

ORACLE

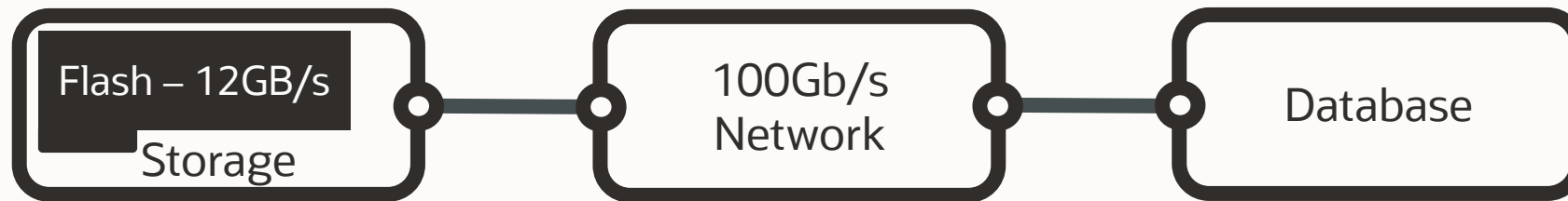
Oracle Database – New Performance Features in 23ai

Danica Porobic
Consulting Member of Technical Staff
Data, In-Memory and AI Technologies



How to achieve high performance?

