSE 3  
  
00, 00, 10, 10  
11, 11, 11, 01  
01, 00, 00, 10  
11, 11, 01, 01

## Dict + RLE Compressed

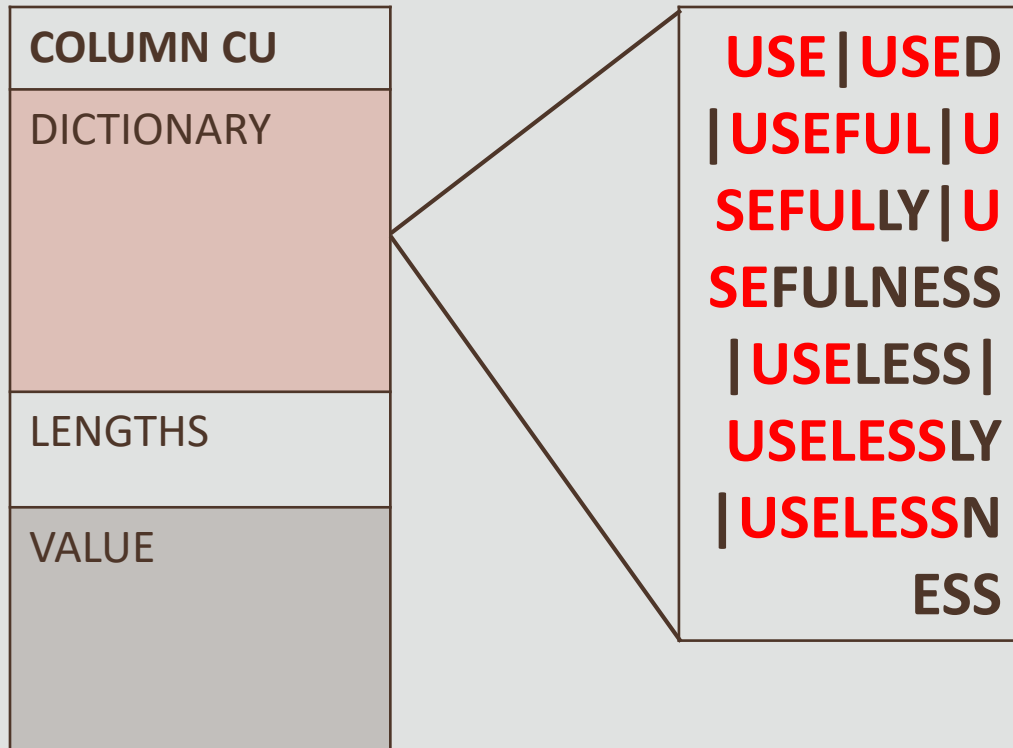
CAT 0  
DOG 1  
FISH 2  
HORSE 3  
  
00, 10, 11, 01  
01, 00, 10, 11  
01  
  
Runs:  
2,2,3,2,2,1,2,2

## OZIP Compressed

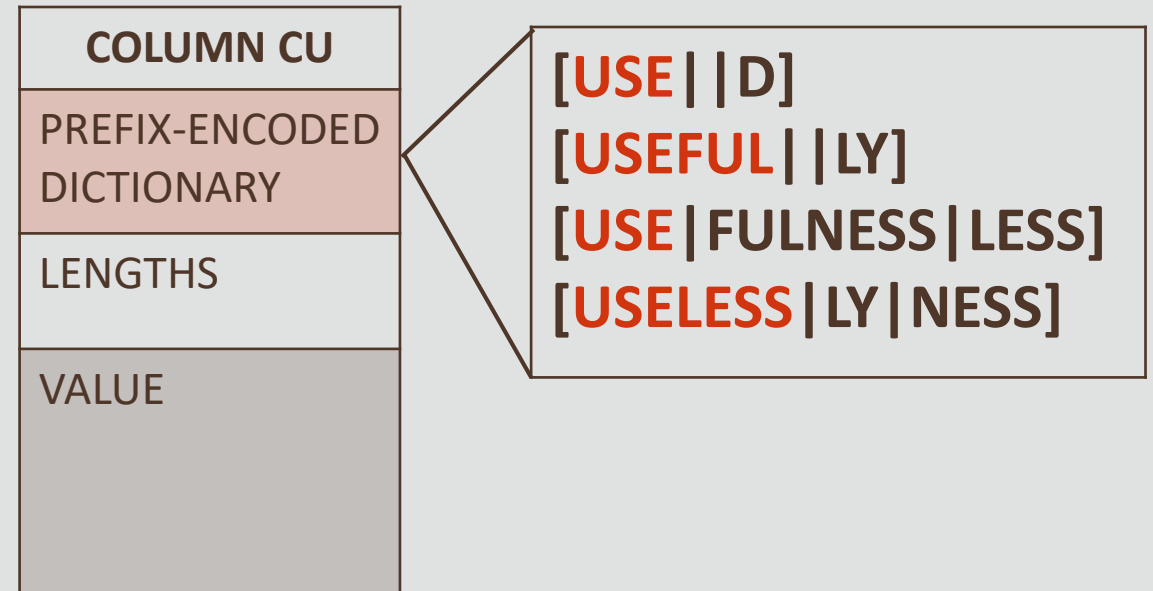
0: 00101101  
1: 01  
  
010

# In-Memory Columnar Format | Compression

## No Compression

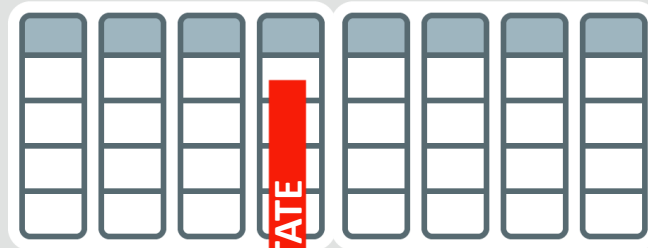


## Prefix Compression



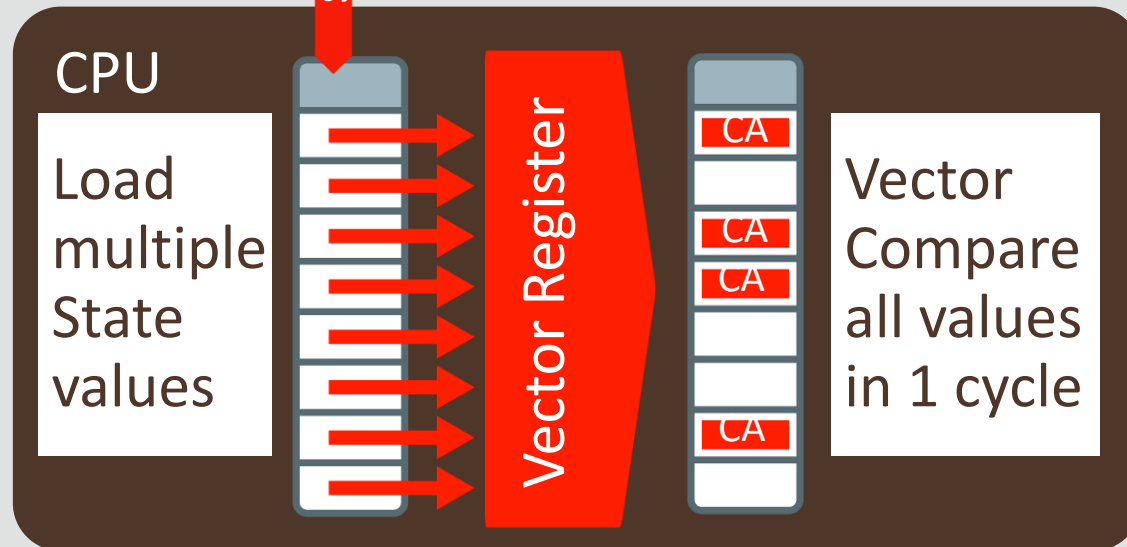
