



ORACLE

ORACLE

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

CMU, 2019

Shasank Chavan

Vice President, In-Memory Database Technologies

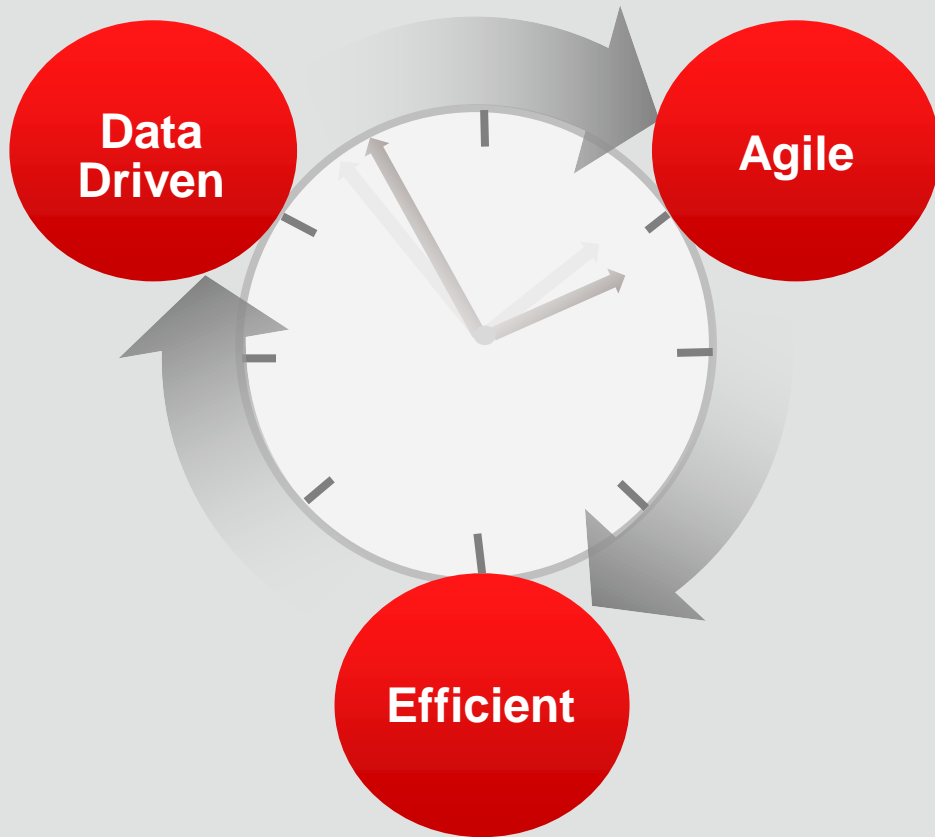


Safe Harbor

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.

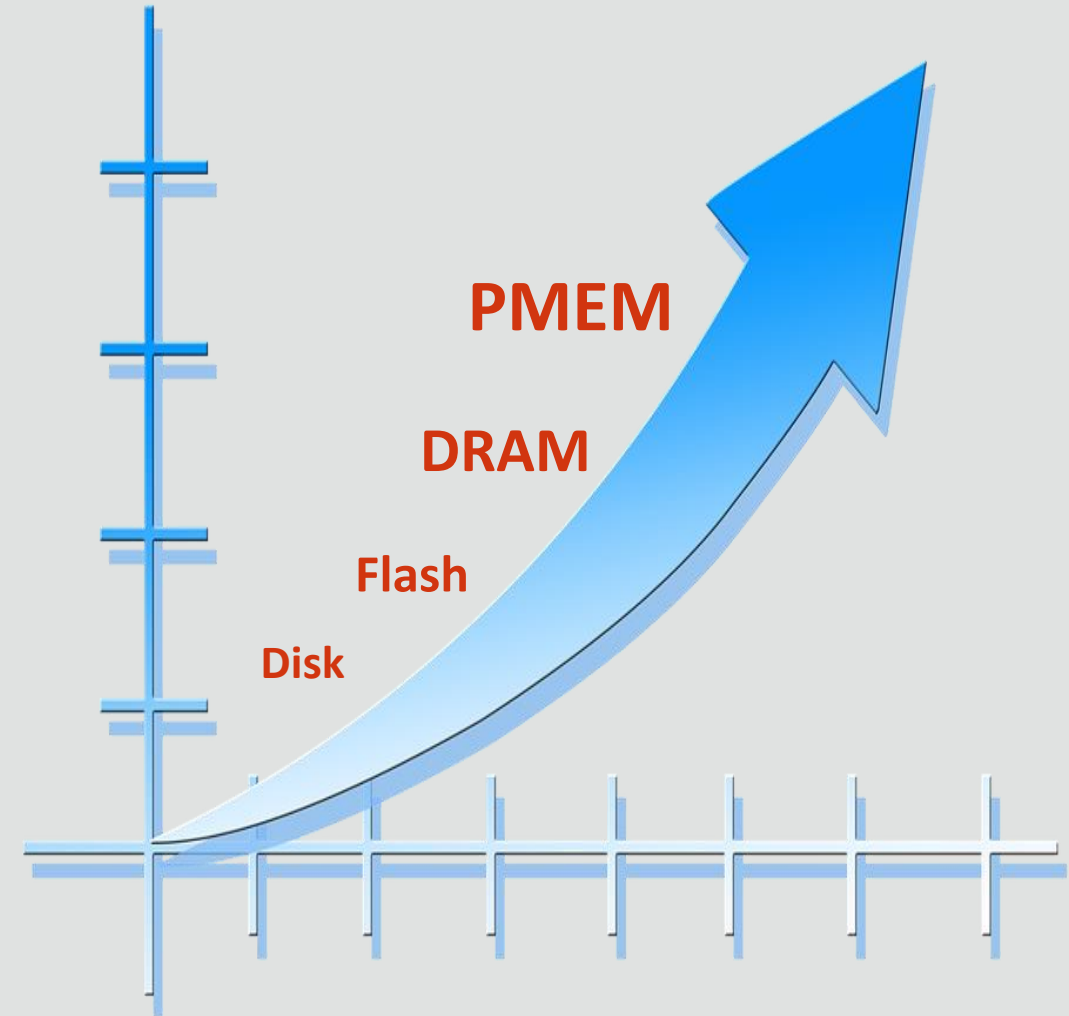
In-Memory Now | Real-Time Enterprises



- **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)



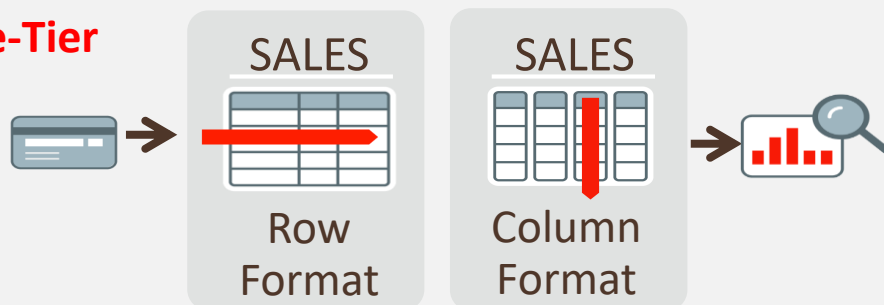
In-Memory Now | Technology Across All Tiers

Application-Tier



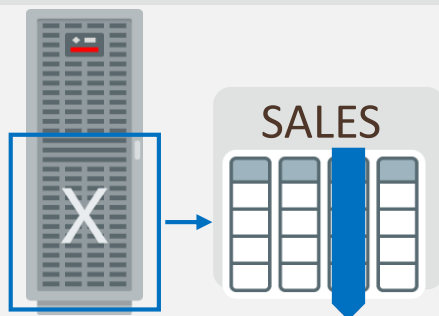
- TimesTen-In-Memory Database
 - Latency Critical OLTP Applications
 - **Microsecond** response time
 - Standalone or Cache for Oracle Database

Database-Tier

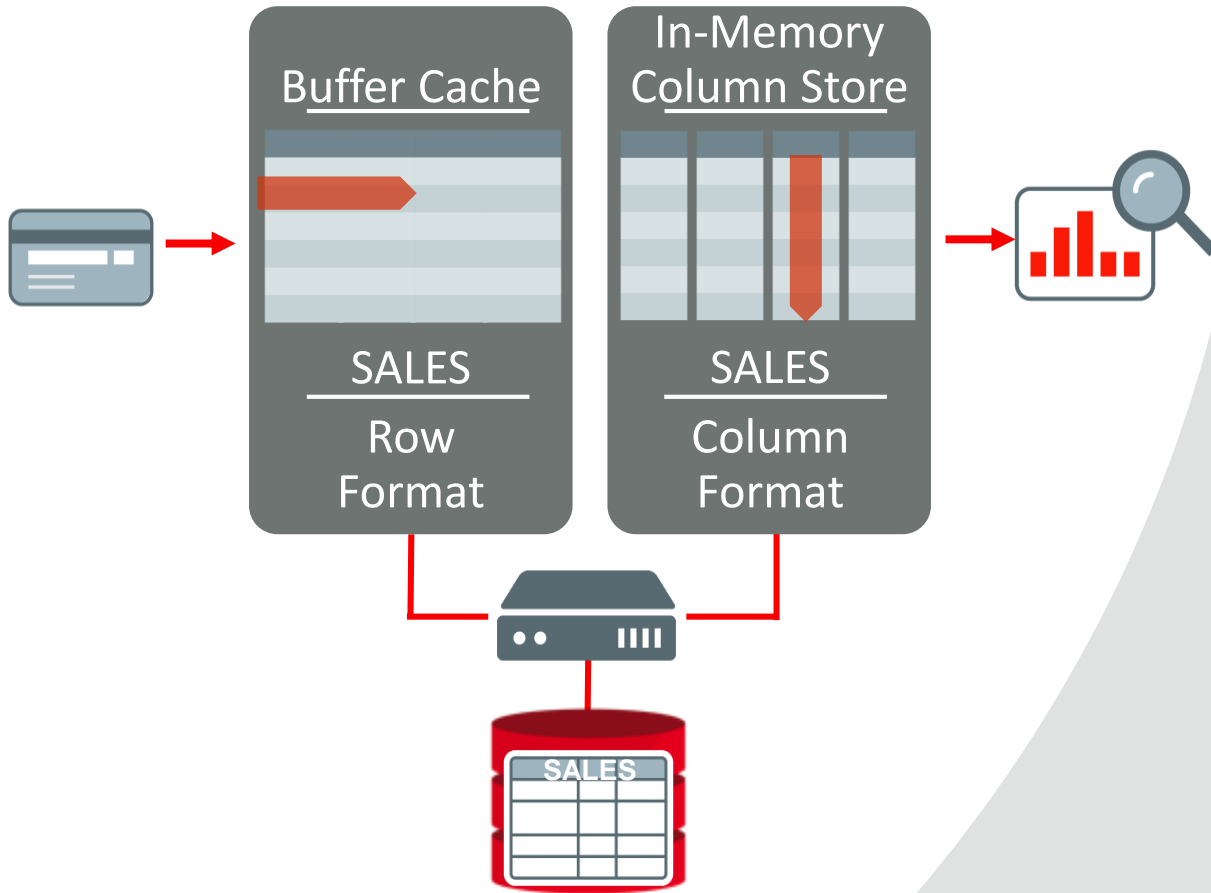


- Database In-Memory
 - Dual Format In-Memory Database
 - **Billions of Rows/sec** analytic data processing
 - **2-3x** Faster Mixed Workloads

Storage-Tier



- In-Memory on Exadata Storage
 - In-memory format on Exadata Flash Cache
 - **5-10x** faster smart scan in storage
 - **15x** increase in total columnar capacity