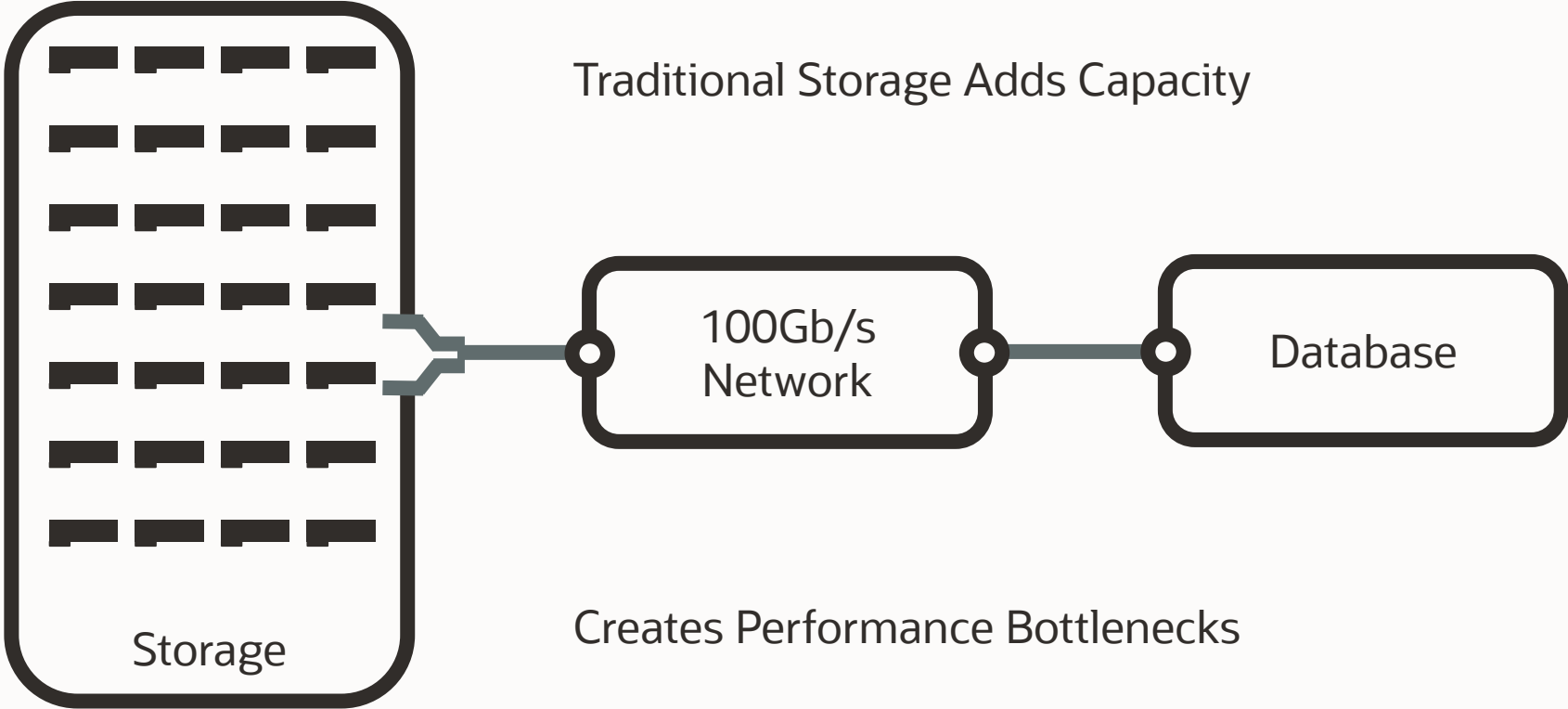
Storage Networking Background



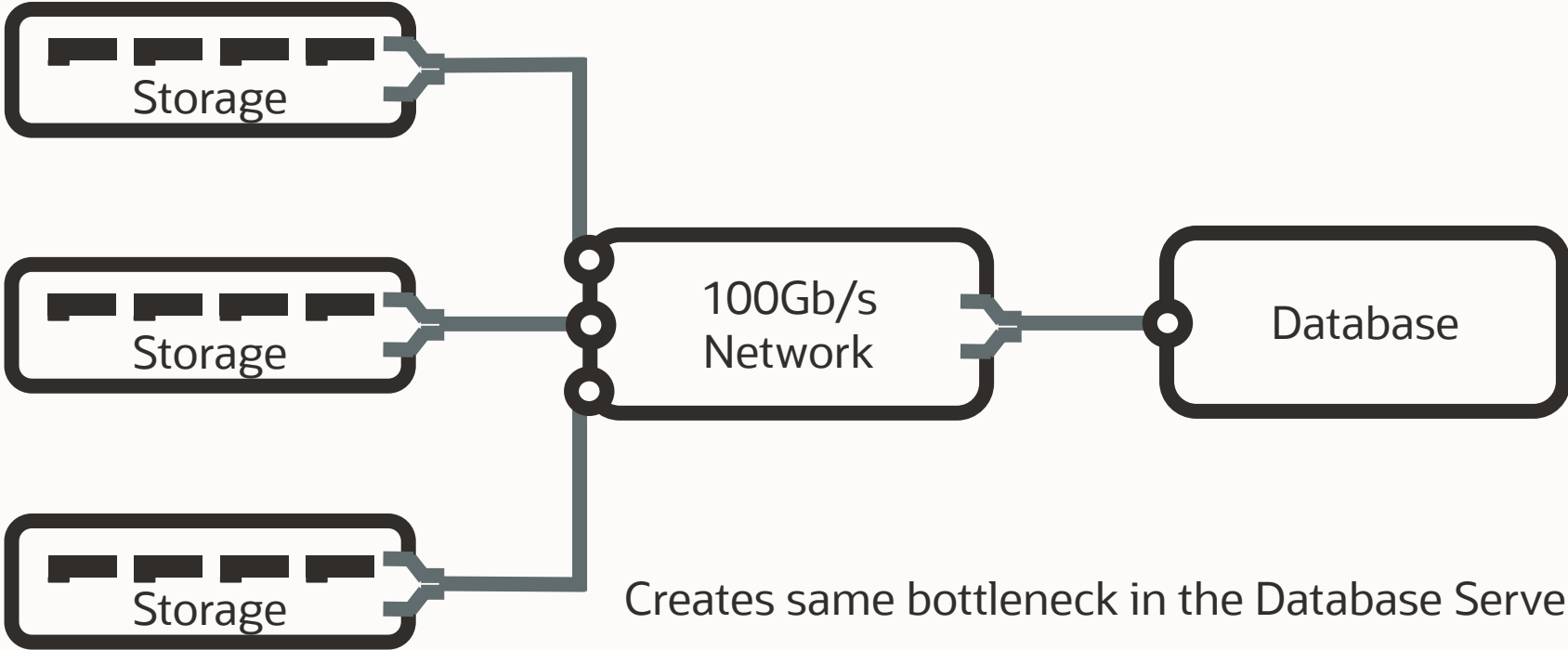
One flash card can saturate a 100Gb Network



Traditional Monolithic Storage Bottleneck



Cloud Storage Bottleneck

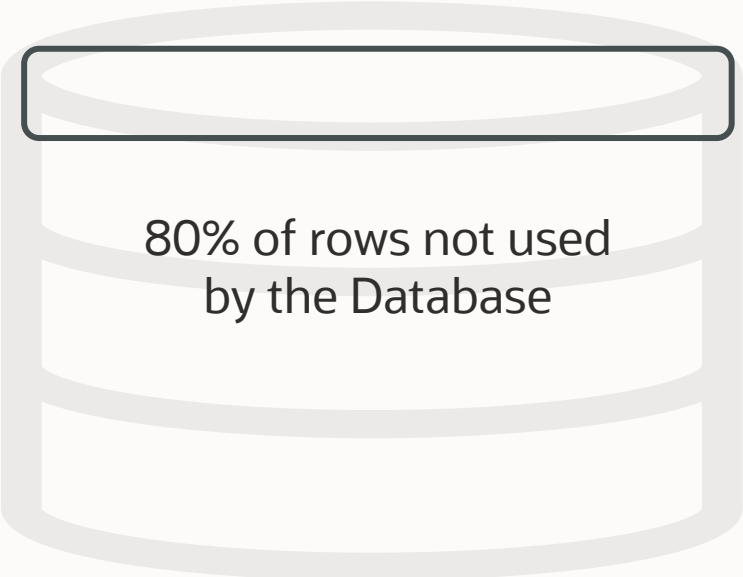


Traditional Scan

Find all Customers who ordered an item over \$1000

Customer	Price	Order ID
Alice	2000	101
Bob	800	102
John	200	103
Jane	125	104
Frank	200	105