# In-Memory Enables **SIMD** Vector Processing

## Memory



**Example:**  
Find sales in  
State of California

- Column format benefit: Need to access only needed columns
- Process multiple values with a single SIMD Vector Instruction
- **Billions of rows/sec** scan rate per CPU core
  - Row format is millions/sec



**> 100x Faster**

# Improves All Aspects of Analytic Workloads...

## Scans



- **Billions** of Rows per second scans

## Joins



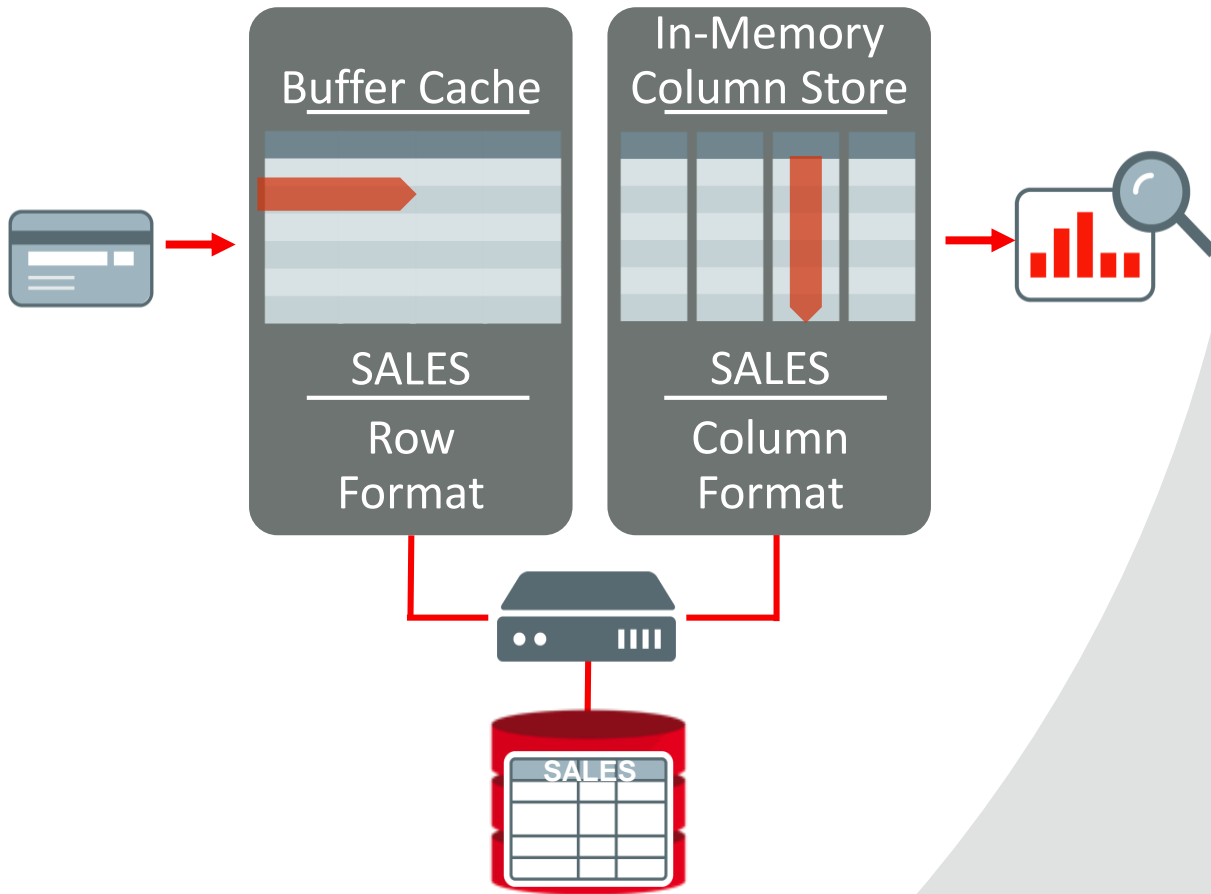
- Convert slower joins into **10x faster** filtered column scans