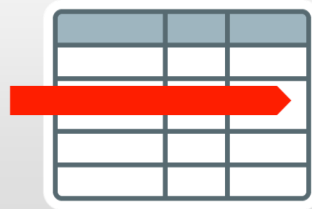
Oracle Database In-Memory

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**

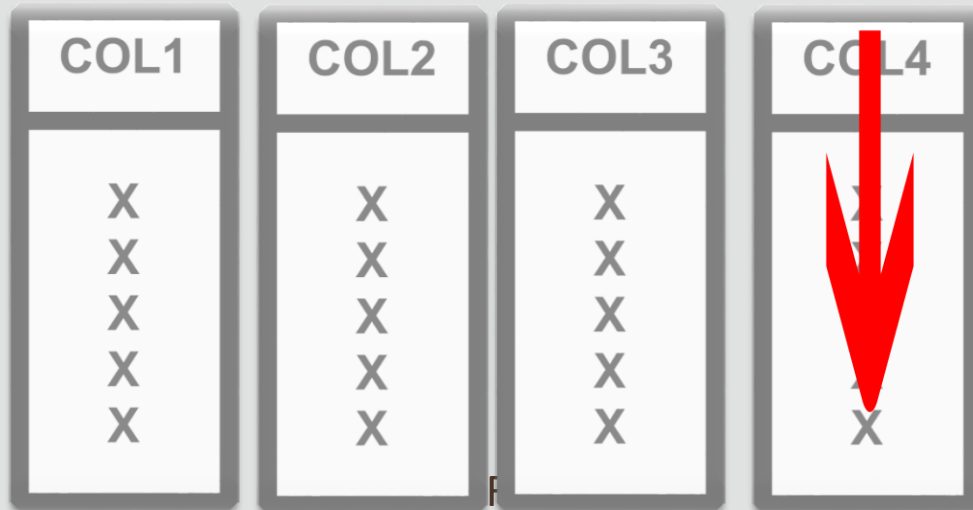
Column



▪ **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

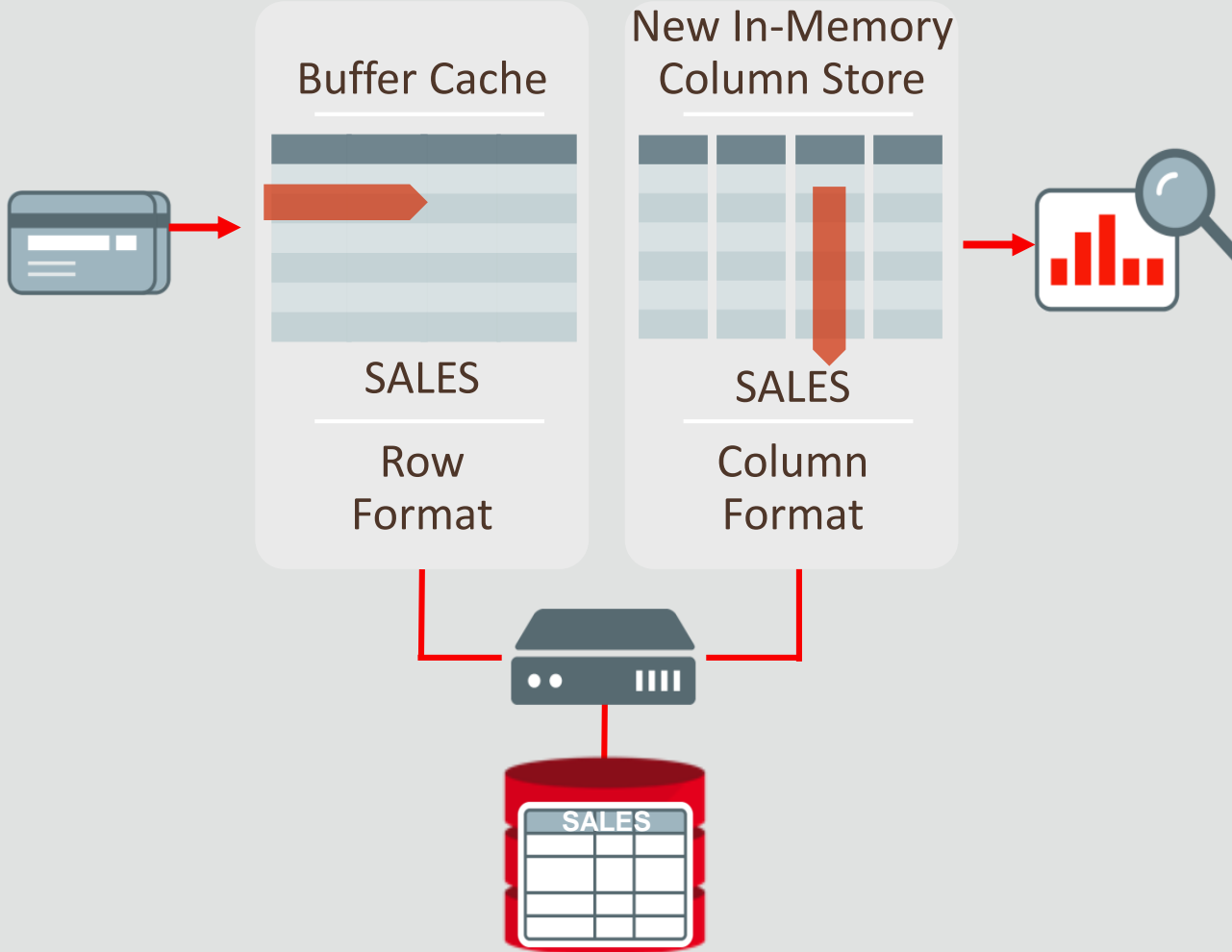
Column



- **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

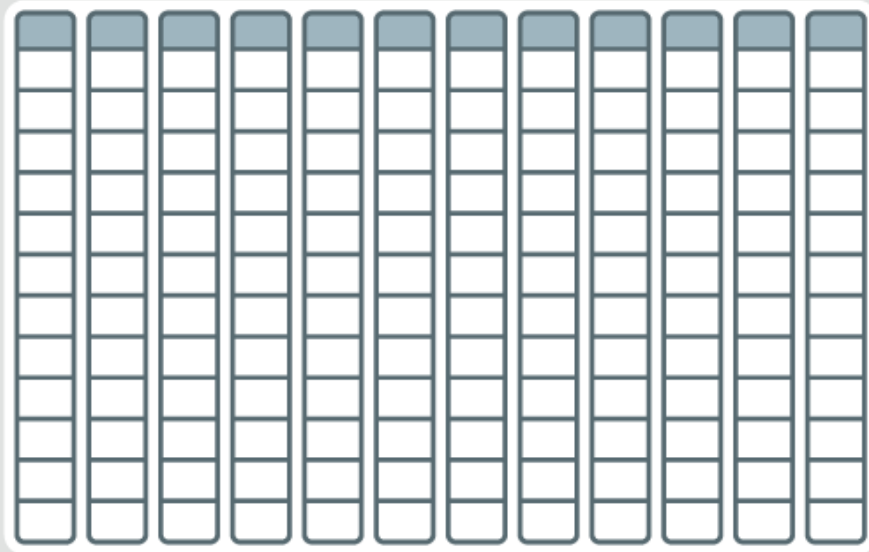
Database In-Memory | Architecture



- **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

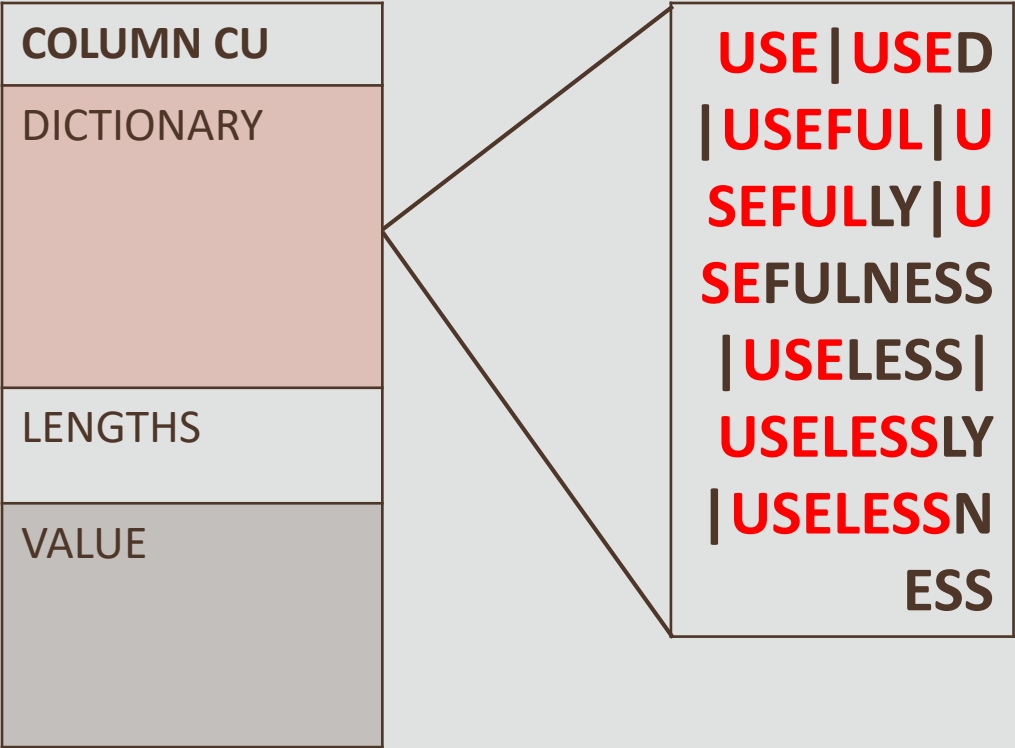
0: 00101101
1: 01

010

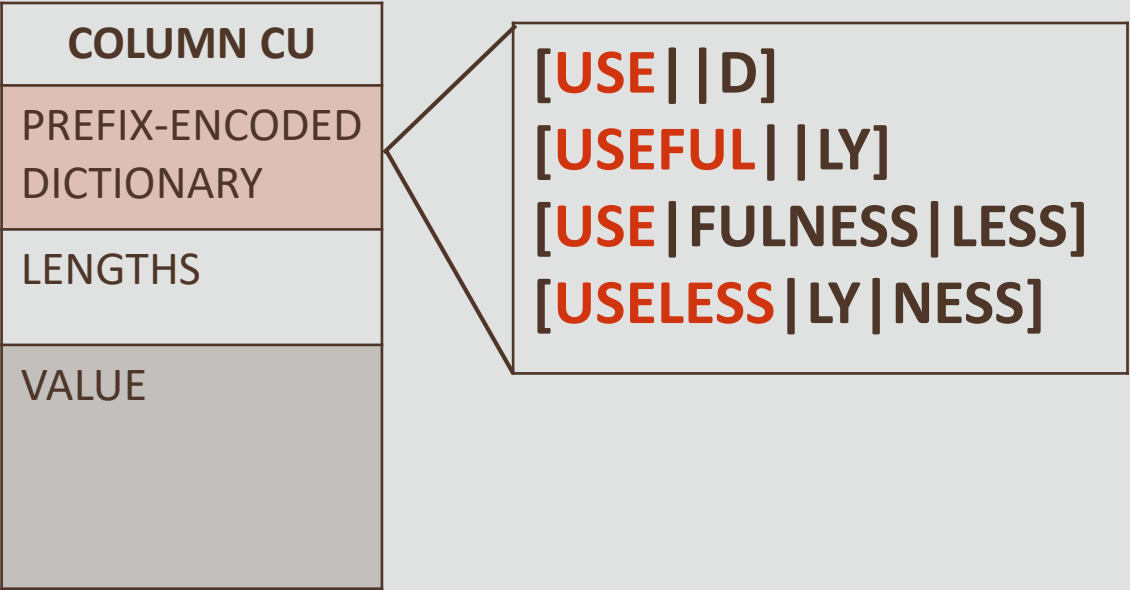


In-Memory Columnar Format | Compression

No Compression

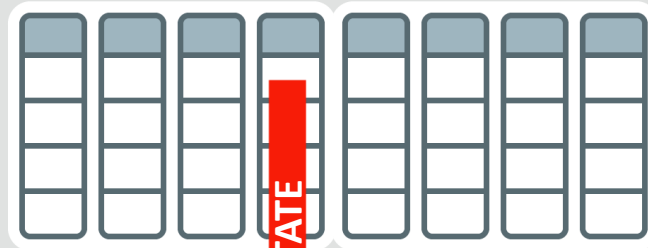


Prefix Compression



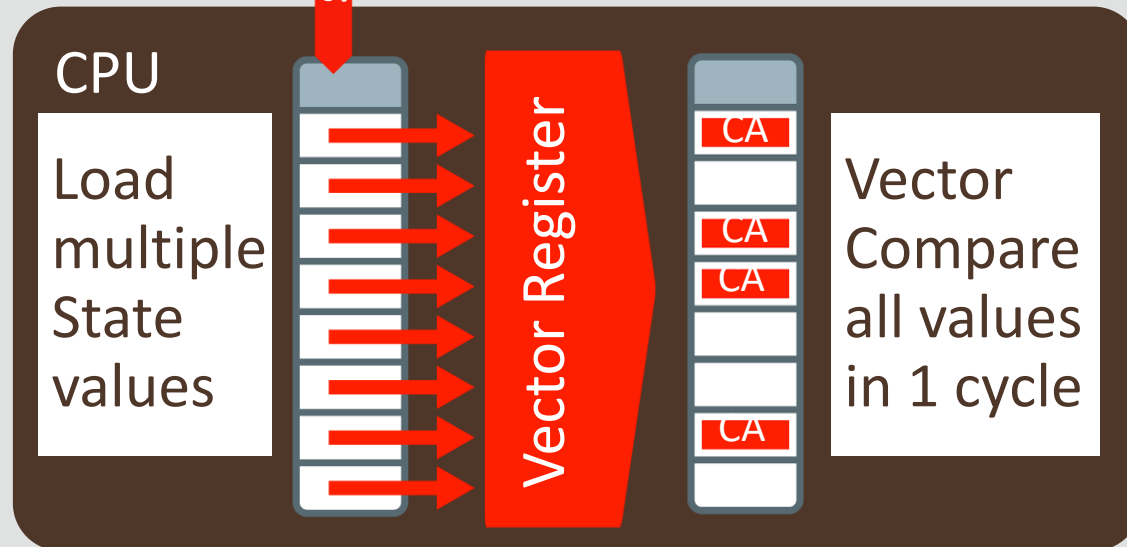
In-Memory Enables **SIMD** Vector Processing