Storage



Database



Smart Scan

Row Filtering and Column Projection

Find all Customers who ordered an item over \$1000

Customer	Price	Order ID
Alice	2000	101
Bob	800	102
John	200	103
Jane	125	104
Frank	200	105

Exadata Storage



Storage Index

Find all Chips sales ordered between April 7 and April 13

Item	Order Date	Ship Date	Customer
Salsa	March 19	March 20	Alice
Chips	March 21	March 22	John
Coffee	March 23	March 23	Bob
Milk	March 24	March 25	Jane

No I/Os performed for this query

Query Answered by Storage Index

Exadata Storage

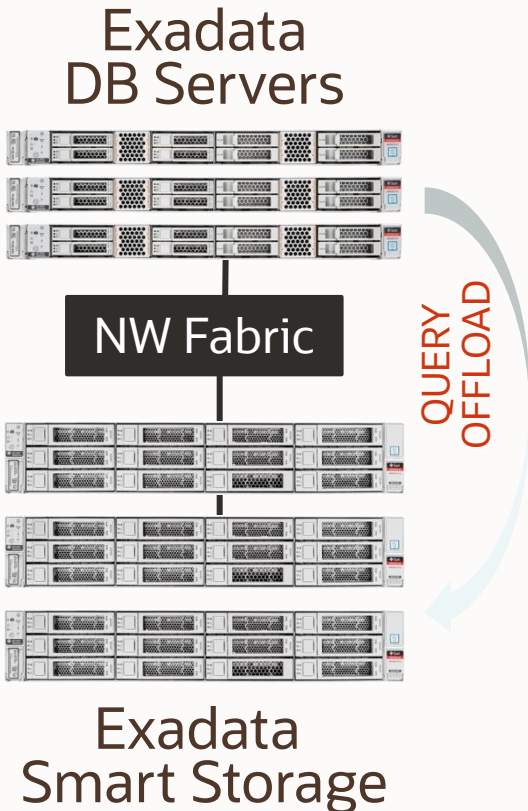
Order Date Min	Order Date Max
March 19	March 21

Order Date Min	Order Date Max
March 23	March 24

Storage Index



Exadata Uniquely Achieves Memory Speed with Shared Flash



Architecturally, storage arrays can share flash *capacity* but not flash *performance* due to network bottlenecks

- Even with next gen scale-out, PCIe networks, NVMe over fabric

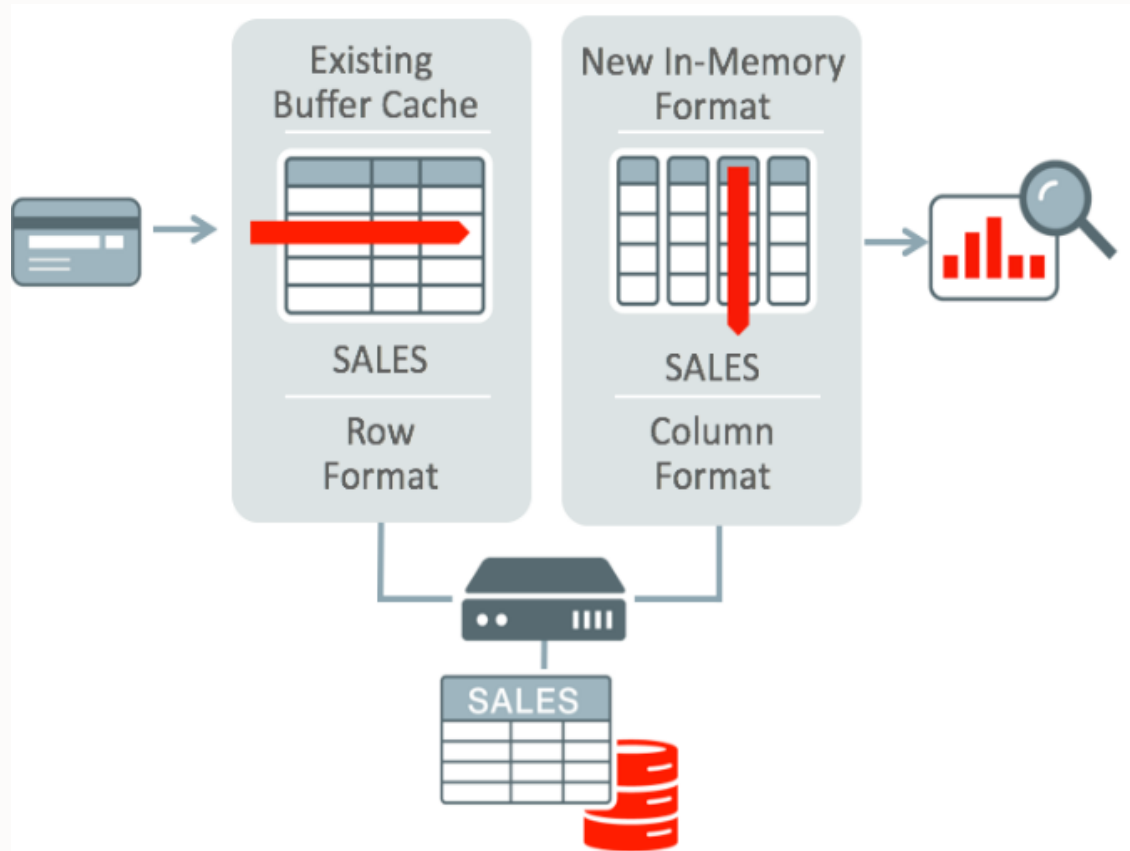
Must move compute to data to achieve full flash potential