## Reporting



- Run reports with aggregations and joins **10x faster**



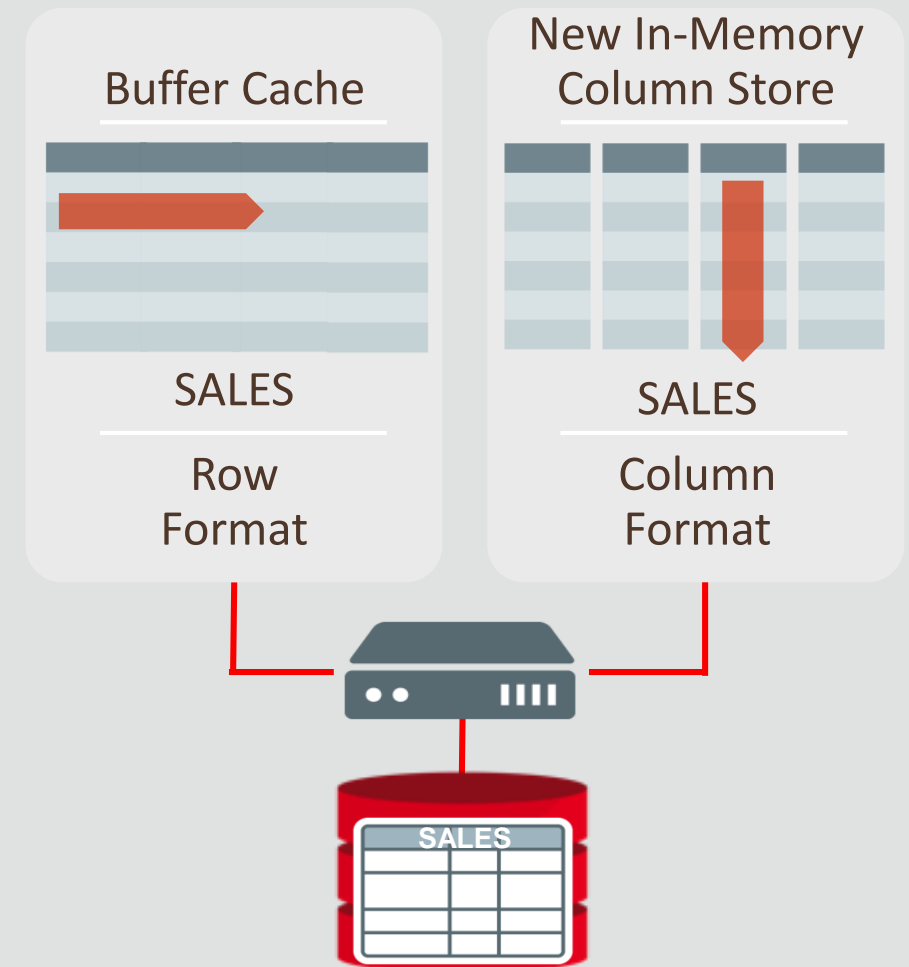


# #1 Dual-Format Architecture

*Fast Mixed Workloads, Faster  
Analytics*

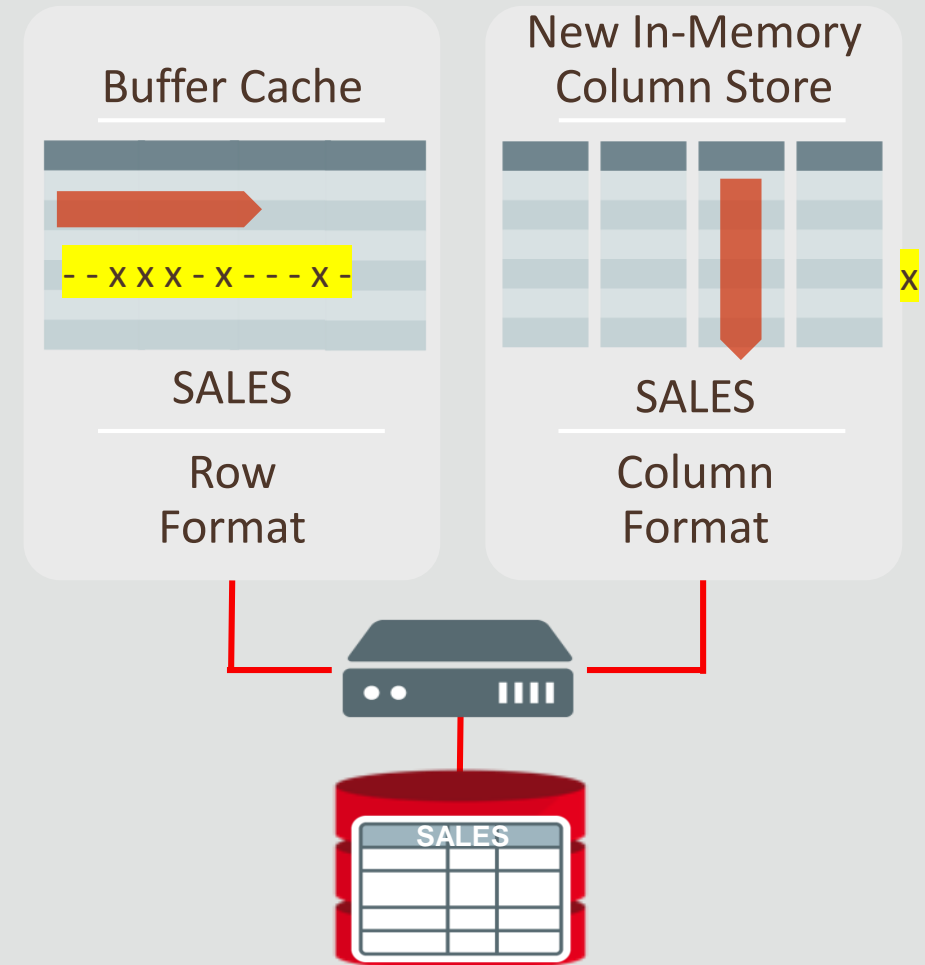
# In-Memory: Dual-Format Architecture

- Dual-Format Architecture enables fast Mixed Workloads and faster Analytics
- Fast In-Memory DML because invalid row is logically removed from column store (just set a bit)