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

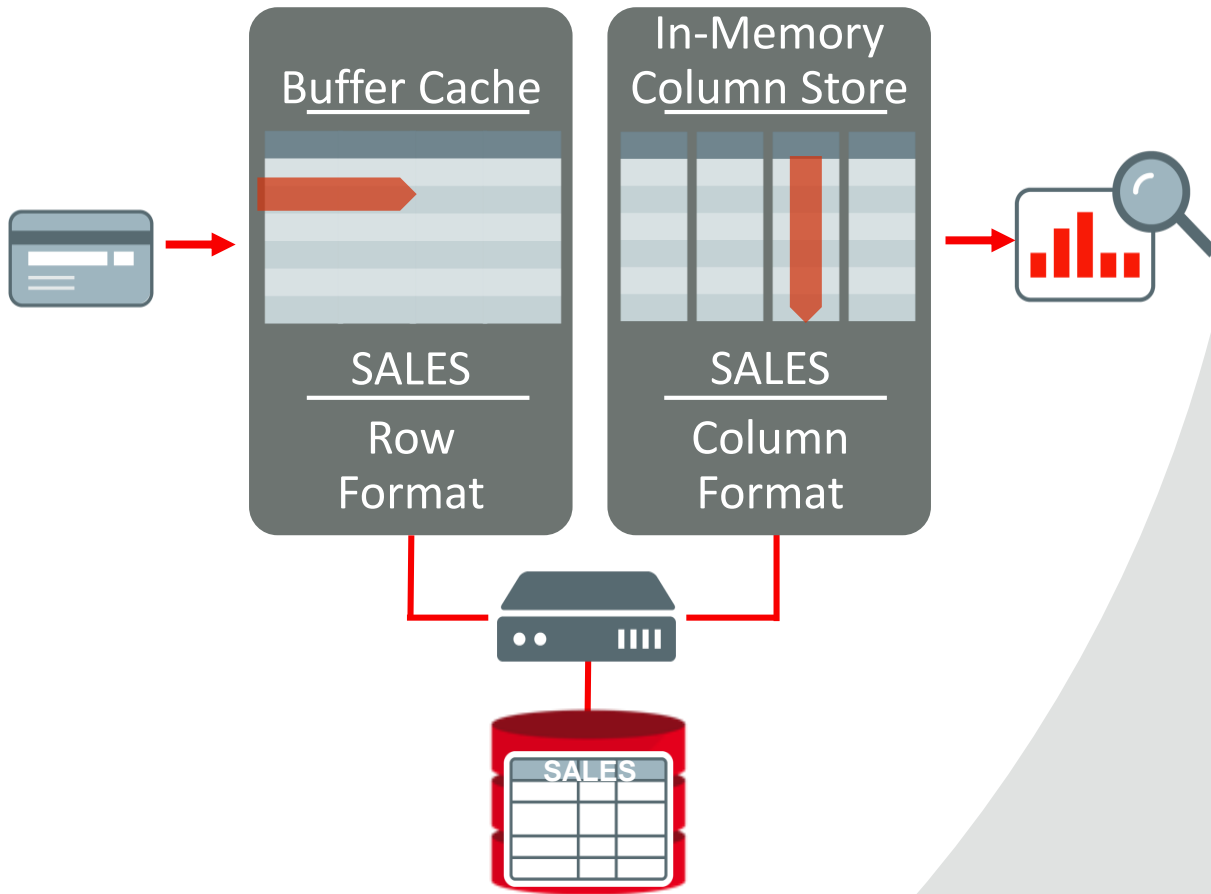


- Run reports with aggregations and joins **10x faster**



Top-5

Oracle Database In-Memory Innovations

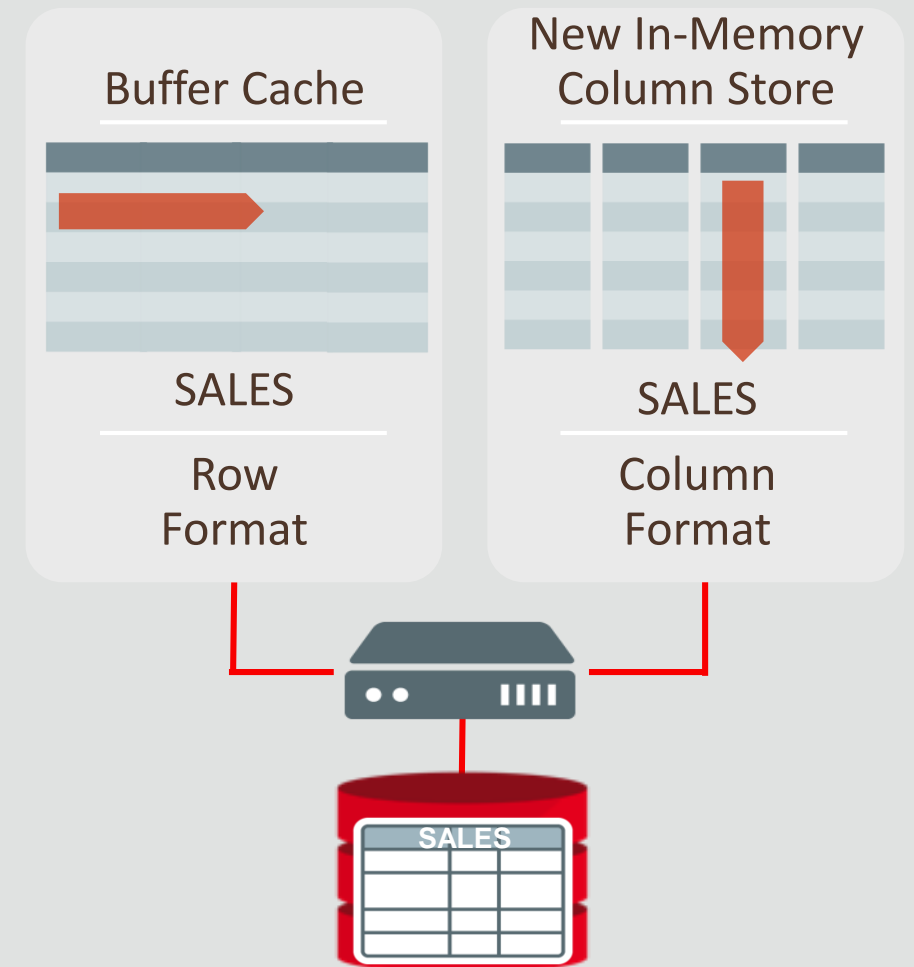


#1 Dual-Format Architecture

*Fast Mixed Workloads, Faster
Analytics*

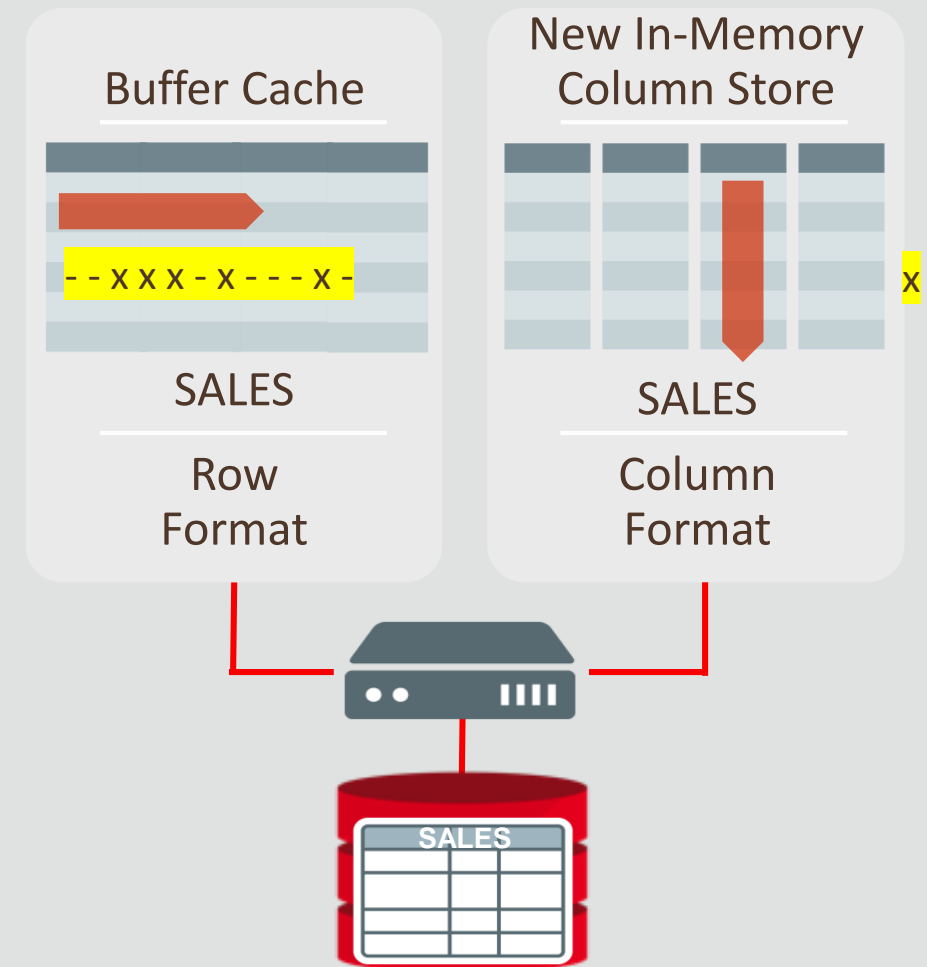
In-Memory: Dual-Format Architecture

- 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)



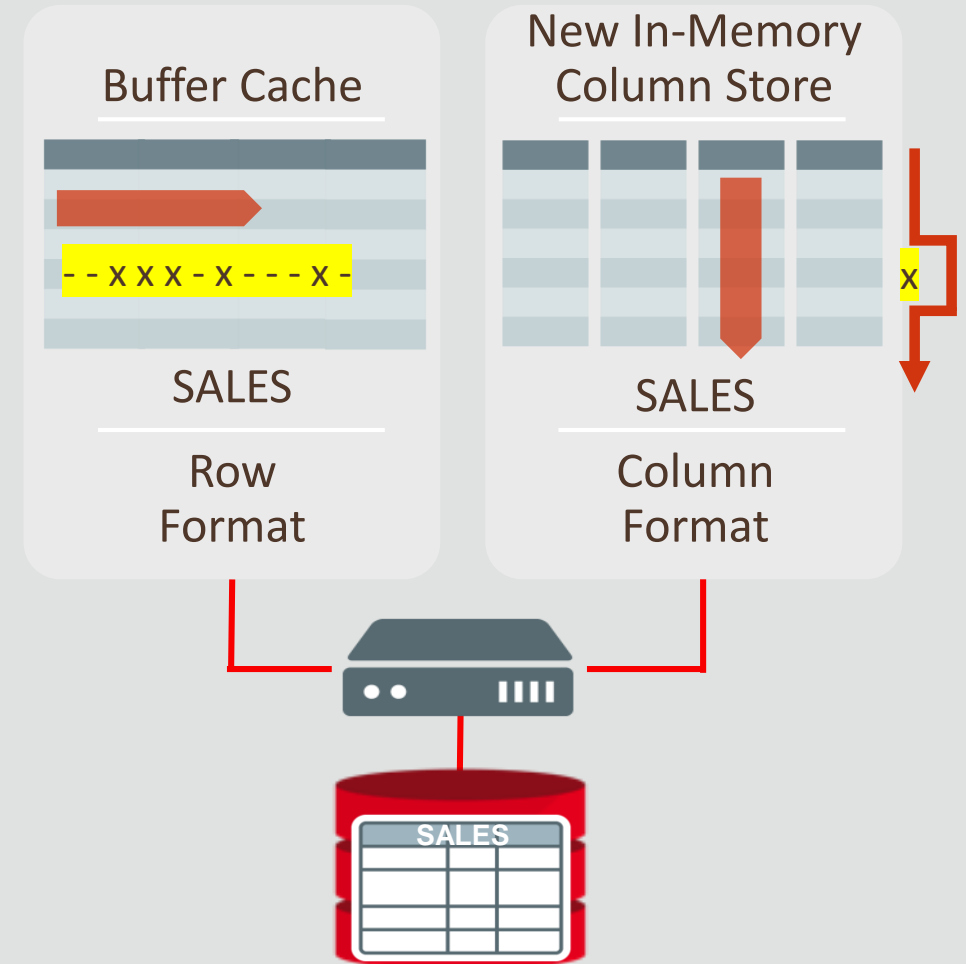
In-Memory: Dual-Format Architecture

- 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



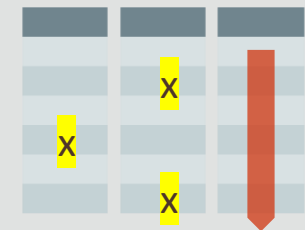
Old IMCU stays online until New IMCU is built. Then A *switcher* happens once New IMCU is ready.

Incremental Repopulation



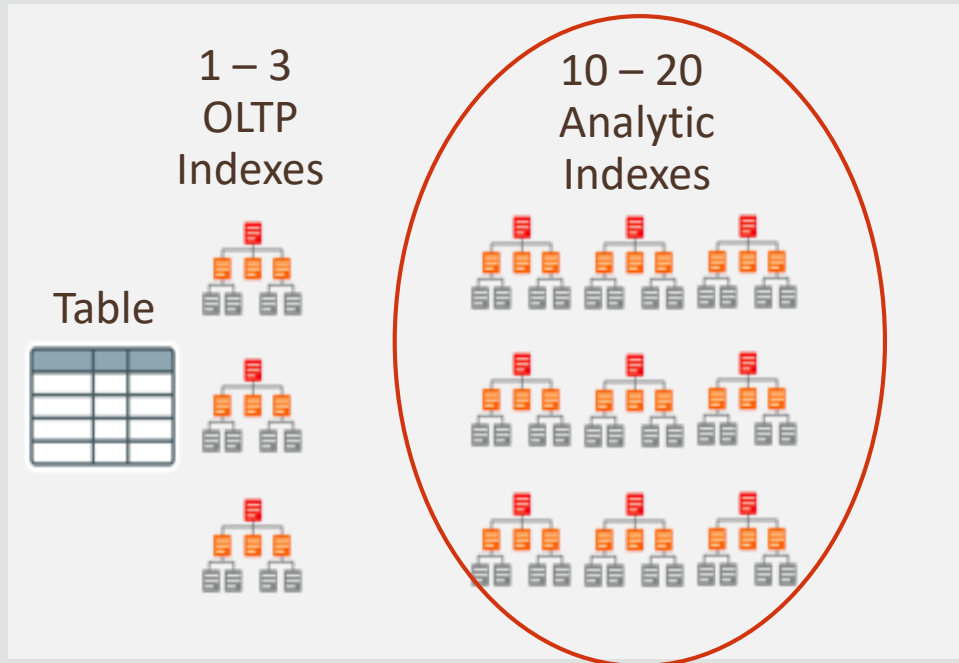
Build new columns in new IMCU using meta-data present in the old IMCU, allowing quick formatting

Column-Level Invalidations



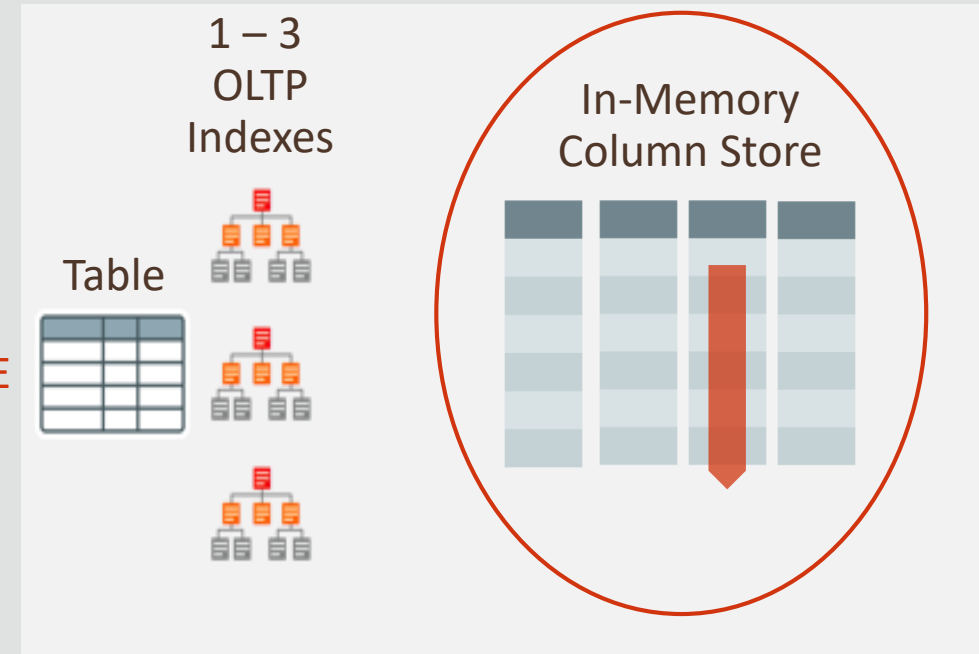
Column-Level Invalidations tracked to still enable IM scans

Accelerates Mixed Workloads (Hybrid OLTP)

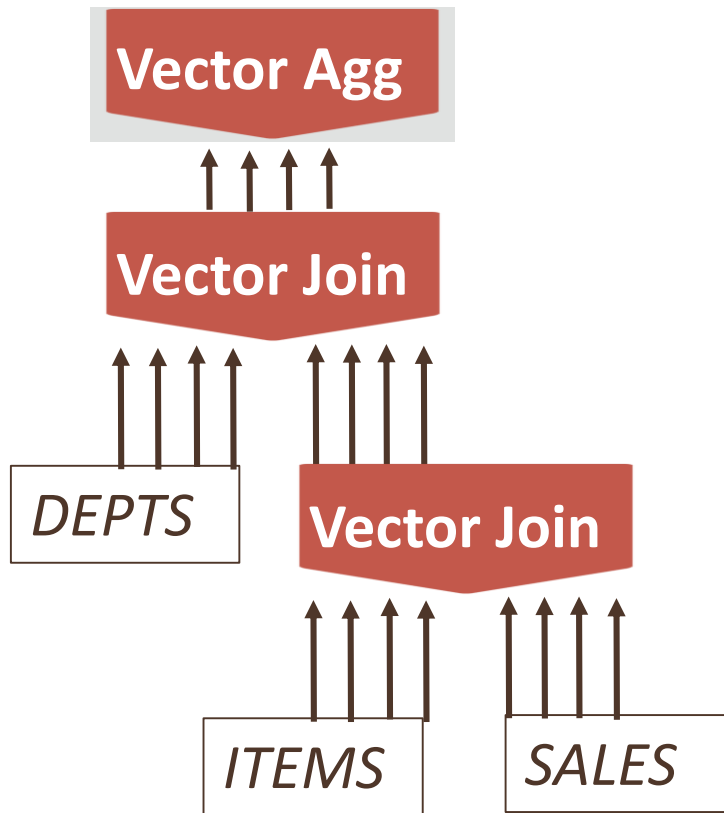


- Inserting one row into a table requires updating 10-20 analytic indexes: **Slow!**
- Fast analytics only on indexed columns
- Analytic indexes **increase** database size

REPLACE



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



#2

Vectorized Analytics

SIMD Vector Processing at Billions of Rows per Second

Faster Scans | SIMD Vector Processing

- Parallelize predicate evaluation – load, eval, store/consume result
- Select count(*) from T where a > 10 and b < 20
 - [Load] A

	96	64	32	0
51	95182	1	69	

Faster Scans | SIMD Vector Processing

- Parallelize predicate evaluation – load, eval, store/consume result
- Select count(*) from T where a > 10 and b < 20
 - [Load] A
 - [Load] Temp = 10

96	64	32	0
51	95182	1	69
10	10	10	10

Faster Scans | SIMD Vector Processing

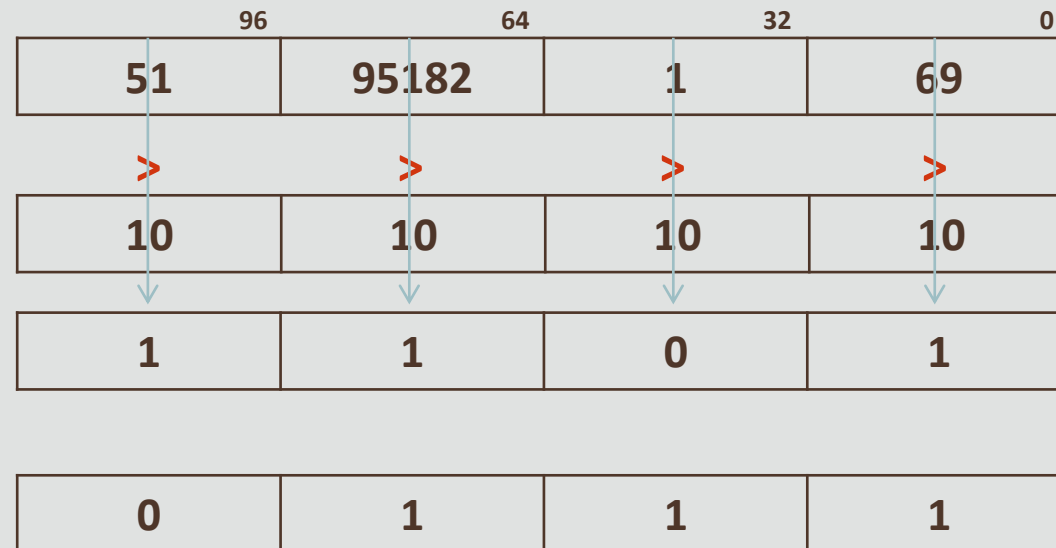
- Parallelize predicate evaluation – load, eval, store/consume result
- Select count(*) from T where $a > 10$ and $b < 20$
 - [Load] A
 - [Load] Temp = 10
 - [Compare] $A > \text{Temp}$

96	64	32	0
51	95182	1	69
>	>	>	>
10	10	10	10
1	1	0	1

Faster Scans | SIMD Vector Processing

- Parallelize predicate evaluation – load, eval, store/consume result
- Select count(*) from T where a > 10 and b < 20