- Requires owning full stack; can't be solved in storage alone

X8M delivers 560 GB/sec flash bandwidth to any server

- Approaches 800 GB/sec aggregate DRAM bandwidth of DB servers

Oracle Database In-Memory: **Architecture**



Dual-Format Architecture

- Both row and column formats for table
 - Transactions benefit with existing row format
 - Analytics benefit with In-Memory columnar format
- Simultaneously active and consistent

Blazing Fast Analytic Scans

- SIMD on Compressed Columnar Data Formats

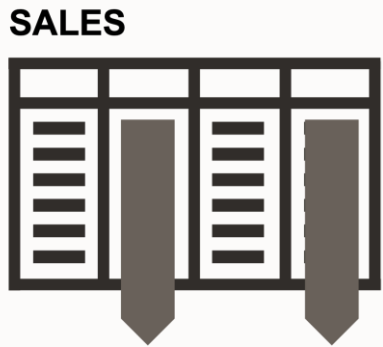
Seamlessly built into Oracle RDBMS

- RAC, Dataguard, Flashback, etc.

Database In-Memory Technology

Scanning and filtering data more efficiently

Columnar Format



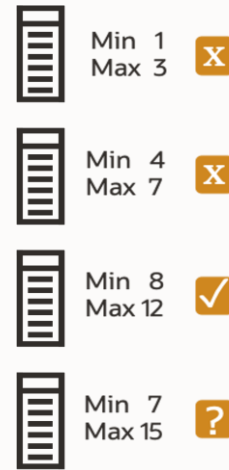
Access only the columns you need

Compression



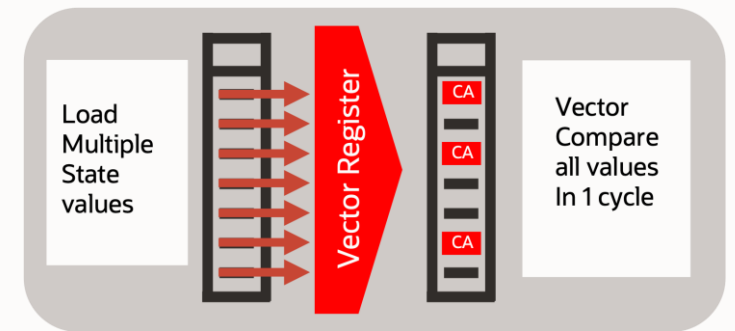
Scan & filter data in compressed format

Storage Indexes



Prune out any unnecessary data from the column

SIMD Vector Processing



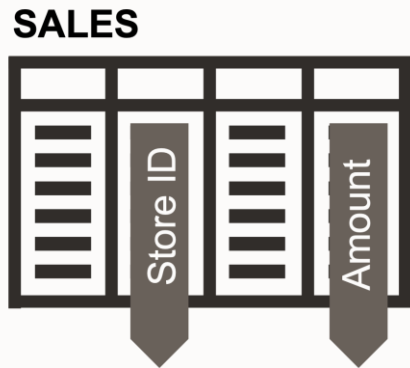
Process multiple column values in a single CPU instruction