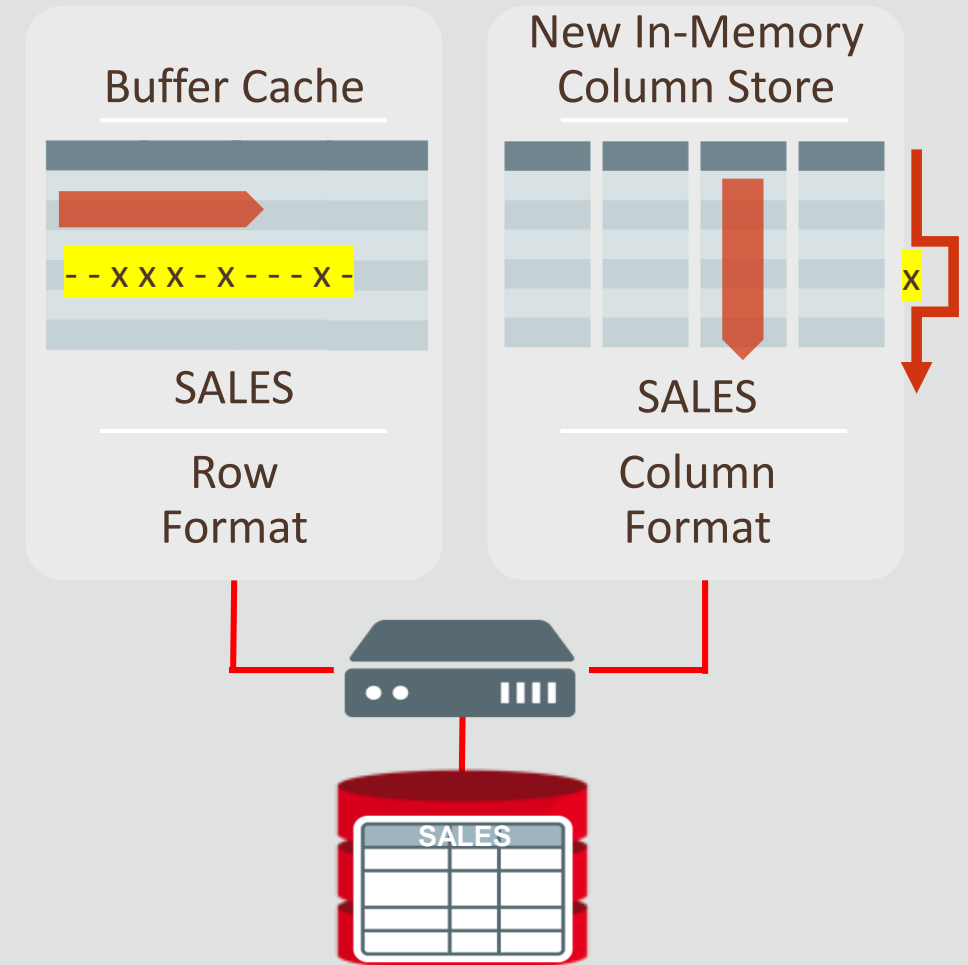
# In-Memory: Dual-Format Architecture

- Dual-Format Architecture enables fast Mixed Workloads and faster Analytics
- Fast In-Memory DML because invalid row is logically removed from column store (just set a bit)



# In-Memory: Dual-Format Architecture

- Dual-Format Architecture enables fast Mixed Workloads and faster Analytics
- Fast In-Memory DML because invalid row is logically removed from column store (just set a bit)
- Analytic query will ignore invalid rows in column store, and just vector process valid rows. Invalid rows are then processed.
  - IMCUs not covering invalid rows are unaffected.
- Mixed workload performance can suffer if the number of invalid rows accumulates in IMCUs
  - **Fast repopulation techniques save the day!**





# In-Memory: Fast Background Repopulation

*Continuous intelligence* to track how dirty an IMCU is, how frequently it is scanned, and when to take action to refresh/repopulate it.

## Double Buffering



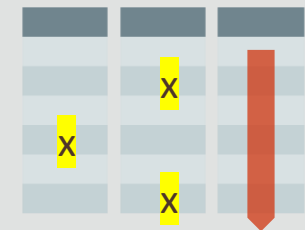
Old IMCU stays online until New IMCU is built. Then A *switcher* happens once New IMCU is ready.

## Incremental Repopulation



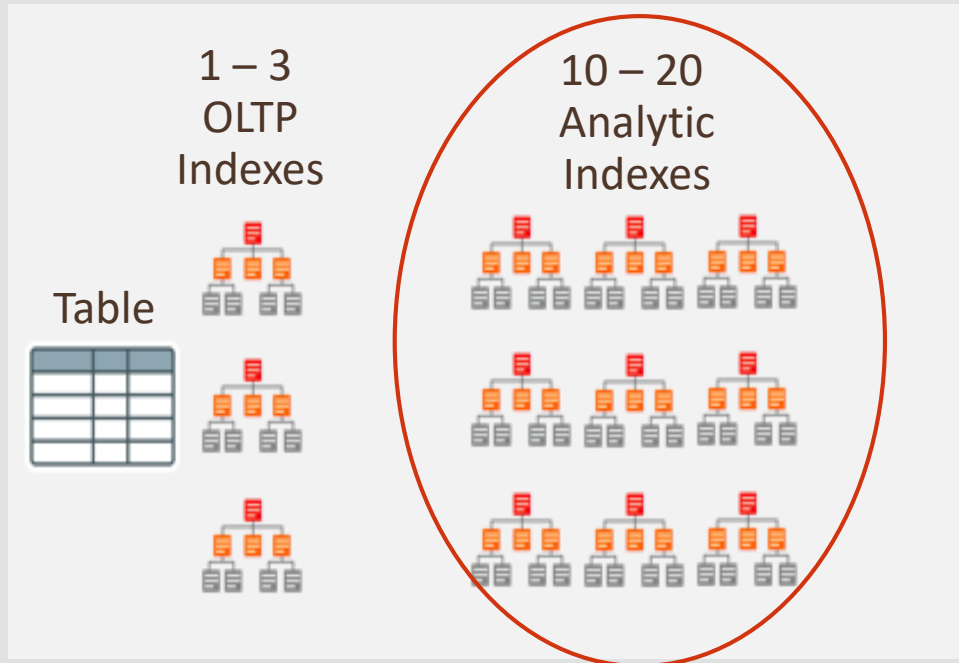
Build new columns in new IMCU using meta-data present in the old IMCU, allowing quick formatting

## Column-Level Invalidations



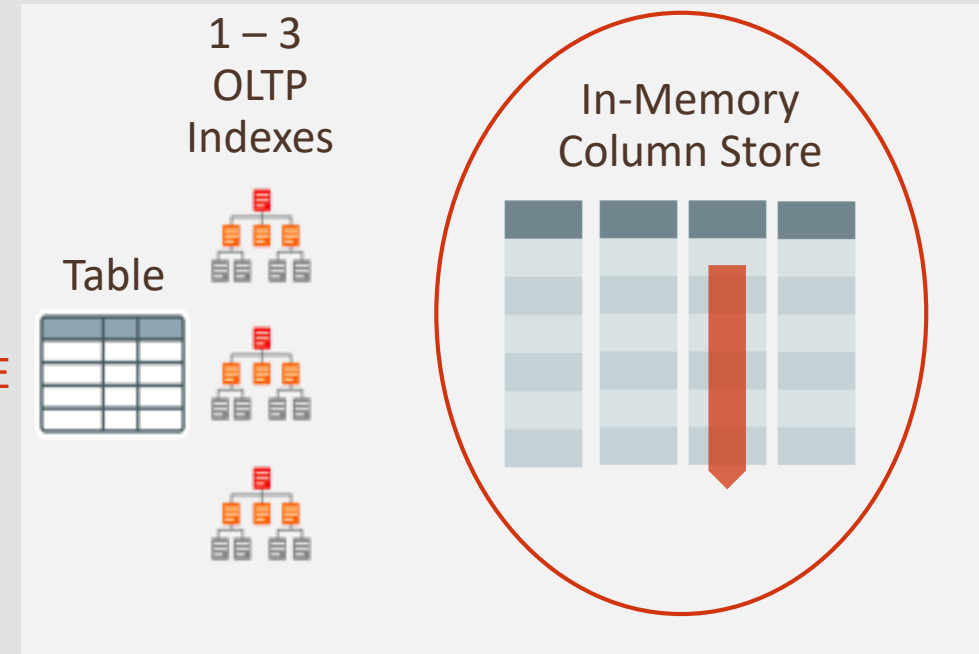
Column-Level Invalidations tracked to still enable IM scans