- [Load] A
- [Load] Temp = 10
- [Compare] A > Temp



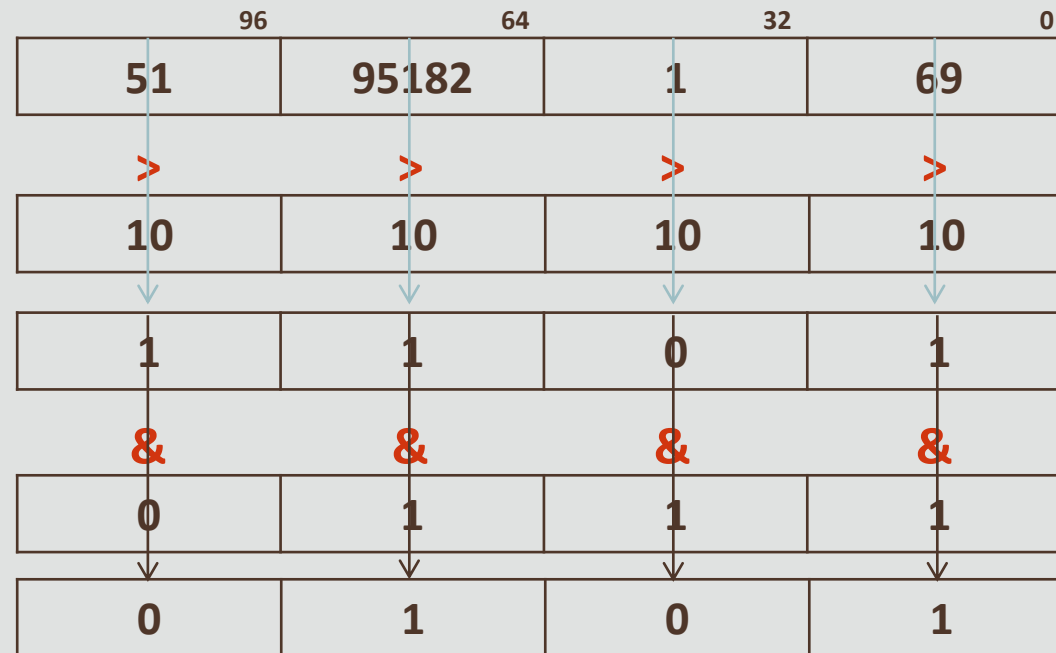
- Load B, Compare 20

Faster Scans | SIMD Vector Processing

- Parallelize predicate evaluation – load, eval, store/consume result
- Select count(*) from T where a > 10 and b < 20

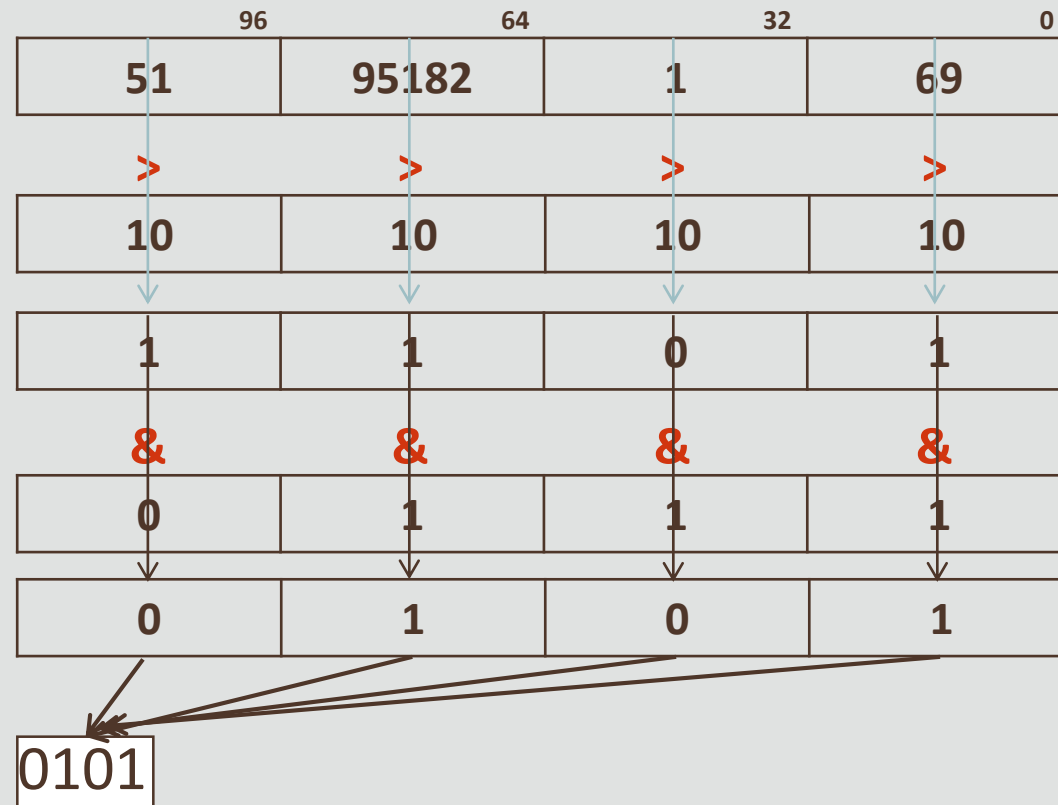
- [Load] A
- [Load] Temp = 10
- [Compare] A > Temp

- Load B, Compare 20
- And



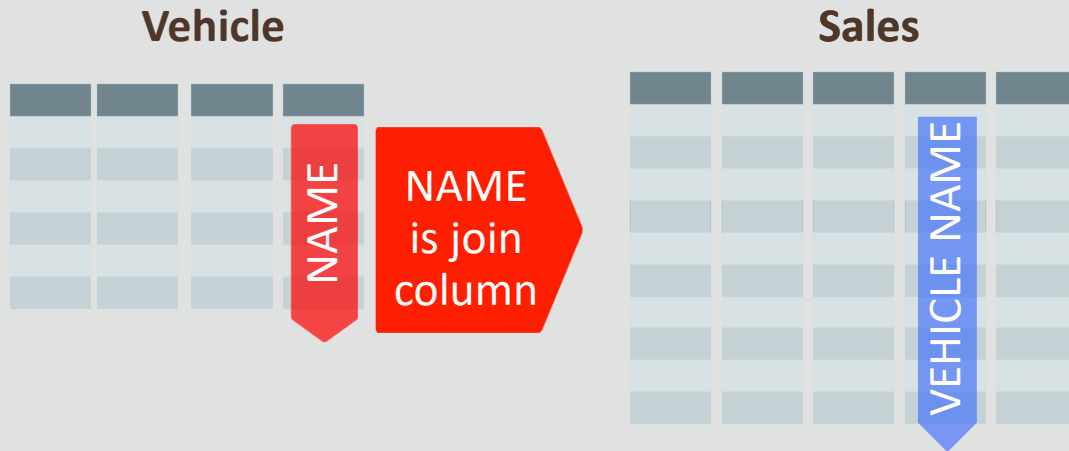
Faster Scans | SIMD Vector Processing

- Parallelize predicate evaluation – load, eval, store/consume result
- Select count(*) from T where a > 10 and b < 20
 - [Load] A
 - [Load] Temp = 10
 - [Compare] A > Temp
 - Load B, Compare 20
 - And
 - Mask, Store Bit-Map



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

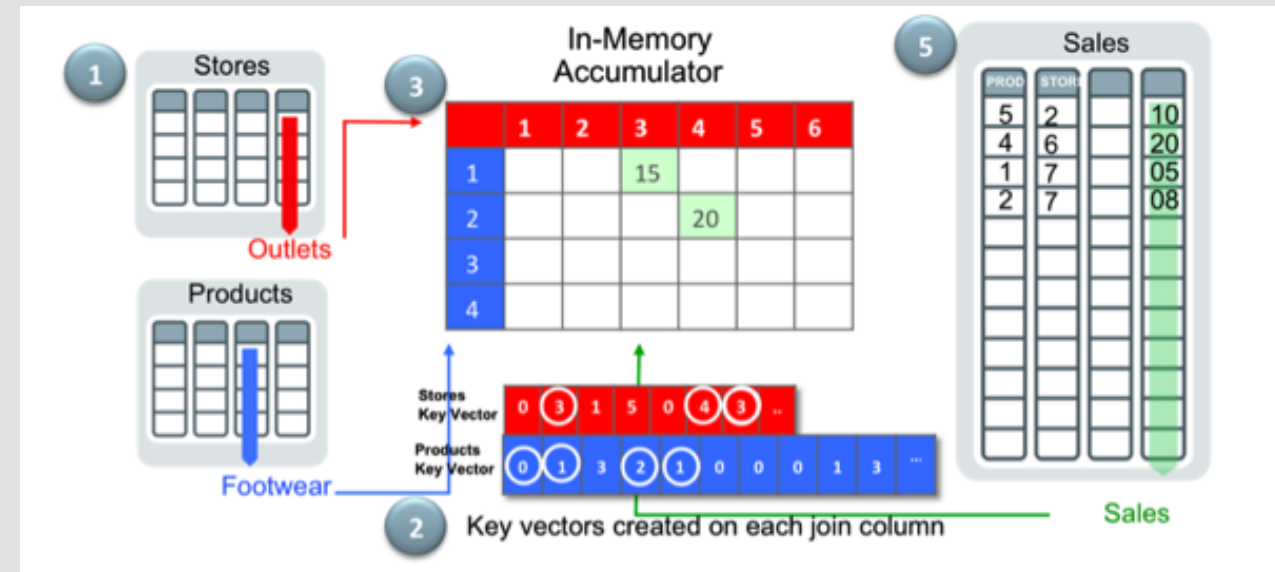
Faster Analytics | In-Memory Aggregation

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.



GBY Sum for VERY LARGE Numbers

- “Select sum(A) from T where ... group by J, K”

A	J	K
51819	0	1
23591	0	1
51819	1	0
39510	0	0
10001	1	1
9599	1	0
10001	1	1

GBY Sum for VERY LARGE Numbers

- “Select sum(A) from T where ... group by J, K”

A	J	K
51819	0	1
23591	0	1
51819	1	0
39510	0	0
10001	1	1
9599	1	0
10001	1	1

← Groups →

←----- Dictionary Codes -----→								
		10001					51819	
{0,0}	0	0	0	1	0	0	0	0
{0,1}	0	0	1	0	0	0	1	0
{1,0}	1	0	0	0	0	0	1	0
{1,1}	0	2	0	0	0	0	0	0

Frequency Table

GBY Sum for VERY LARGE Numbers

- “Select sum(A) from T where ... group by J, K”

←----- Dictionary Codes ----->

10001

51819

A	J	K
51819	0	1
23591	0	1
51819	1	0
39510	0	0
10001	1	1
9599	1	0
10001	1	1

Groups
↑
↓

		10001						51819	
{0,0}	0	0	0	1	0	0	0	0	
{0,1}	0	0	1	0	0	0	1	0	
{1,0}	1	0	0	0	0	0	1	0	
{1,1}	0	2	0	0	0	0	0	0	

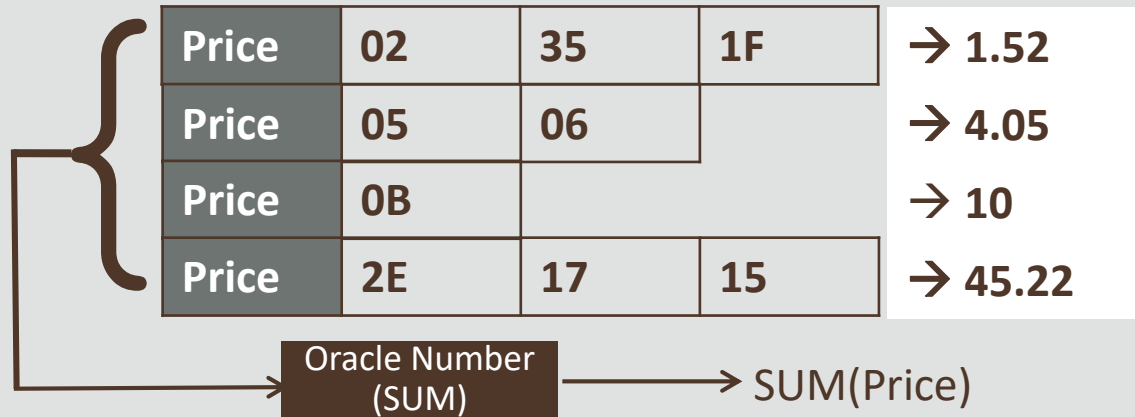
Frequency Table

AGG[GROUP_ID] = FREQUENCY * DICT_SYMBOL

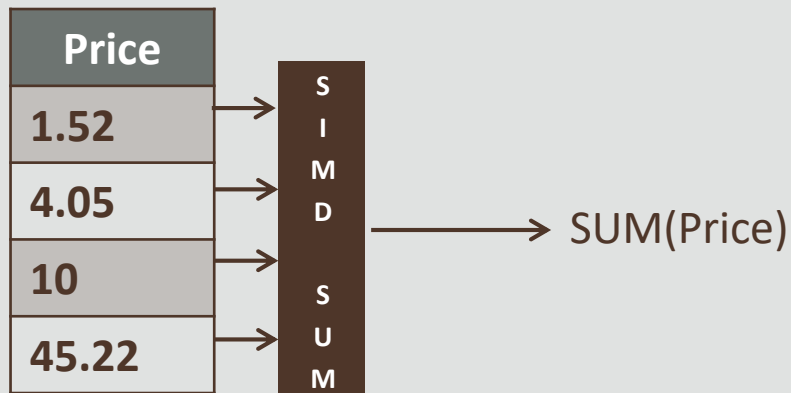
AGG{0,0} = (0 * DICT[0]) + (0 * DICT[1]) + + (1 * DICT[39510]) +
AGG{0,1} = (0 * DICT[0]) + ... + (1 * DICT[23591]) + ... + (1 * DICT[51819]) +
AGG{1,0} = (0 * DICT[0]) + ... + (1 * DICT[9599]) + ... + (1 * DICT[51819]) + ...
AGG{1,1} = (0 * DICT[0]) + ... + (2 * DICT[10001]) +

Faster Analytics | In-Memory Numbers

SLOW Row-by-Row Oracle Number Processing



FAST SIMD Vector Processing of In-Memory Numbers



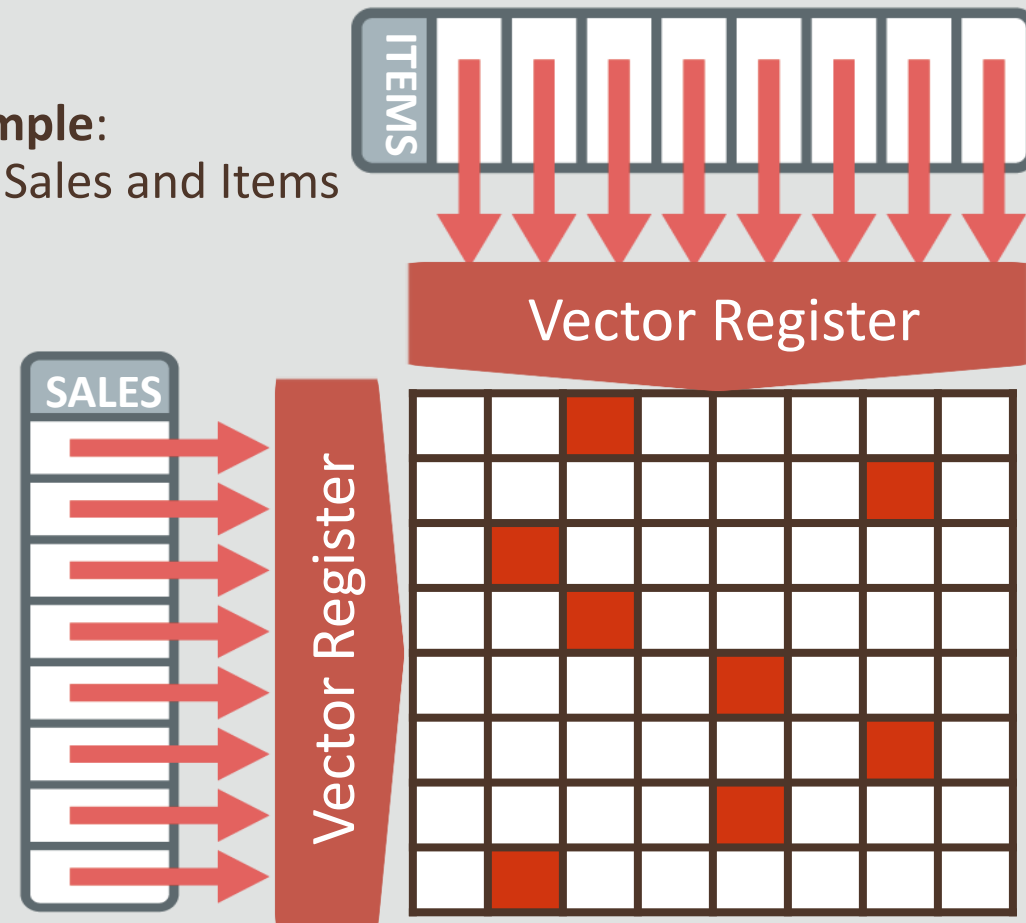
- In-Memory optimized format for NUMBER columns
 - Instead of software-implemented, variable-width ORACLE NUMBERS
 - Enabled using new parameter **inmemory_optimized_arithmetic**
- SIMD Vector Processing on optimized inmemory number format
- Aggregation and Arithmetic operators can improve **up to 20X**