Billions of rows/sec scan rate per CPU core

Query Engine Enhancements

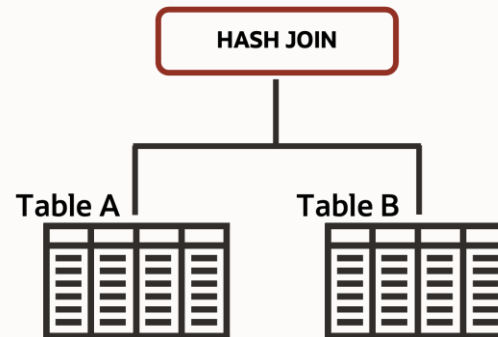
Improves all aspects of analytic queries

Data Scans



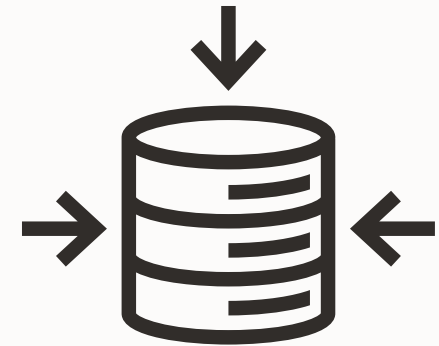
- Speed of memory
- Scan and Filter only the needed Columns
- Vector Instructions

Joins



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

In-Memory Aggregation



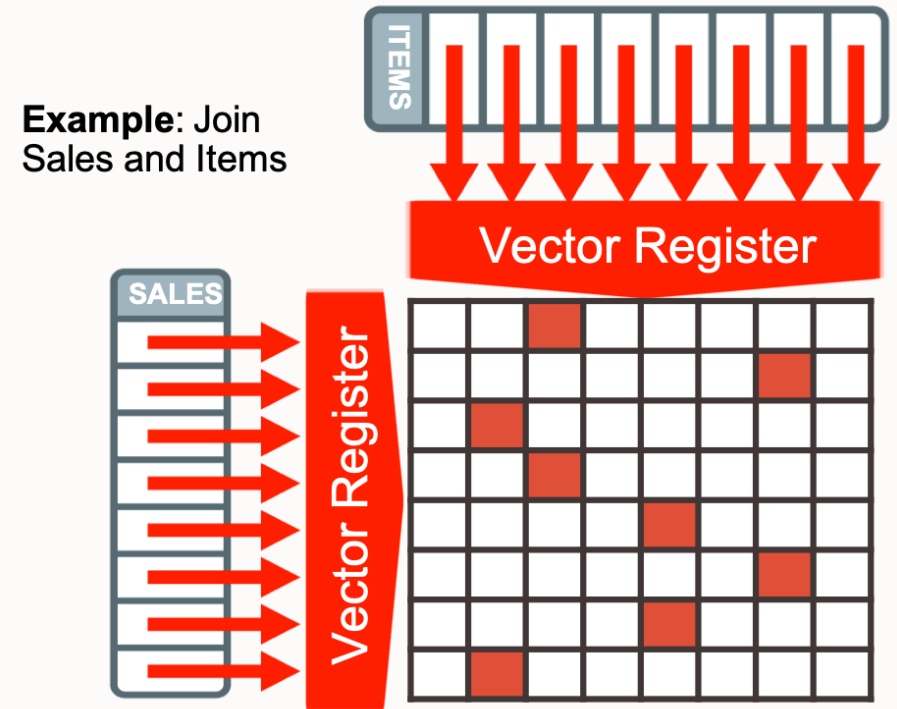
- Create In-Memory Report Outline that is Populated during Fast Scan
- Runs Reports Instantly

New In-Memory Features



In-Memory Vectorized Joins

- In-Memory Vectorized Joins uses a new *In-Memory Vectorization framework* to accelerate **Complex Joins**
- Optimizes Hash Join and Group By Aggregations
 - Performs join and aggregation operations during in-memory scans
 - Utilizes Join Groups for enhanced performance
 - Match multiple rows between SALES and ITEMS tables in a single SIMD Vector Instruction
- Leverages IMDS for increased performance
- Can improve performance by up to 15x versus a non-IM hash join