# Accelerates Mixed Workloads (Hybrid OLTP)



- Inserting one row into a table requires updating 10-20 analytic indexes: **Slow!**
- Fast analytics only on indexed columns
- Analytic indexes **increase** database size

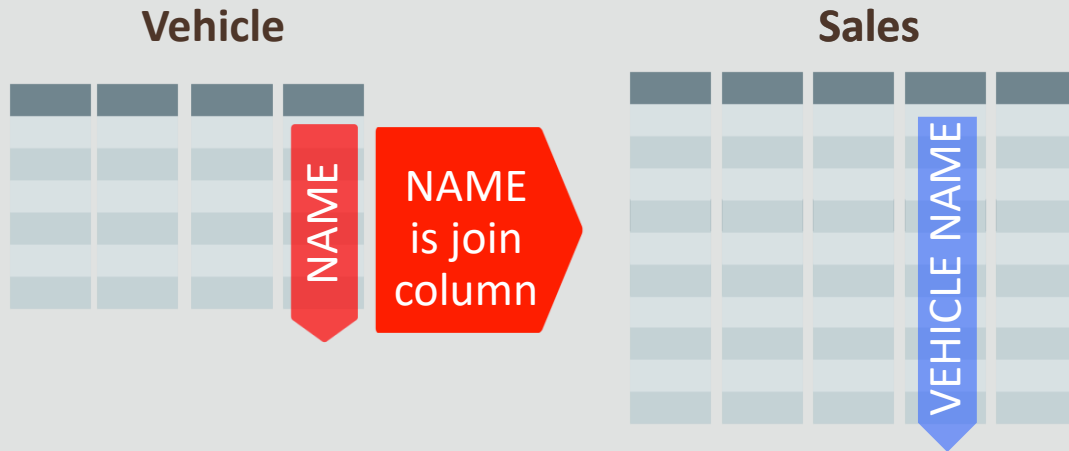
REPLACE



- Column Store not persistent so updates are: **Fast!**
- Fast analytics on any columns
- No analytic indexes: **Reduces** database size

# Faster Analytics | In-Memory Joins

**Example:** Find sales price of each Vehicle



```
CREATE INMEMORY JOIN GROUP V_name_jg  
(VEHICLES (NAME) , SALES (NAME) ) ;
```

- Joins are a significant component of analytic queries
  - Joins on inmemory tables are 10x faster already
- **Join Groups** enables faster joins
  - Specifies columns used to join tables
  - Join columns compressed using exact same encoding scheme.
  - This enables a faster array-based indexing join to be used instead of expensive hash join.
- Enables **2-3x speedup** over already fast inmemory joins

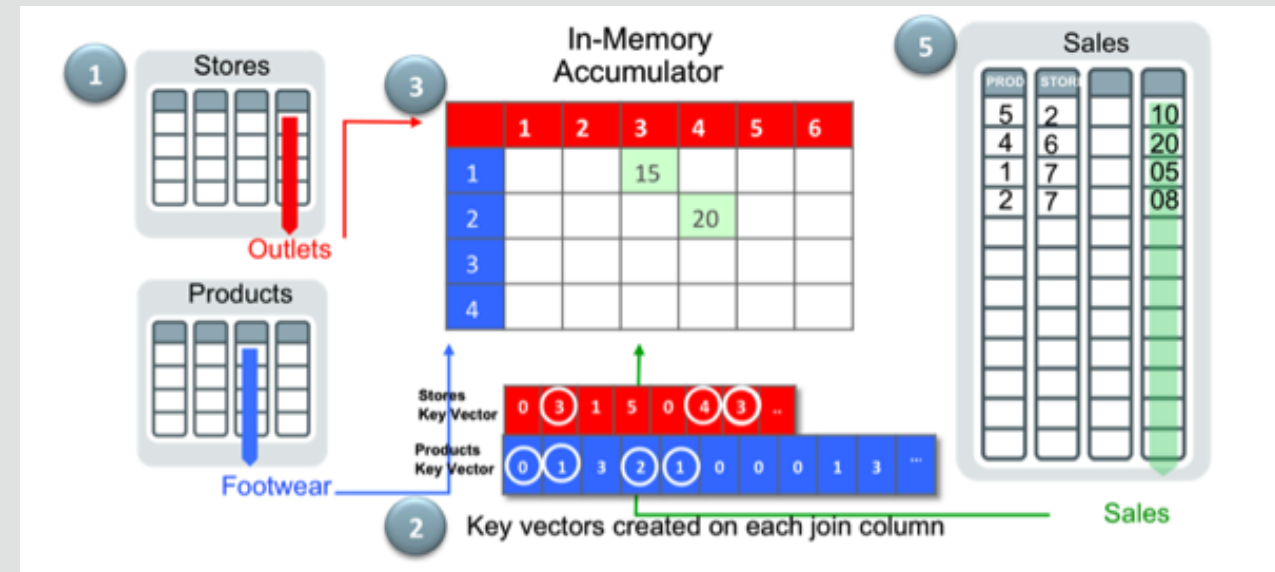
# Faster Analytics | In-Memory Aggregation

## Aggregation Push-Down

- Improve Single Table Aggregation
- Push aggregation operators down into the scan operators
- Reduce number of rows flowing back up to SQL layer
- New aggregation algorithms leveraging In-Memory data formats and SIMD
- **2-10X improvements**