Preview | In-Memory Vector Joins

NEW IN
20^C

- 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

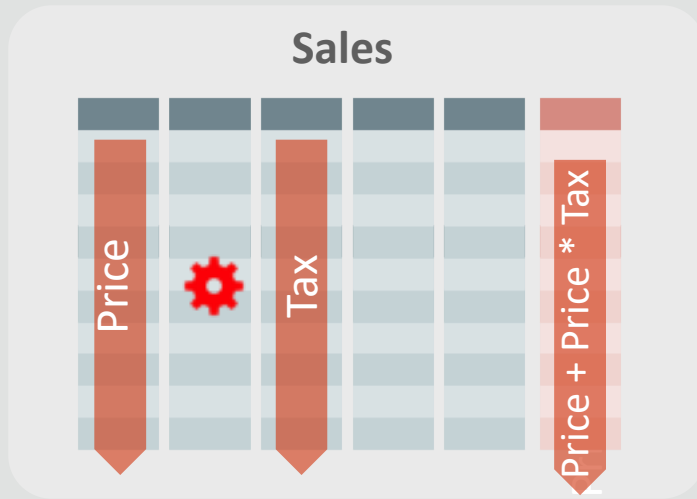
Example:
Join Sales and Items



Faster Analytics | In-Memory Expressions

Example: Compute total sales price

Net = Price + Price * Tax

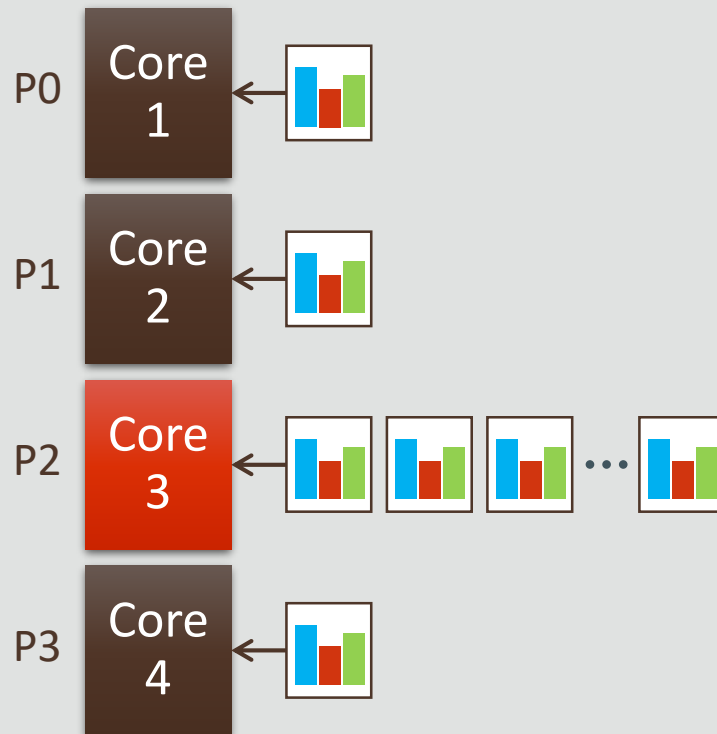


```
CREATE TABLE SALES (  
  PRICE NUMBER, TAX NUMBER, ...,  
  NET AS (PRICE + PRICE * TAX)  
)  
INMEMORY;
```

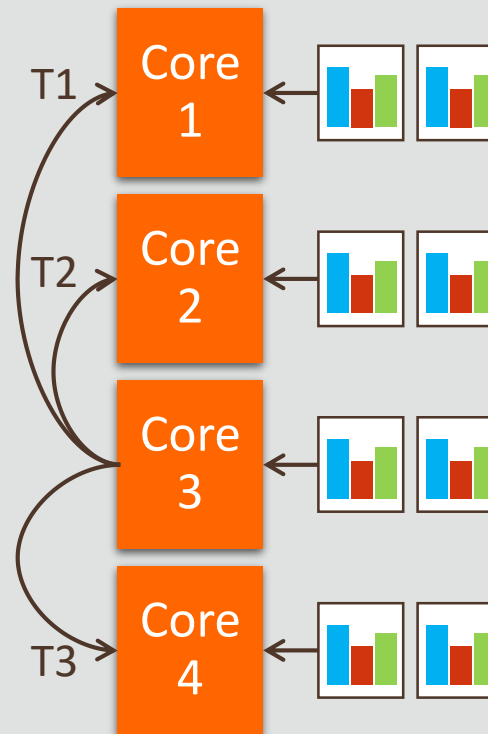
- Hot expressions can be stored as additional columns in memory
- All In-Memory optimizations apply to expression columns (e.g. Vector processing, storage indexes)
- Two modes:
 - **Manual:** Declare virtual columns for desired inmemory expressions
 - **Auto:** Auto detect frequent expressions
- **3-5x**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

Faster Analytics | In-Memory Dynamic Scans

Parallel SQL



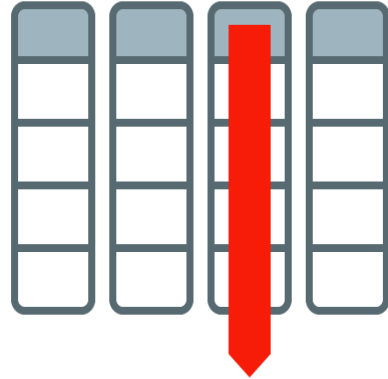
Parallel SQL + IMDS



- 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**



+

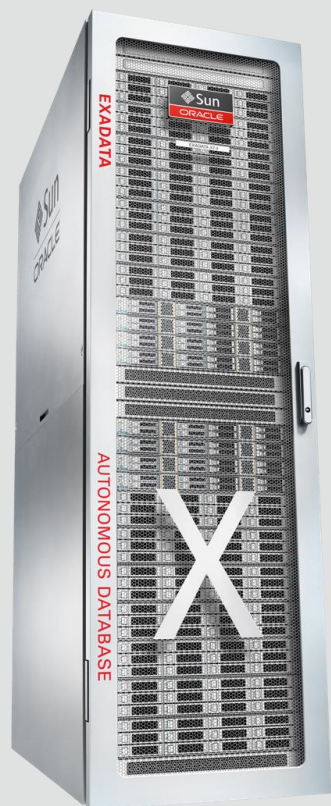


#3 In-Memory + Exadata

*In-Flash Columnar Processing at
Cloud Scale*

Background | Exadata Vision

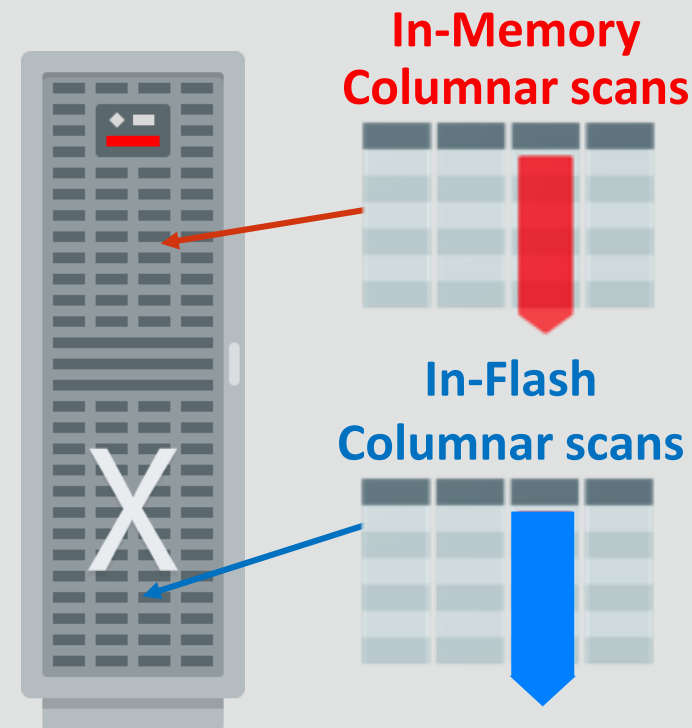
Dramatically Better Platform for All Database Workloads



- **Ideal Database Hardware** – Scale-out, database optimized compute, networking, and storage for fastest performance and lowest cost
- **Smart System Software** – Specialized algorithms vastly improve all aspects of database processing: OLTP, Analytics, Consolidation
- **Automated Management** – Automation and optimization of configuration, updates, performance, and management culminating in Fully Autonomous Infrastructure and Database

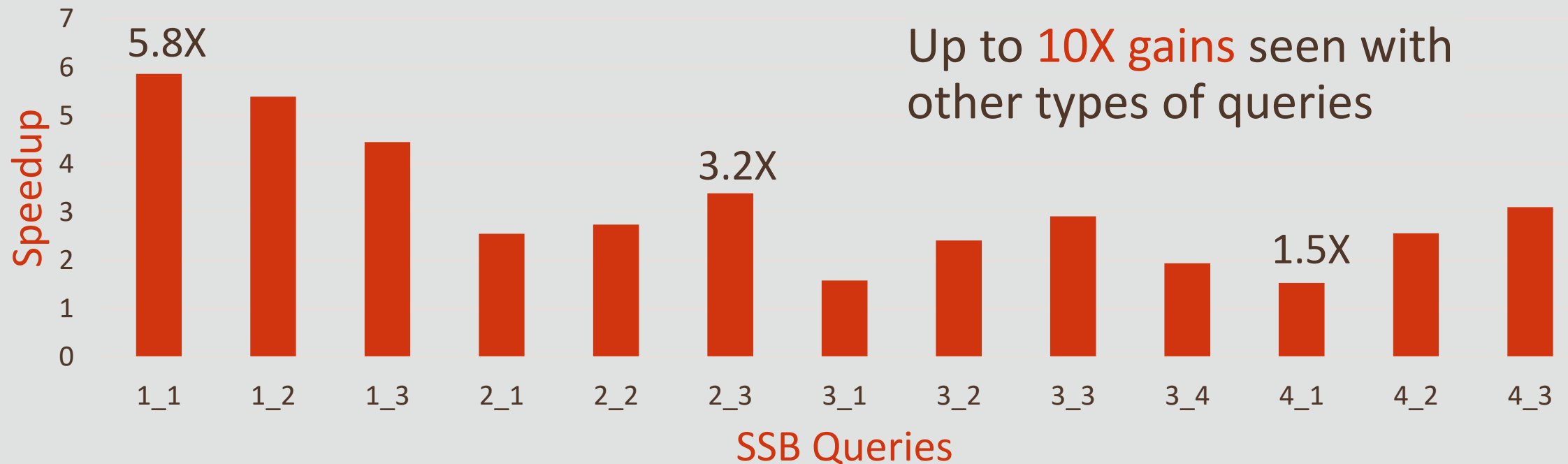
In-Memory **Accelerates** Exadata Flash Cache

- In-Memory format in Smart Columnar Flash
 - Enables **SAME** in-memory optimizations on data in Exadata flash as on Exadata DB compute nodes DRAM
 - Extends in-memory processing to Storage
 - **15x** Columnar Capacity (**100s** of TB on full rack)
- In-memory format – offloaded queries **10x faster**
 - **Huge advantage over other in-memory databases and storage arrays !!!!!**
- Completely automatic -no user intervention needed
 - Powers Autonomous Database



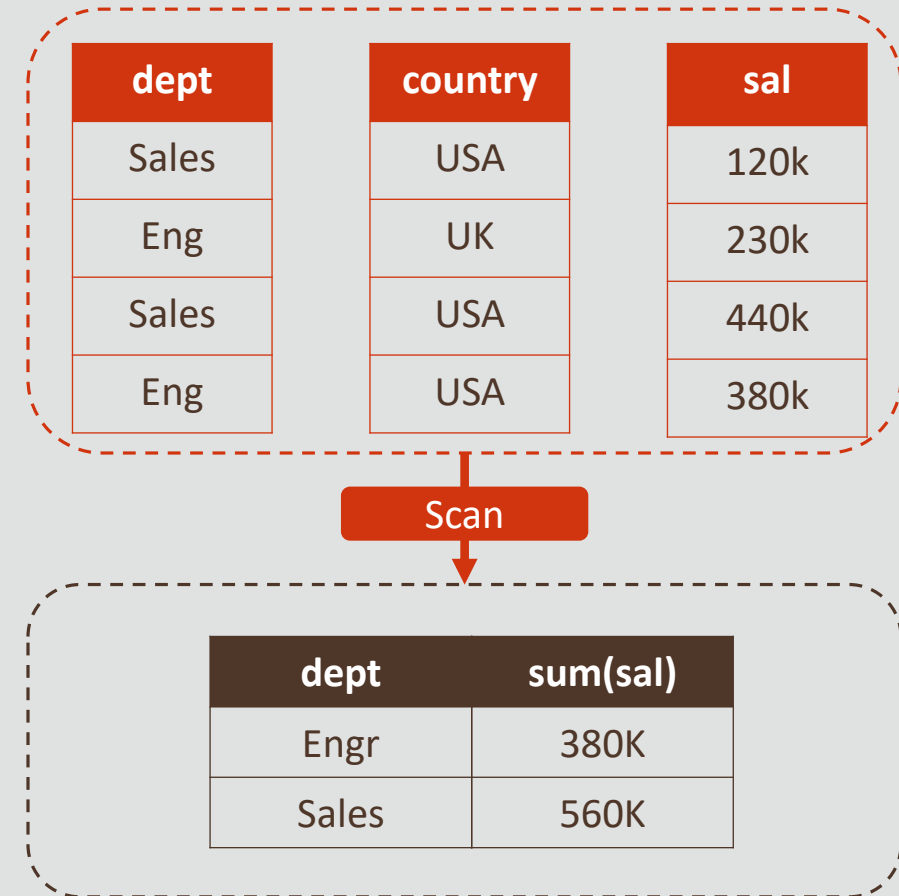
Database In-Memory Formats in Flash Cache (*CellMemory*) : Performance