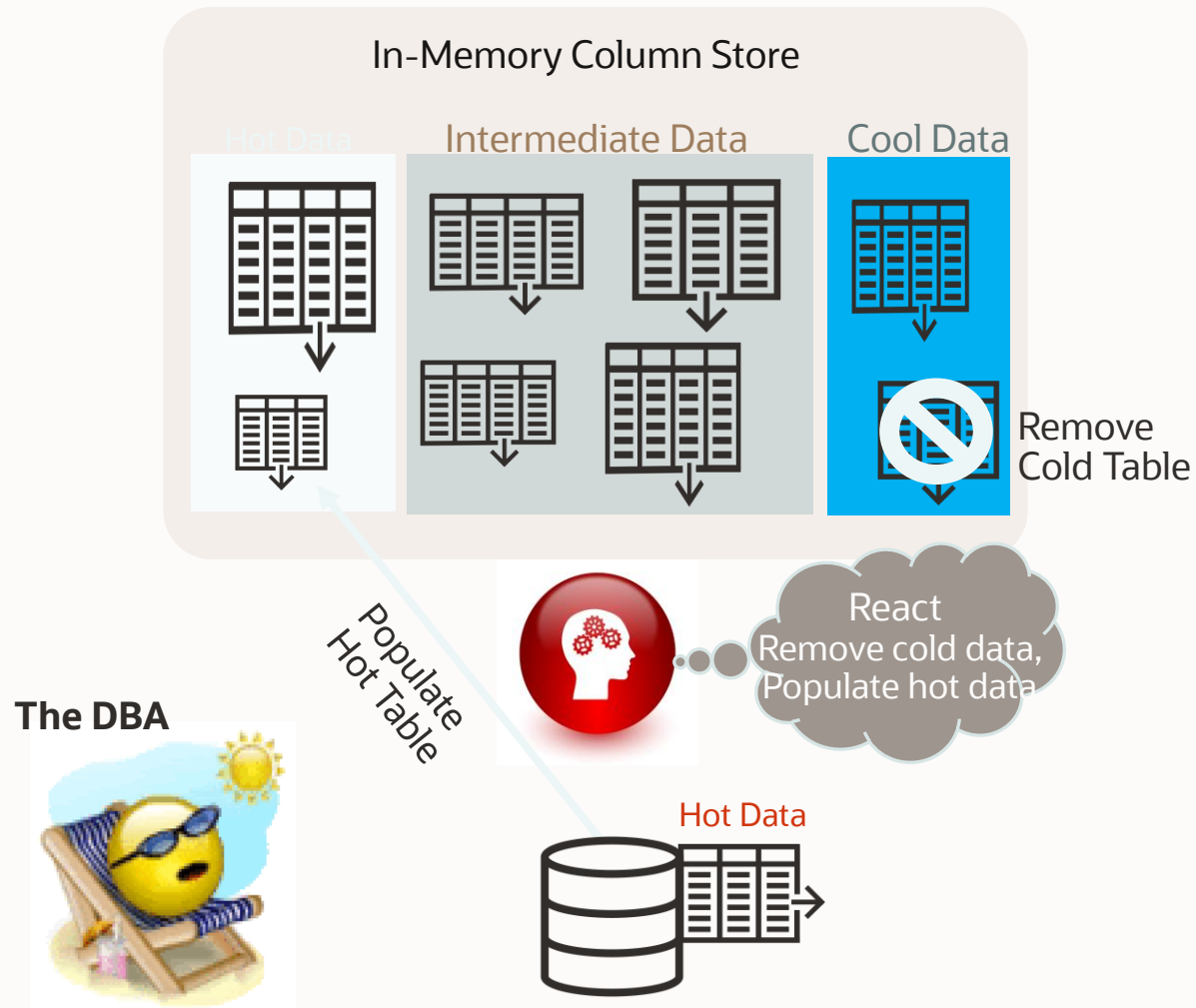
In-Memory Vectorization Framework: Multi-Level Joins and Aggregations

- In Oracle Database 23ai the In-Memory Deep Vectorization Framework adds more join support:
 - Multiple Join Keys
 - Semi and Outer Join
 - Enhanced Grouping and Aggregation
 - Multiple Levels of Joins
- This translates to faster join performance by utilizing SIMD optimizations during join processing

In-Memory RAC-Level Global Dictionary

- Enables common dictionary codes across RAC nodes
 - Previously dictionary codes were only common for each instance
- Improved performance for distributed hash joins
 - Enables RAC global codes for distributed hash joins
 - Enables the use of Join Group aware distributed hash joins