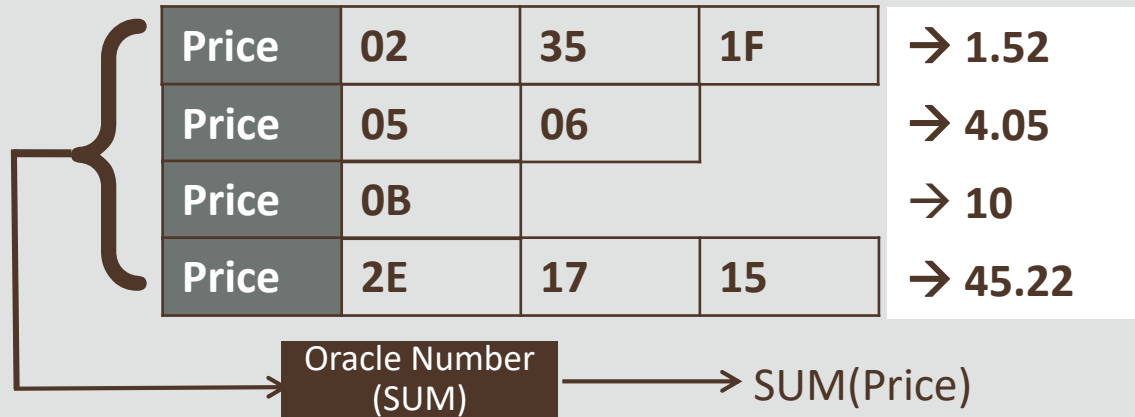
## Vector Transformation

- Improve Aggregation over Joins
- Query transformation replaces aggregation over hash-joins with new push-down operators.

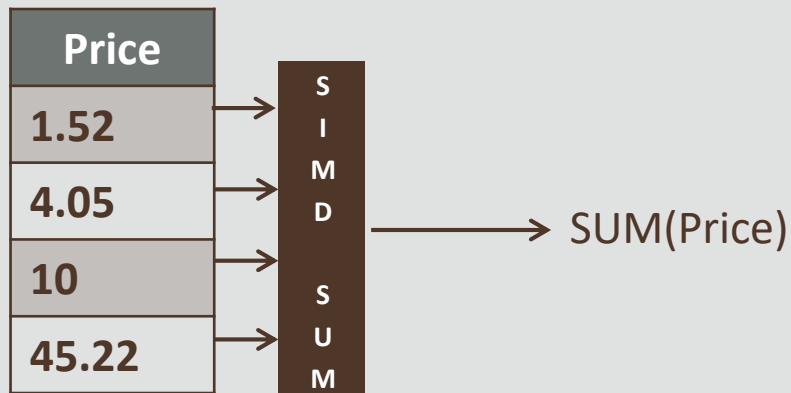


# Faster Analytics | In-Memory Numbers

**SLOW** Row-by-Row Oracle Number Processing



**FAST** SIMD Vector Processing of In-Memory Numbers



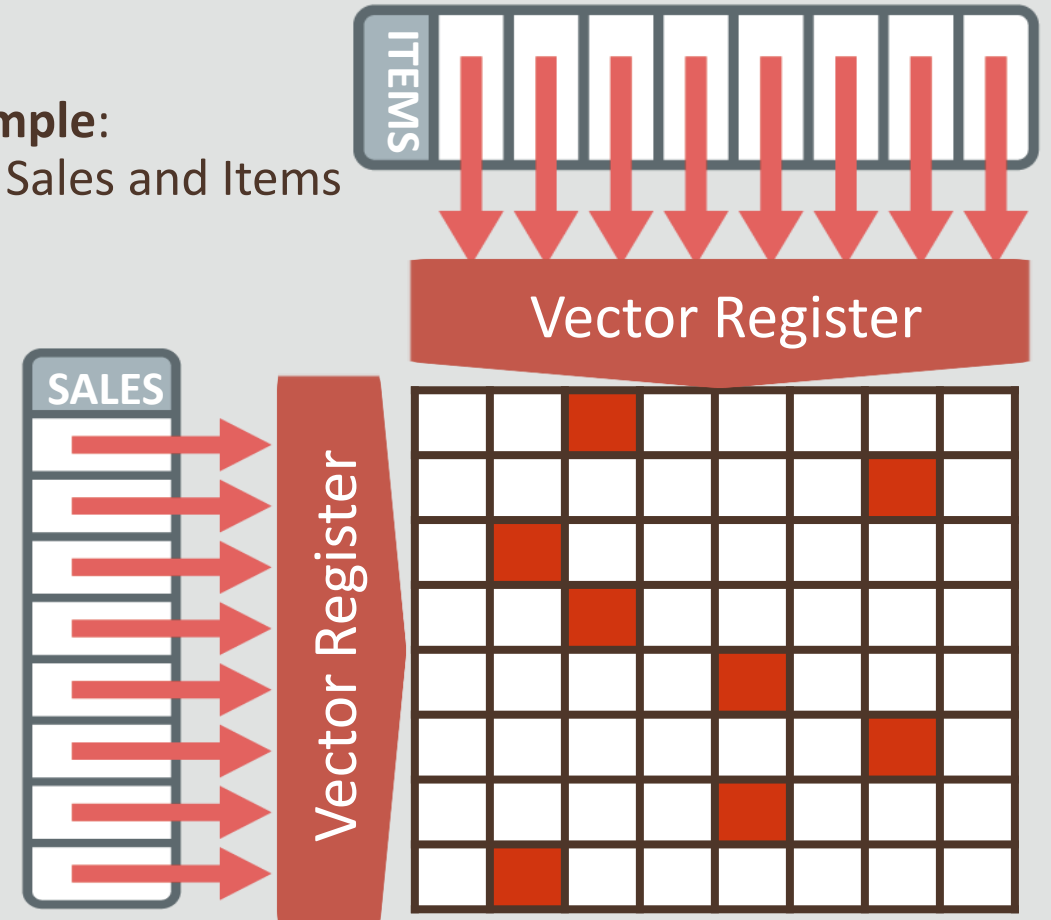
- In-Memory optimized format for NUMBER columns
  - Instead of software-implemented, variable-width ORACLE NUMBERS
  - Enabled using new parameter **inmemory\_optimized\_arithmetic**
- SIMD Vector Processing on optimized inmemory number format
- Aggregation and Arithmetic operators can improve **up to 20X**

# Preview | In-Memory Vector Joins

NEW IN  
20<sup>C</sup>

- New *Deep Vectorization* framework allows SIMD vectorization for a wide range of query operators
- *In-Memory Vector Joins* uses this framework to accelerate **Complex Joins**
  - Match multiple rows between SALES and ITEMS tables in a single SIMD Vector Instruction
  - **5-10x faster** in-memory join processing