CellMemory speedup over Columnar Flash Cache



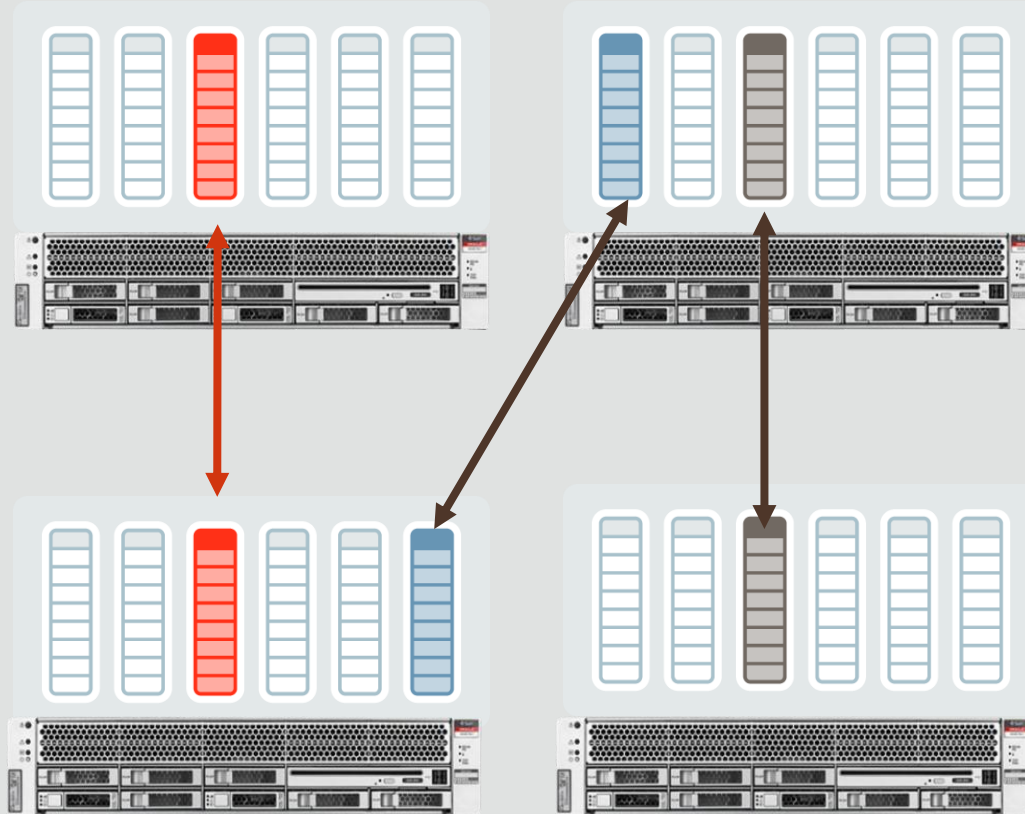
Example Benefit of In-Memory on Flash: Aggregation Offload

- In-Memory Format on Exadata Flash allows SUM and GROUP BY aggregations to be offloaded to storage servers:
 - Reduces data sent to the database server
 - Improves CPU utilization on the database server
- Example:
 - `select dept, sum(sal) from emp`
`where country='USA' group by dept`
 - Sum , group by operations performed on storage server
- **2x faster aggregation queries and reduced DB server CPU**



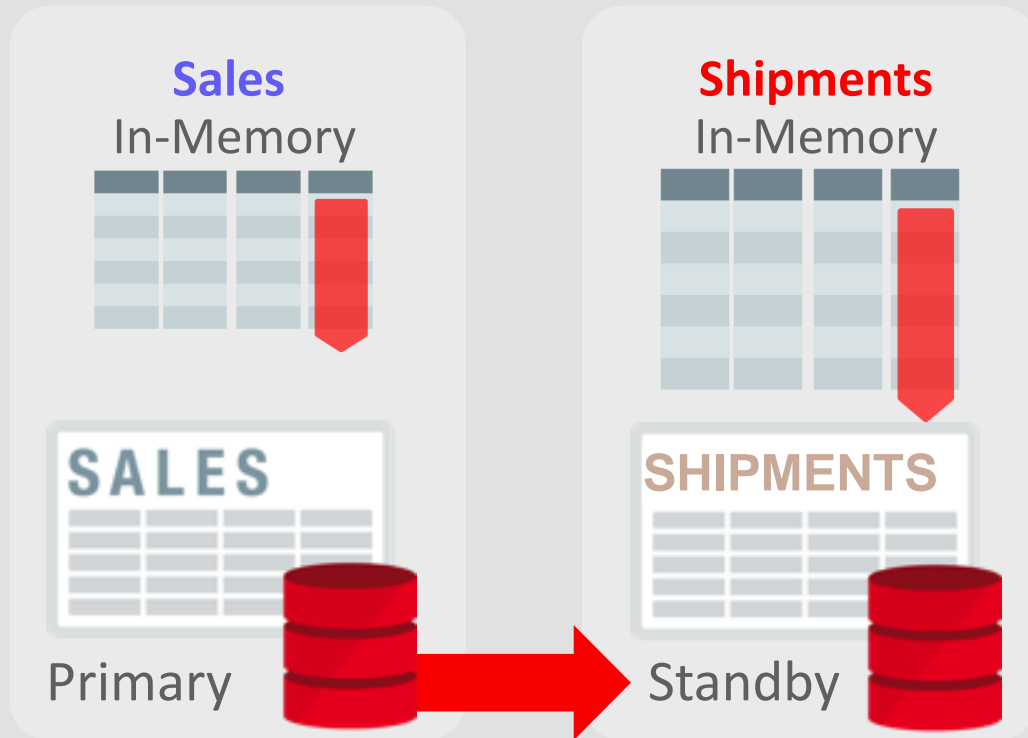
Enabled with Exadata System Software 19.3 for Database In-Memory customers, DB version 18.1 and later

In-Memory Duplication: Fault Tolerance and Performance



- Optionally duplicate in-memory columns across 2 nodes
 - Like storage mirroring
 - Can also duplicate across ALL nodes (e.g. small dimension tables)
 - Enabled per table/partition
 - Application transparent
- Eliminates column store repopulate after failure
- Improves performance due to greater locality

In-Memory on Active Data Guard



***Primary Database or Data Guard
Standby must be on Exadata***

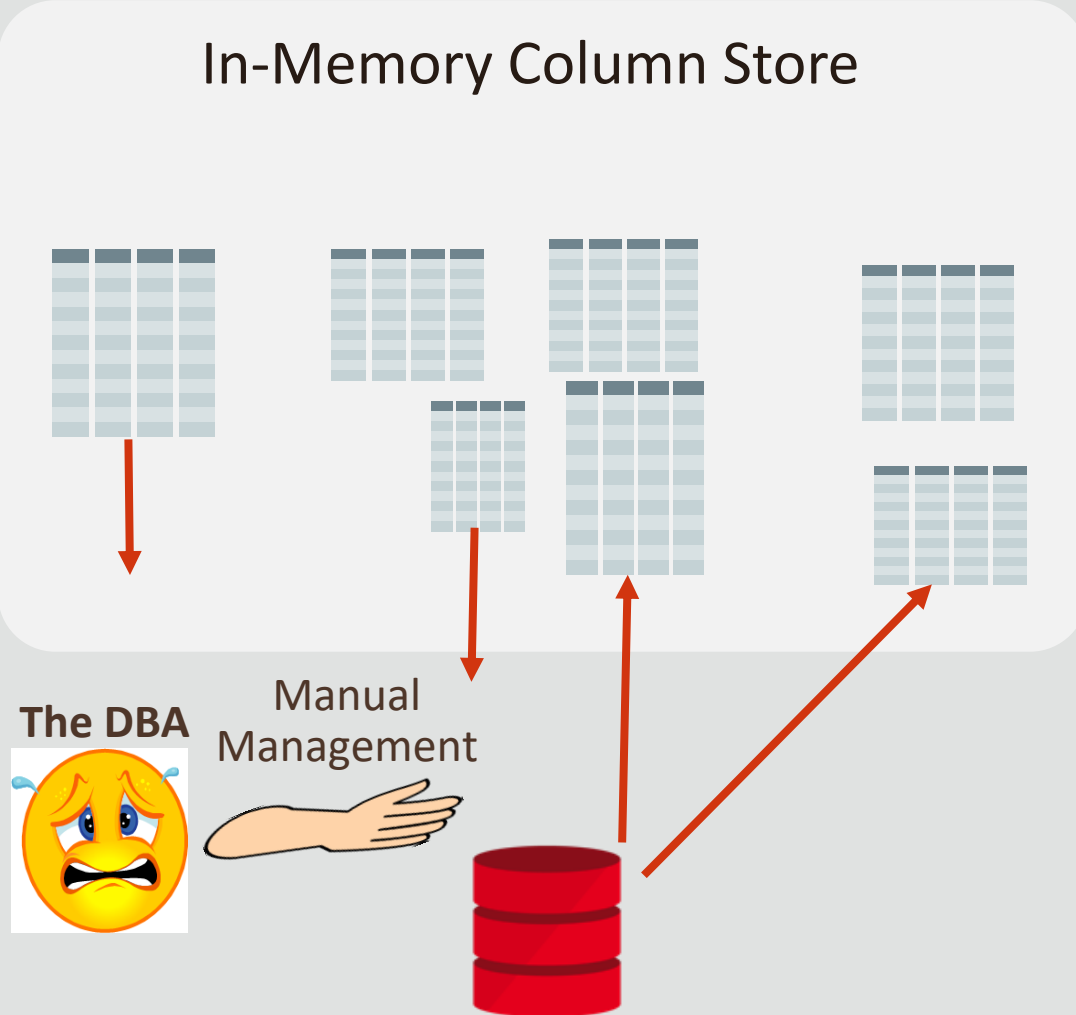
- Inmemory queries can run on Active Data Guard standby
 - No impact on primary database
 - Full use of standby database resources
- Standby can have different in-memory contents from Primary
 - Increases total effective inmemory columnar capacity
 - Increases column store availability:
 - Reporting workload on standby unaffected by primary site outage



#4 Intelligent Automation

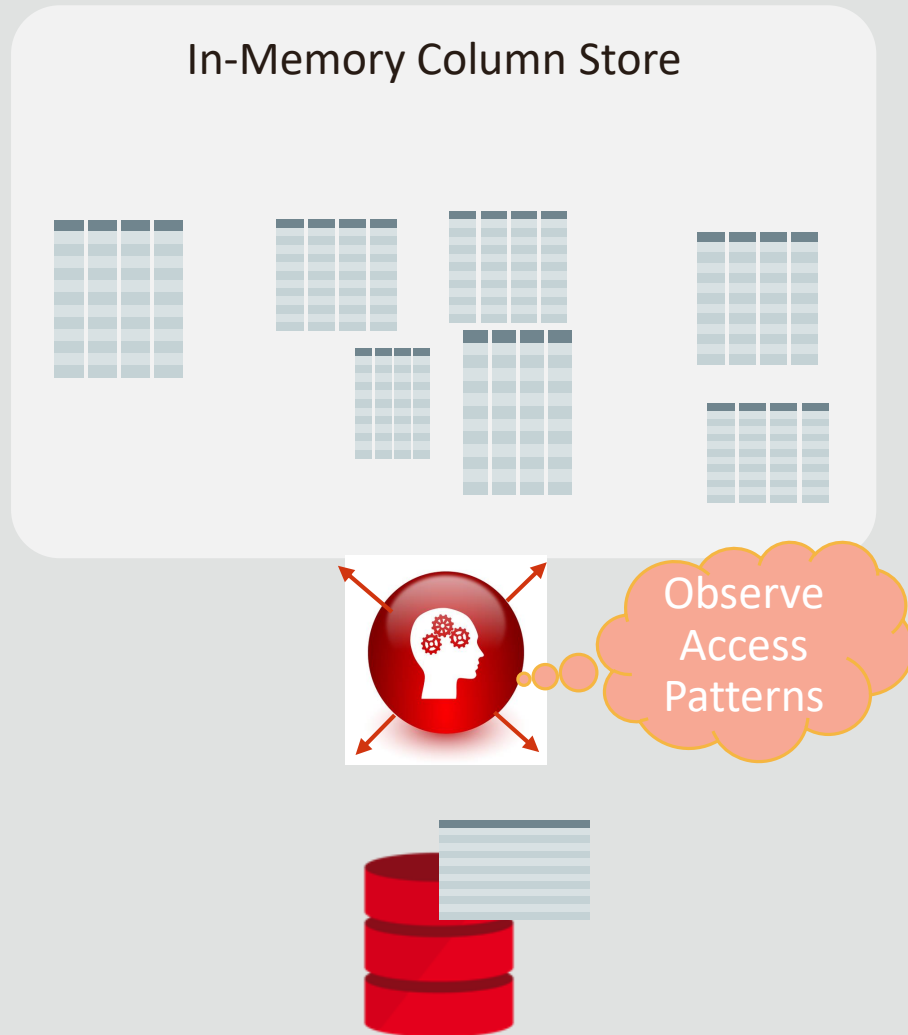
*Automatic In-Memory
Management & Storage Tiering*

Manual In-Memory Management



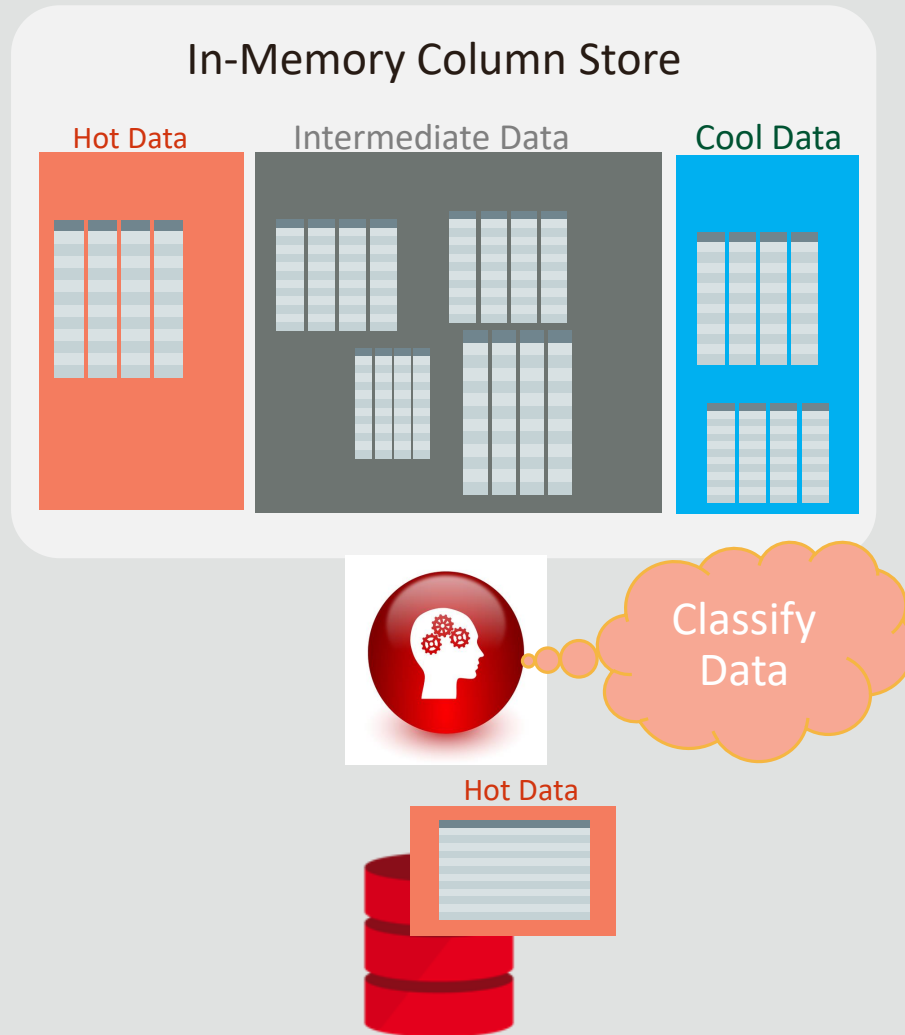
- If entire database fits within in-memory area, no need for DBA involvement!
- Otherwise, need to intelligently select in-memory candidates
- **Desired outcome:** Keep hot objects in-memory, remove colder objects
 - Access patterns are not known in advance and change over time
 - Hard for DBAs to achieve manually