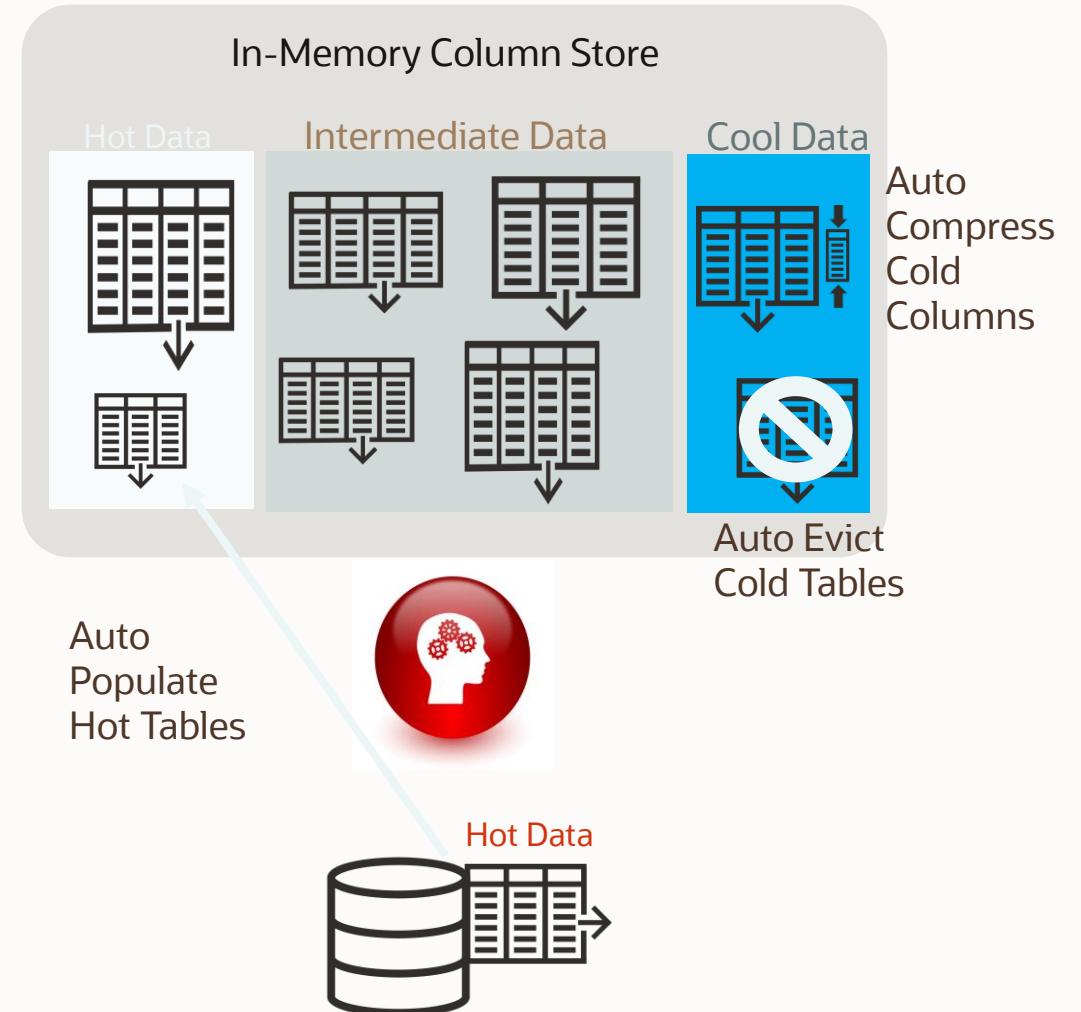
Automatic In-Memory



- Introduced in 18c
- Helps eliminate 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`**

AIM: Self-Managing In-Memory (21c)

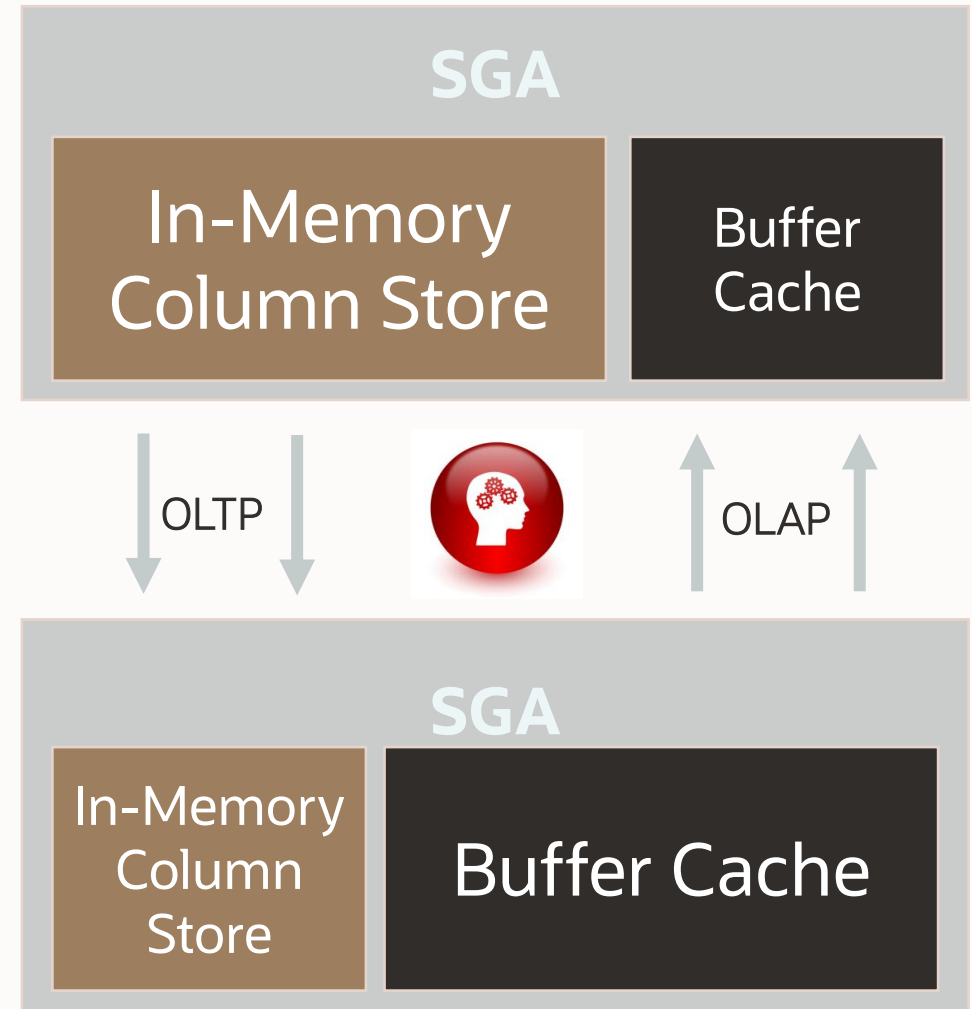
- New Automatic In-Memory (AIM) option
 - `inmemory_automatic_level = HIGH`
- AIM enables self-managing in-memory column store
 - No need to mark tables INMEMORY
- Automatically manages objects
 - Intelligent population and eviction without user input
 - Automatically compresses less frequently accessed in-memory columns