Example:  
Join Sales and Items

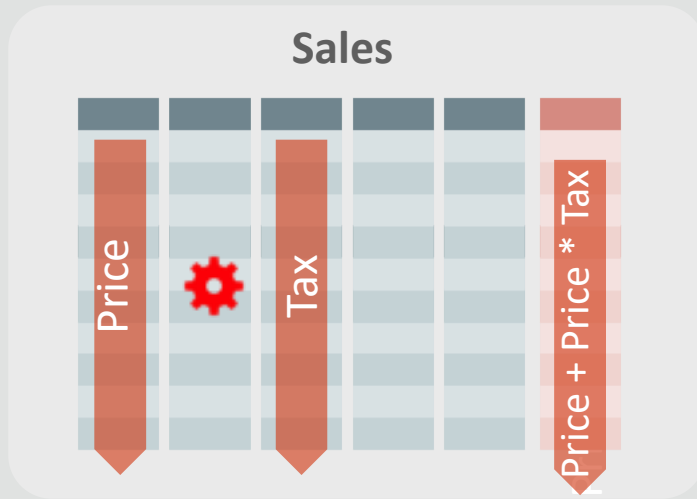




# Faster Analytics | In-Memory Expressions

Example: Compute total sales price

**Net = Price + Price \* Tax**

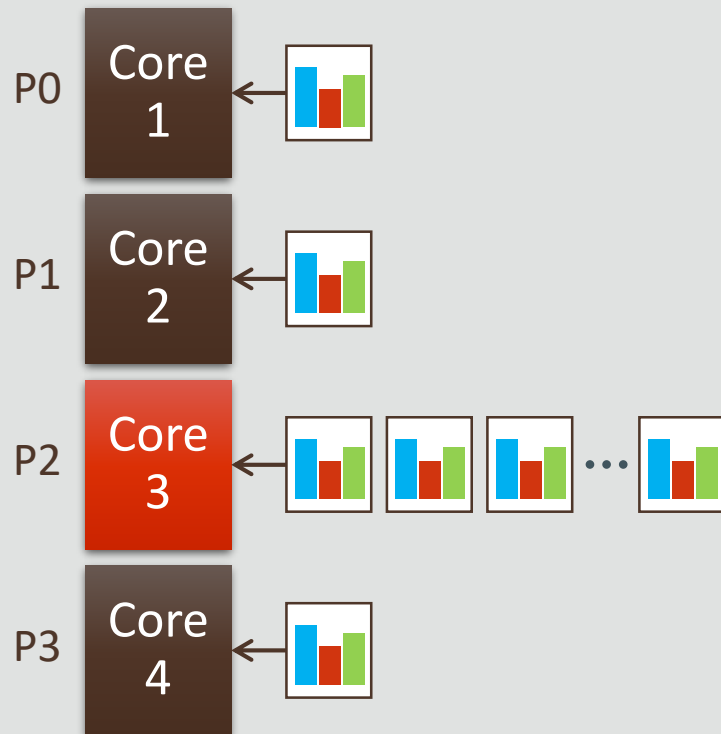


```
CREATE TABLE SALES (  
  PRICE NUMBER, TAX NUMBER, ...,  
  NET AS (PRICE + PRICE * TAX)  
)  
INMEMORY;
```

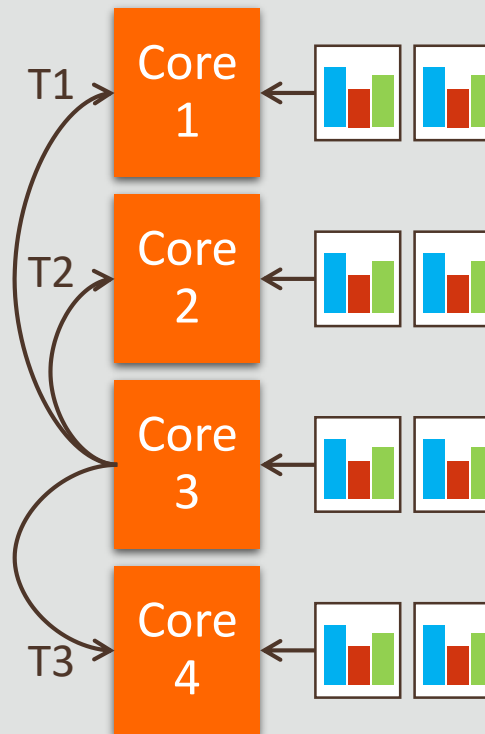
- Hot expressions can be stored as additional columns in memory
- All In-Memory optimizations apply to expression columns (e.g. Vector processing, storage indexes)
- Two modes:
  - **Manual:** Declare virtual columns for desired inmemory expressions
  - **Auto:** Auto detect frequent expressions
- **3-5x** faster complex queries