Automatic In-Memory



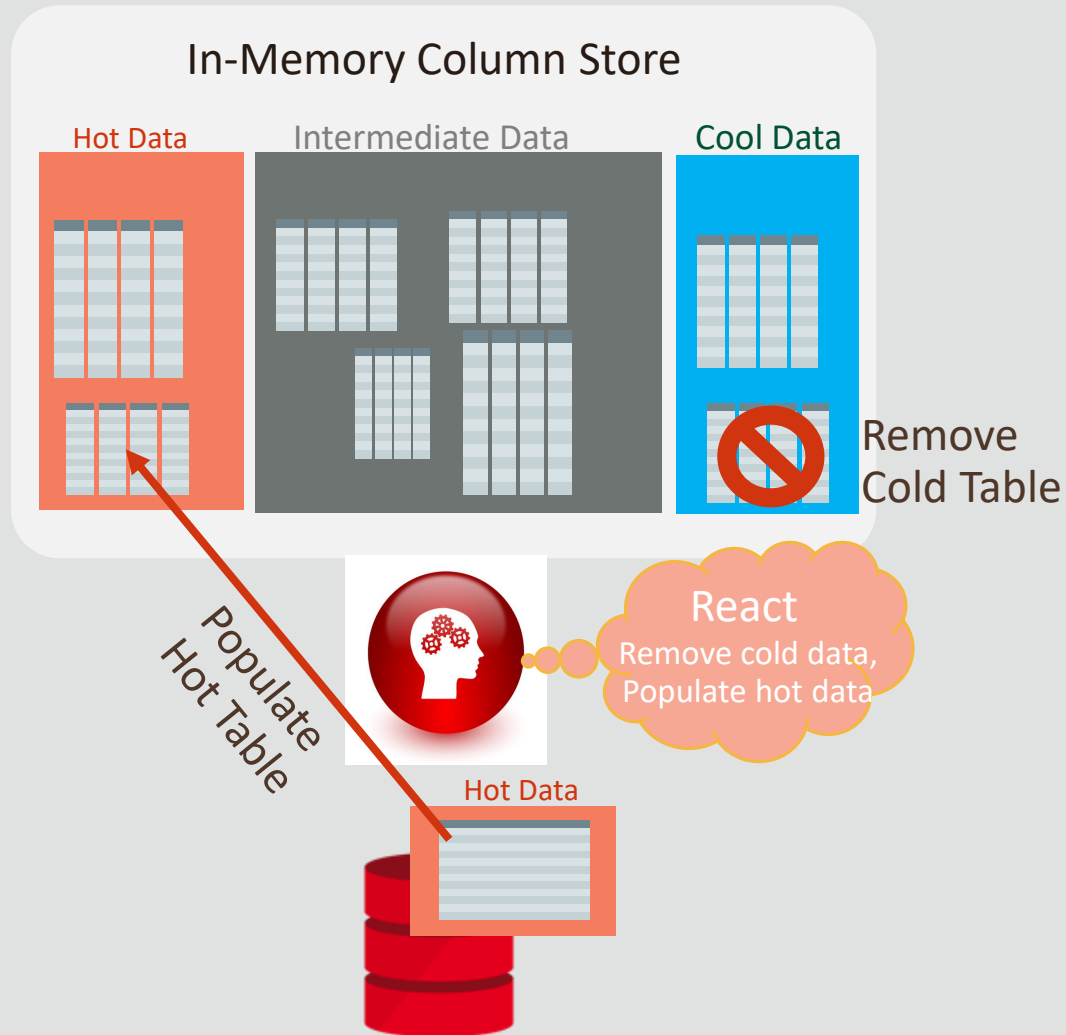
- Eliminates trial and error regarding in-memory area contents
- Constant background action:
 - Classifies data as hot, intermediate or cold
 - Hotter in-memory tables automatically populated
 - Colder in-memory tables automatically removed
 - Intelligent algorithm takes into account space-benefit tradeoffs
- Controlled by new parameter **inmemory_automatic_level**
- Useful for autonomous cloud services since no user intervention required

Automatic In-Memory



- Eliminates trial and error regarding in-memory area contents
- Constant background action:
 - Classifies data as hot, intermediate or cold
 - Hotter in-memory tables automatically populated
 - Colder in-memory tables automatically removed
 - Intelligent algorithm takes into account space-benefit tradeoffs
- Controlled by new parameter **inmemory_automatic_level**
- Useful for autonomous cloud services since no user intervention required

Automatic In-Memory



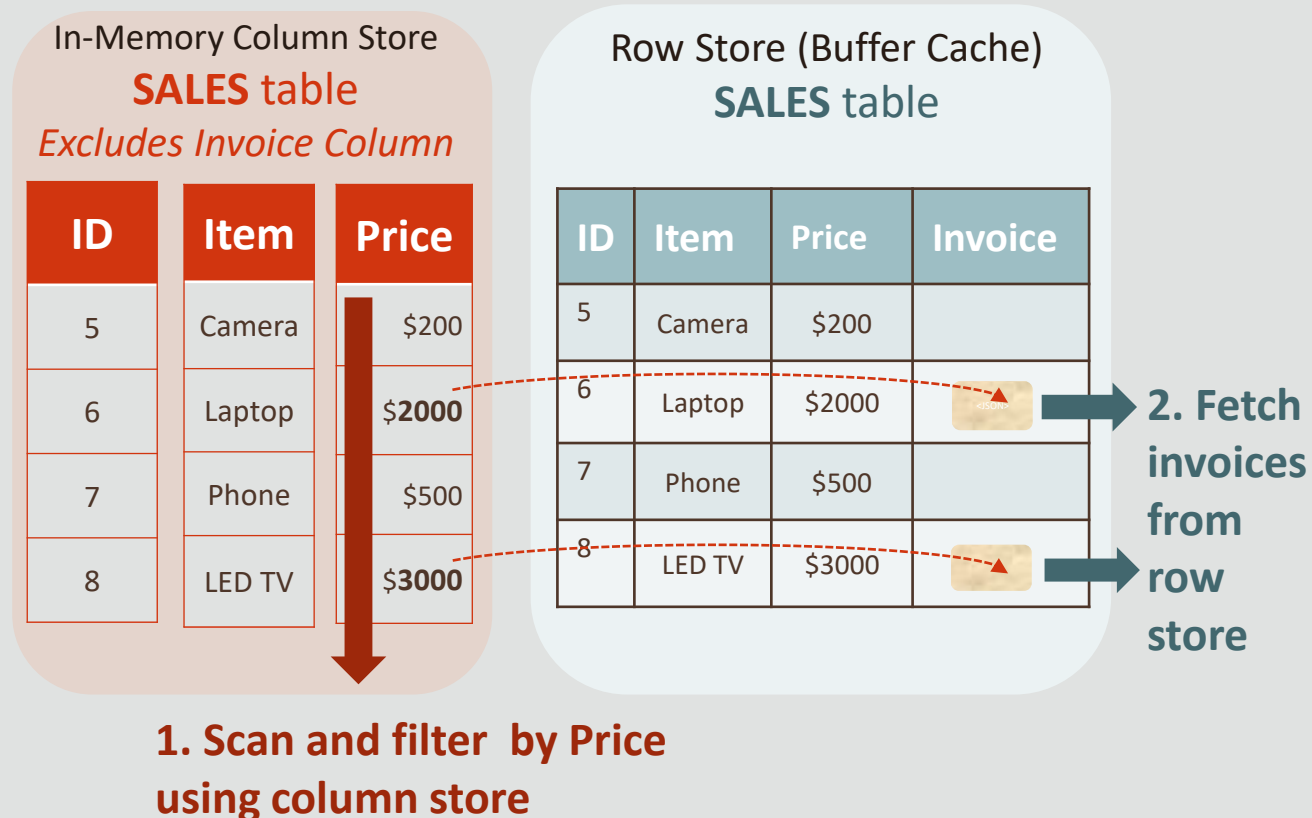
- Eliminates trial and error regarding in-memory area contents
- Constant background action:
 - Classifies data as hot, intermediate or cold
 - Hotter in-memory tables automatically populated
 - Colder in-memory tables automatically removed
 - Intelligent algorithm takes into account space-benefit tradeoffs
- Controlled by new parameter **inmemory_automatic_level**
- Useful for autonomous cloud services since no user intervention required

Preview | Hybrid In-Memory Scans



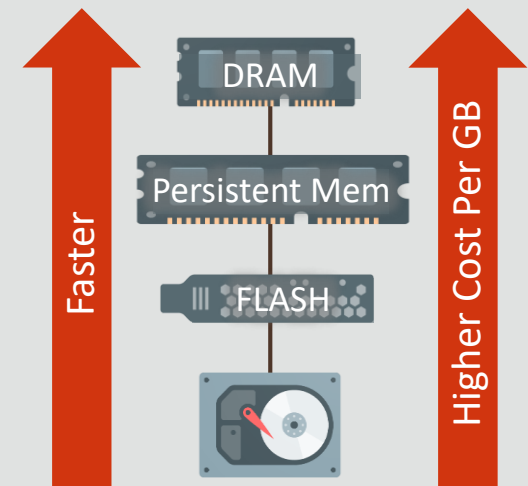
- Large, infrequently accessed columns can be excluded from the in-memory column store
 - e.g. Images, Documents, etc.
- **Current behavior:** In-Memory access disallowed if query accesses any excluded column
- **20c: Hybrid** In-Memory Scans
 - Scan/filter using in-memory column store
 - Fetch excluded column values from row store
 - Over **10x** performance improvement

```
SELECT Invoice FROM Sales
WHERE Price > 1000
```



Preview | Persistent Memory

- Persistent memory is a new silicon technology
 - Capacity, performance, and price are between DRAM and flash
- Intel® Optane™ DC Persistent Memory:
 - Reads at memory speed – much faster than flash
 - Writes survive power failure unlike DRAM
- Exadata implements sophisticated algorithms to maintain integrity of data on PMEM during failures
 - Call special instructions to flush data from CPU cache to PMEM
 - Complete or backout sequence of writes interrupted by a crash



Preview | Persistent Memory (In-Memory)

Today (Baseline)

- Not all data can fit into Memory
- Queries go against column store in DRAM and row store on DISK
- DRAM Dimms up to 128GB, and very expensive.



New : Intel® Optane™ DC Persistent Memory

- Entire workload can fit into Memory
- With Memory Mode, hottest tables are cached in DRAM for fastest access
- Apache Dimms up to 512GB



New : Intel® Optane™ DC Persistent Memory + Oracle 20c

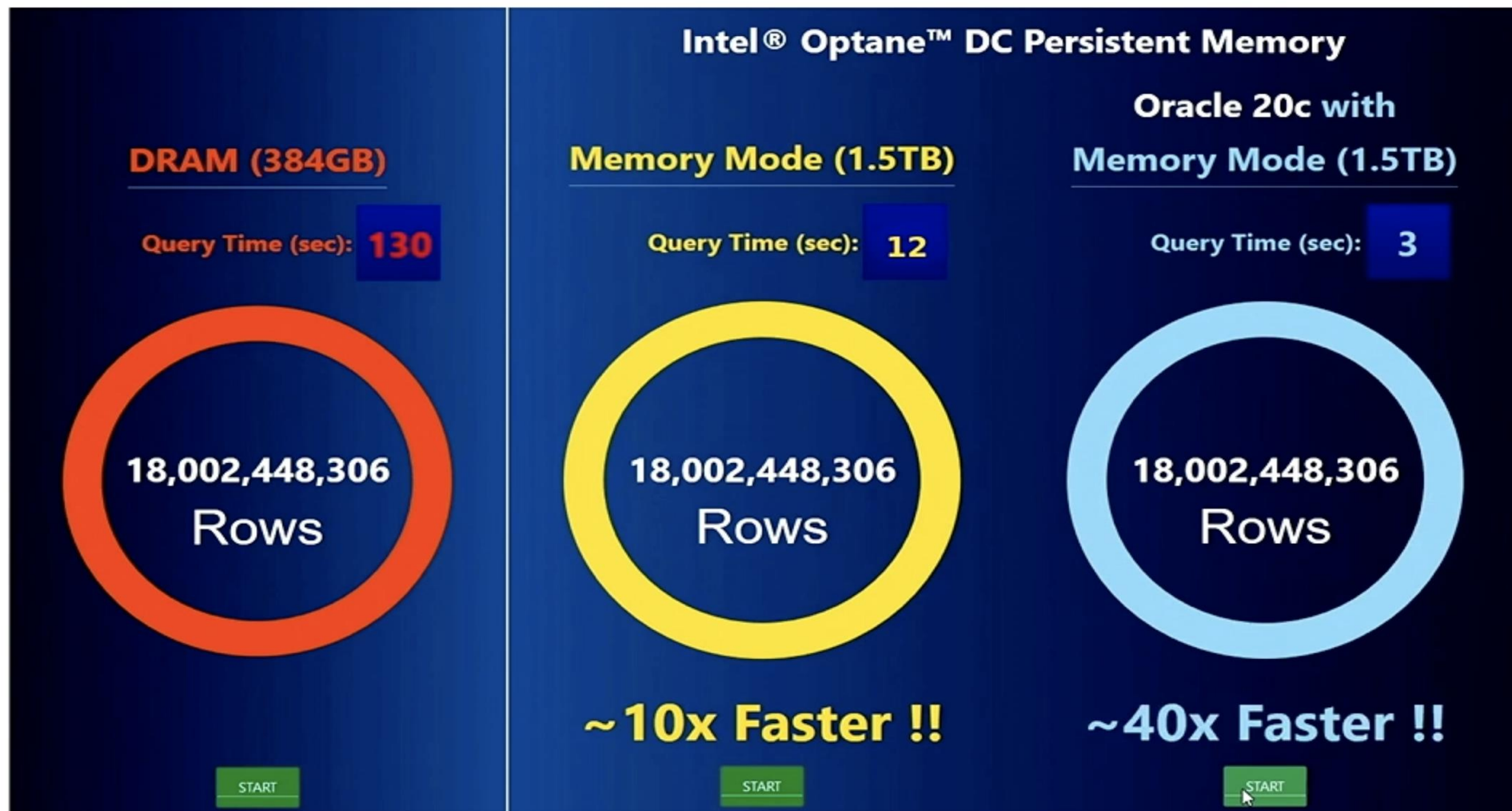
- **Oracle 20c** introduces new *Deep Vectorization* framework that extends vector processing to all SQL operators

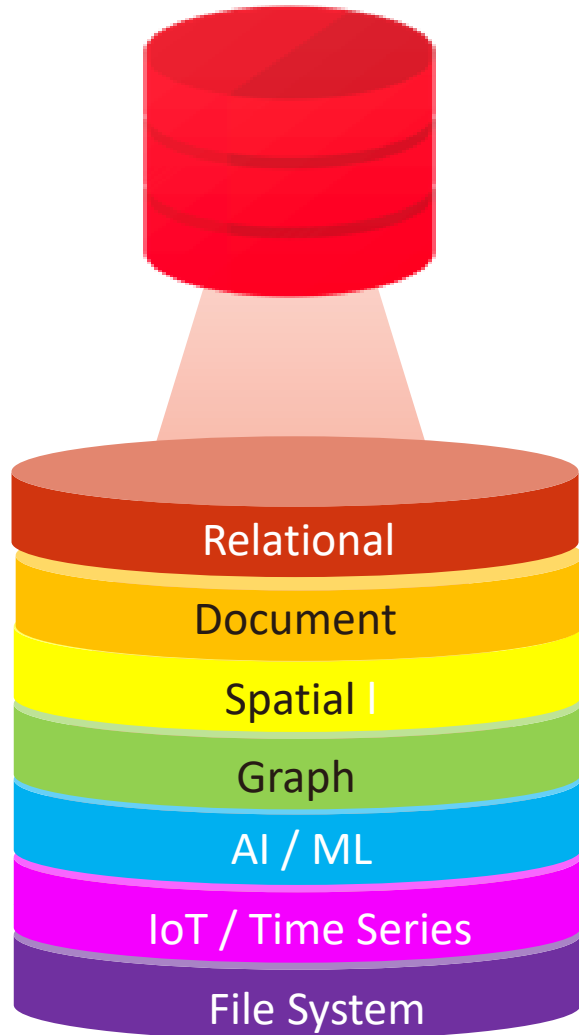


+



Intel[®] Optane[™] DC Persistent Memory with Oracle In-Memory Database

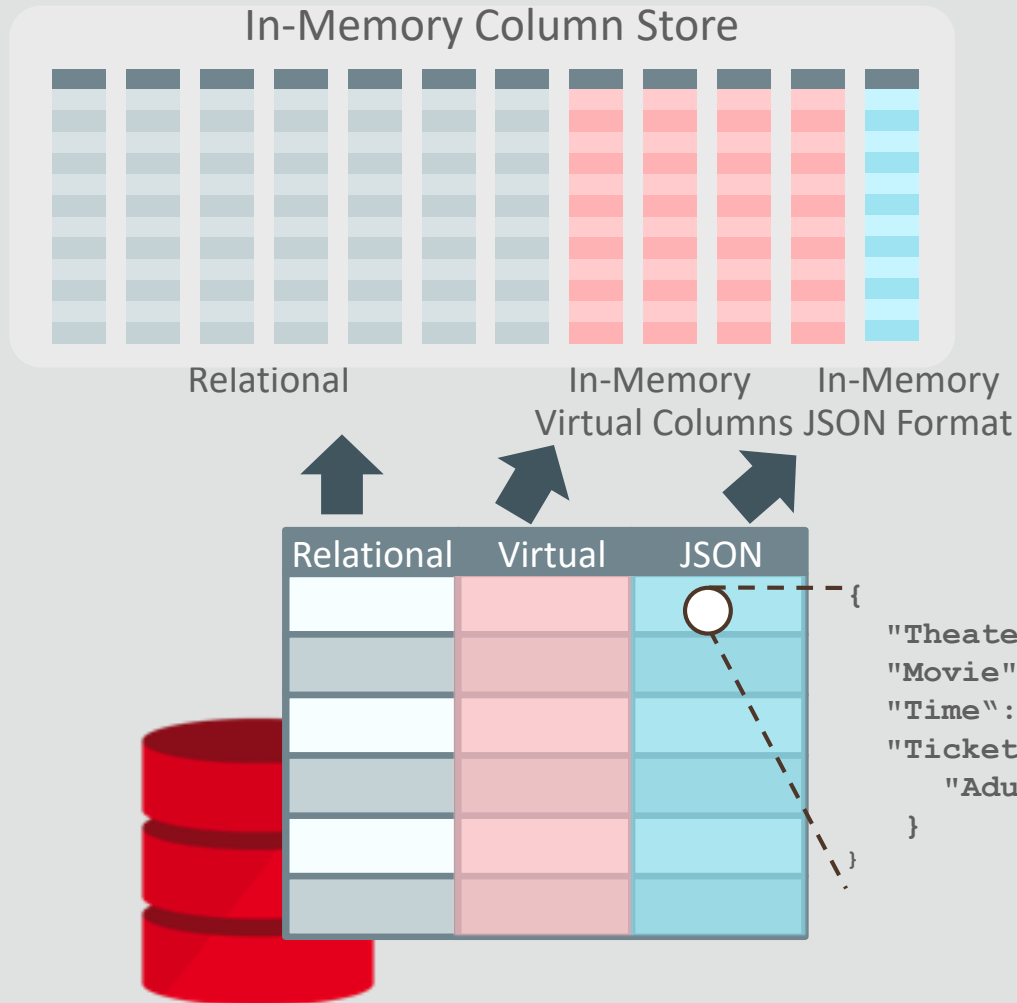




#5 Converged Analytics

*One Database for All : Relational,
Text, JSON, Spatial,...*

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

In-Memory For External Tables

Fast Analytics on External Data

External Tables allow transparent SQL on external data

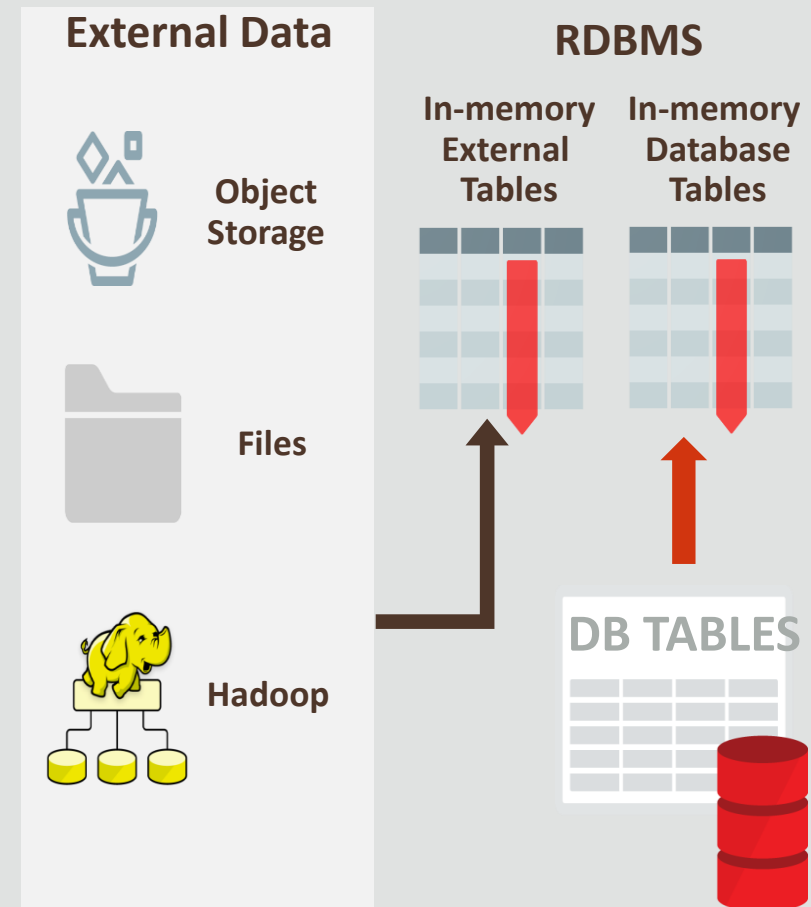
In-Memory External Tables: **100x faster** analytics on external data

All In-Memory Optimizations

Vector processing, JSON expressions extend transparently to external data

Simple to enable:


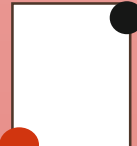
```
create table EXT1(...) organization
external (...) inmemory
```



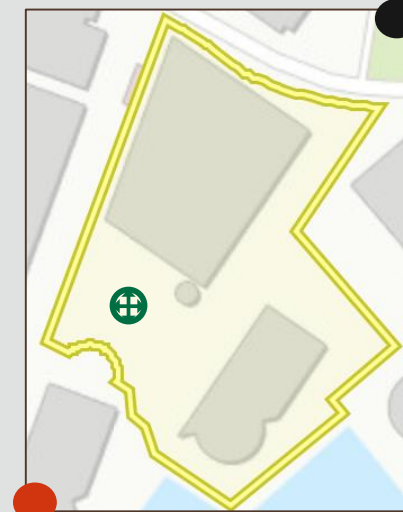
Preview | In-Memory Spatial Analytics

NEW IN
20^C

- In-Memory only *Spatial Summary* column added to each spatial column
 - Compact approximation of complex spatial detail
 - Stored in optimized In-Memory format
 - Quickly filter using SIMD vector scans
 - Replace R-Tree Indexes for Spatial Analytics
- Spatial Queries up to **10x** faster
 - No analytic R-tree index maintenance needed

In-Memory (IM) Table Columns			Additional IM columns
Parcel Number	Parcel Address	Spatial Details	Spatial Summary
095040390	300 Oracle Pkwy		
095040310	400 Oracle Pkwy		
095040250	500 Oracle Pkwy		
095040260	600 Oracle Pkwy		

Which parcel is the utility valve located in?



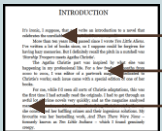


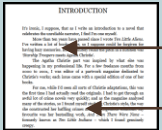
Search 140 Million
US land parcels

Preview | In-Memory Text Analytics

NEW IN
20^C

Find job candidates with "PhD" degrees who have "database" in their resumes

In-Memory Column Store

Name	Degree	Resume (Text)	Text Index
			Words
John	PhD		..
Ram	BS		..
Emily	MS		database
Sara	MS		..

- In-Memory only *Inverted Index* added to each text column
 - Maps words to documents which *contain* those words
 - Replaces on-disk text index for analytic workloads
- Converged queries (*relational + text*) can benefit from in-memory
 - 3x faster

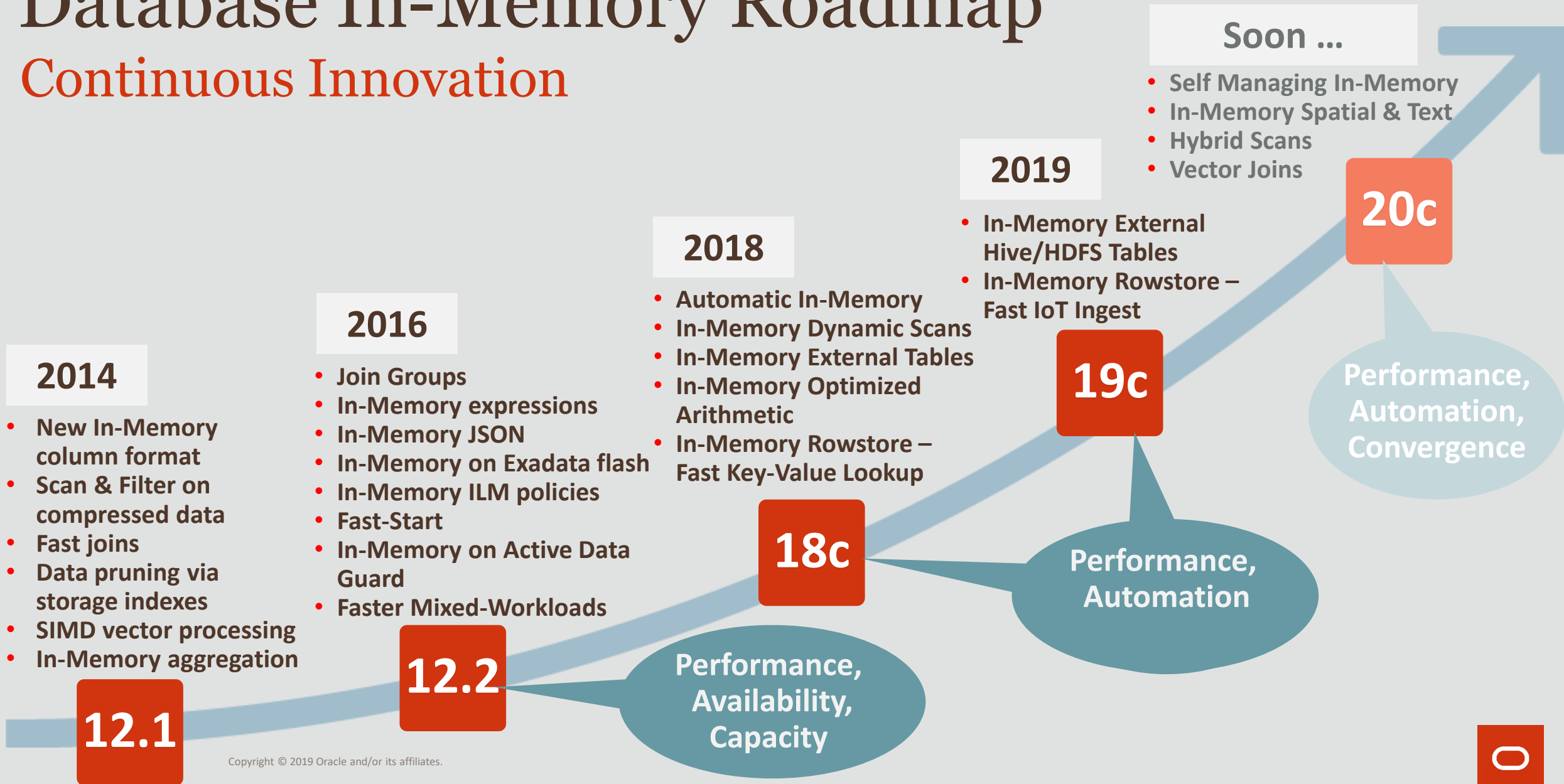




Top-5 Innovations Summary

Database In-Memory Roadmap

Continuous Innovation

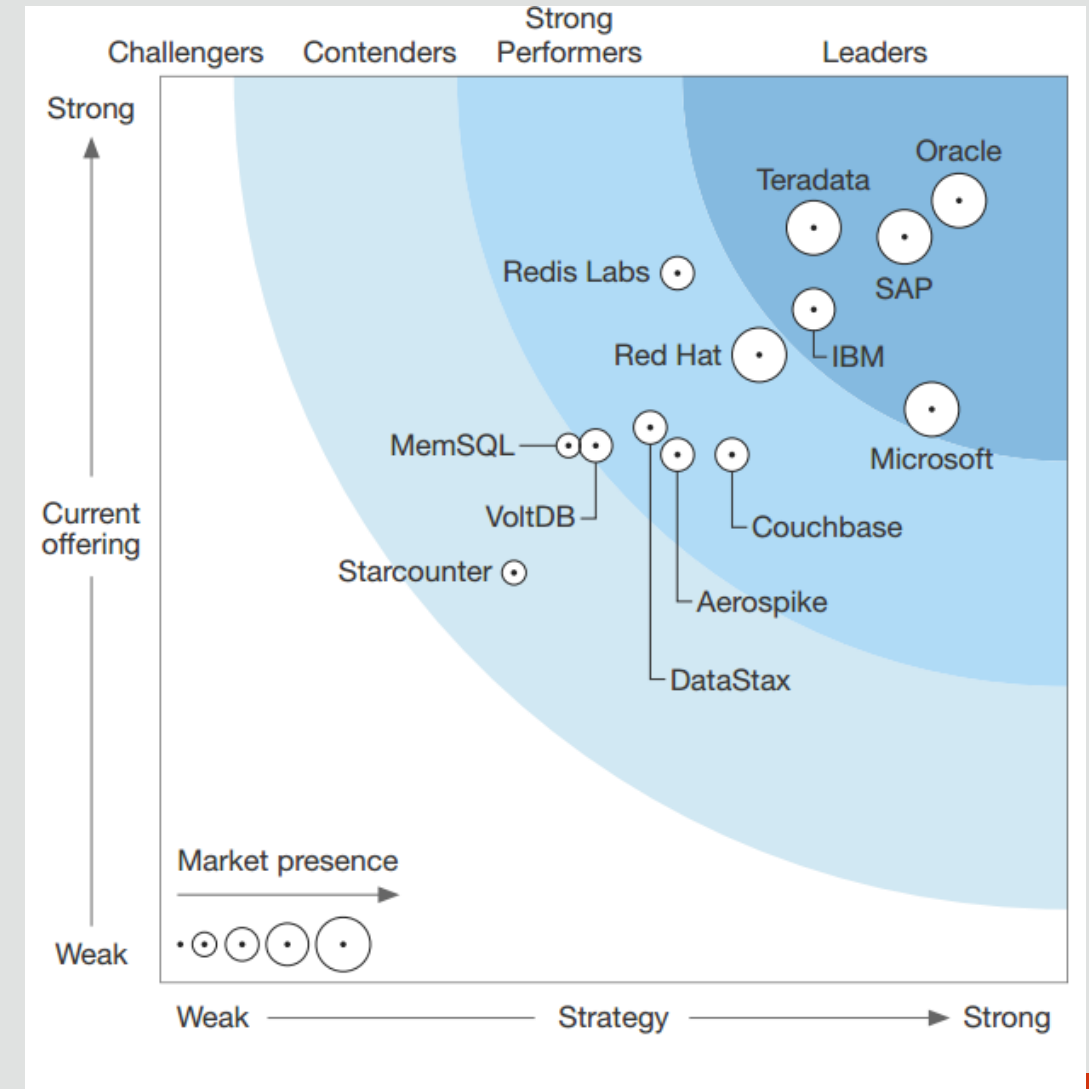


The Forrester Wave™: In-Memory Databases, Q1 2017

**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.



Hot off the Press: 2019 Forrester Wave Translytical Data Platforms

Oracle Position: Leader

Oracle ranked highest on both Axis

- “Unlike other vendors, Oracle uses a **dual-format database** (row and columns for the same table) to deliver optimal translytical performance.”
- “Customers like Oracle’s capability to support many workloads including OLTP, IoT, microservices, **multimodel**, data science, AI/ML, spatial, graph, and analytics”
- “Existing Oracle applications **do not require any changes** to the application in order to leverage Oracle Database In-Memory”

THE FORRESTER WAVE™

Translytical Data Platforms

Q4 2019



Additional Resources

Related White Papers

- [Oracle Database In-Memory White Paper](#)
- [Oracle Database In-Memory Aggregation Paper](#)
- [When to use Oracle Database In-Memory](#)
- [Oracle Database In-Memory Advisor](#)
- [SQL Plan Management White Paper](#)
- [POC / Implementation Guidelines](#)

Related Videos





- [In-Memory YouTube Channel](#)
- [Managing Oracle Database In-Memory](#)
- [Database In-Memory and Oracle Multitenant](#)
- [Industry Experts Discuss Oracle Database In-Memory](#)
- [Software on Silicon](#)



Additional Details

- [Oracle Database In-Memory Blog](#)
- [Optimizer blog](#)

Join the Conversation

-  [@dbinmemory](#)
-  <https://blogs.oracle.com/in-memory/>
-  <https://www.facebook.com/OracleDatabase>
-  <http://www.oracle.com/goto/dbim.html>



ORACLE