# Faster Analytics | In-Memory Dynamic Scans

## Parallel SQL

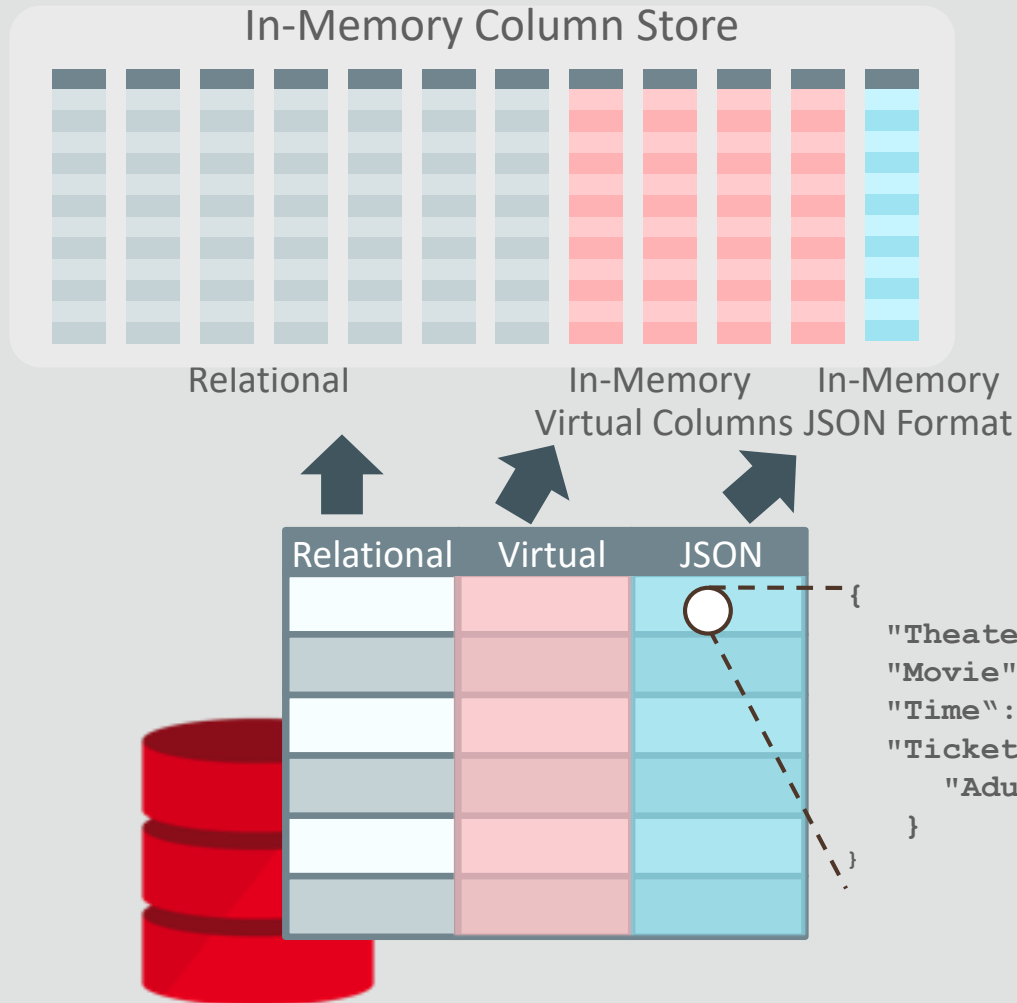


## Parallel SQL + IMDS



- Parallelize operations pushed down to SCAN layer using light-weight threads
- Supplements *static* PQ plans with faster response times for shorter queries.
  - Achieve PQ execution times for single-threaded queries
- Elastic DOP Rebalancing using Resource Manager
- **Up to 2X gains seen**

# Faster Converged Analytics | In-Memory JSON



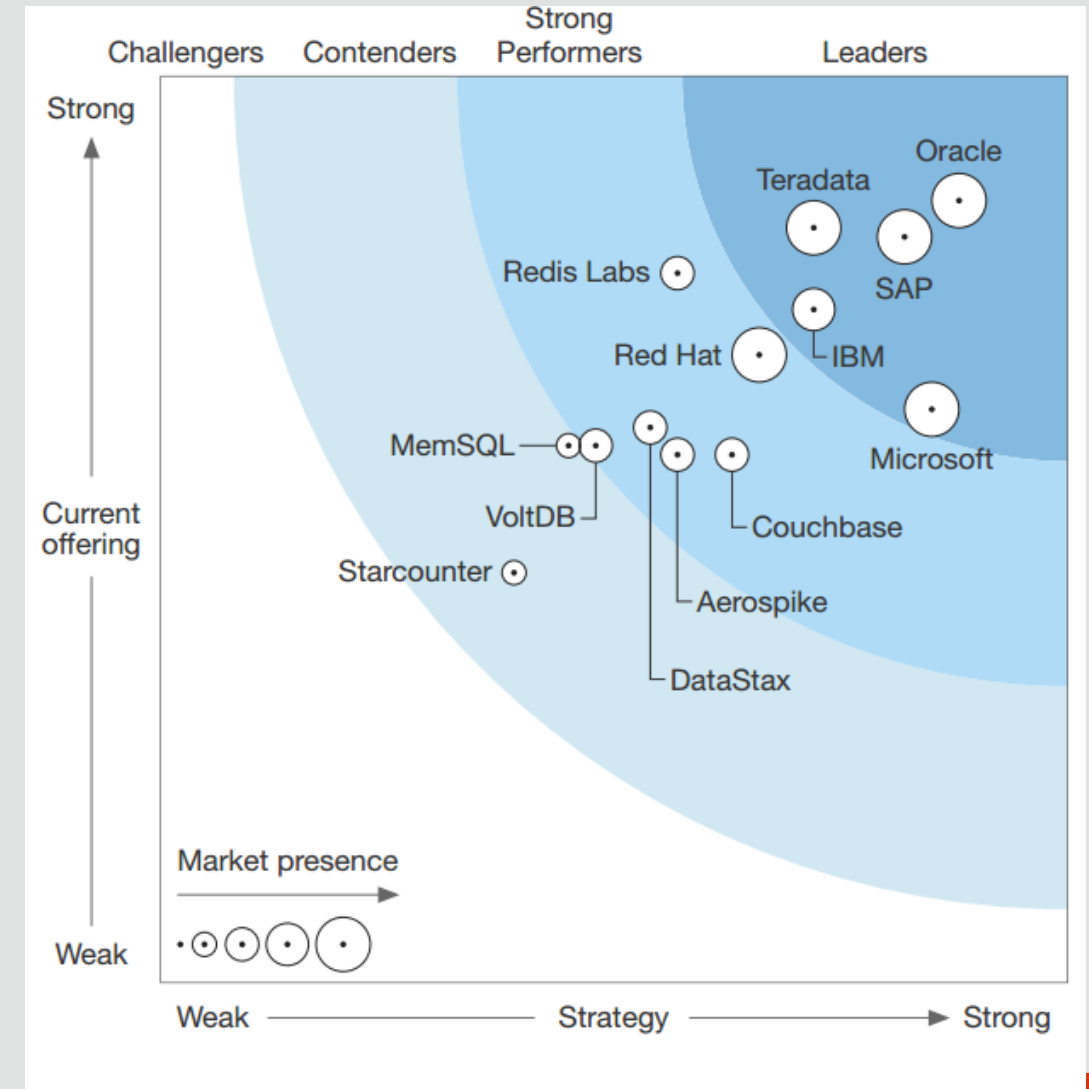
- Full JSON documents populated using an optimized binary format
- Additional expressions can be created on JSON columns (e.g. JSON\_VALUE) & stored in column store
- Queries on JSON content or expressions automatically directed to In-Memory format
  - E.g. Find movies where movie.name contains "Rogue"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

# The Forrester Wave™: In-Memory Databases, Q1 2017

**Oracle In-Memory Databases  
Scored Highest by Forrester  
on both Current Offering  
and Strategy**

<http://www.oracle.com/us/corporate/analystreports/forrester-imdb-wave-2017-3616348.pdf>

The Forrester Wave™ is copyrighted by Forrester Research, Inc. Forrester and Forrester Wave™ are trademarks of Forrester Research, Inc. The Forrester Wave™ is a graphical representation of Forrester's call on a market and is plotted using a detailed spreadsheet with exposed scores, weightings, and comments. Forrester does not endorse any vendor, product, or service depicted in the Forrester Wave. Information is based on best available resources. Opinions reflect judgment at the time and are subject to change.





ORACLE