Automatic In-Memory Sizing

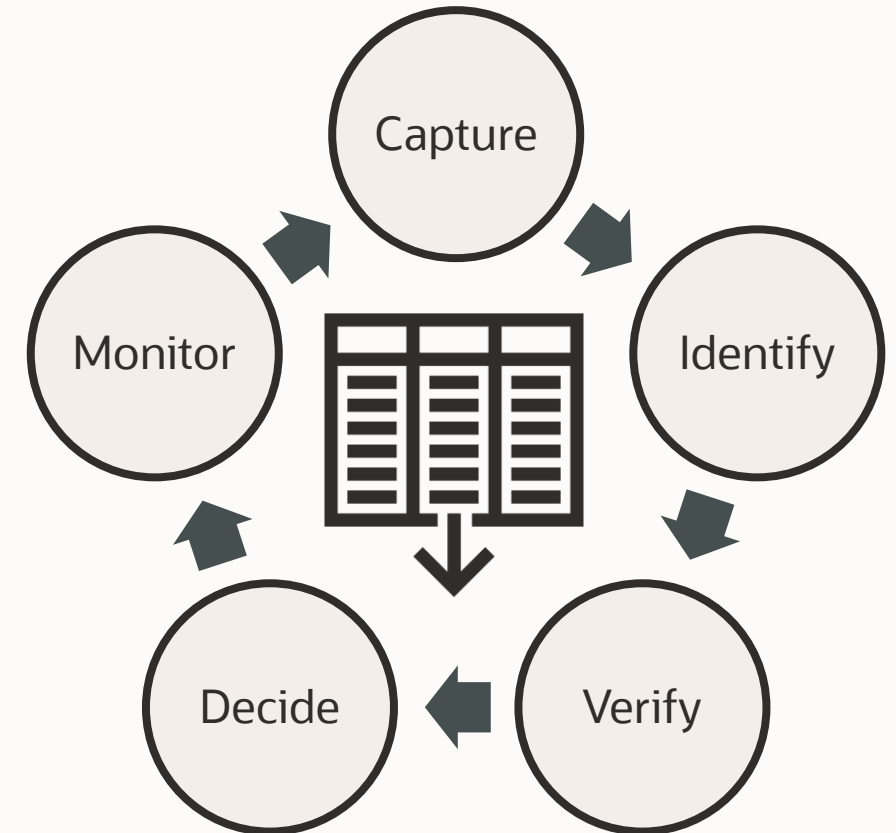
Automatic Sizing of the In-Memory Column Store

- Auto-size column store to factor in different workload types sharing resources at the same time
 - Shrink column store and increase buffer cache if DML intensive workload predicted.
 - Grow column store and shrink buffer cache if OLAP intensive workload predicted
- Works with Automatic In-Memory (AIM)
 - Requires AIM level HIGH and ASMM to be enabled (i.e. SGA_TARGET)
- The INMEMORY_SIZE parameter sets the minimum IM column store size



Automatic Enablement of In-Memory Features

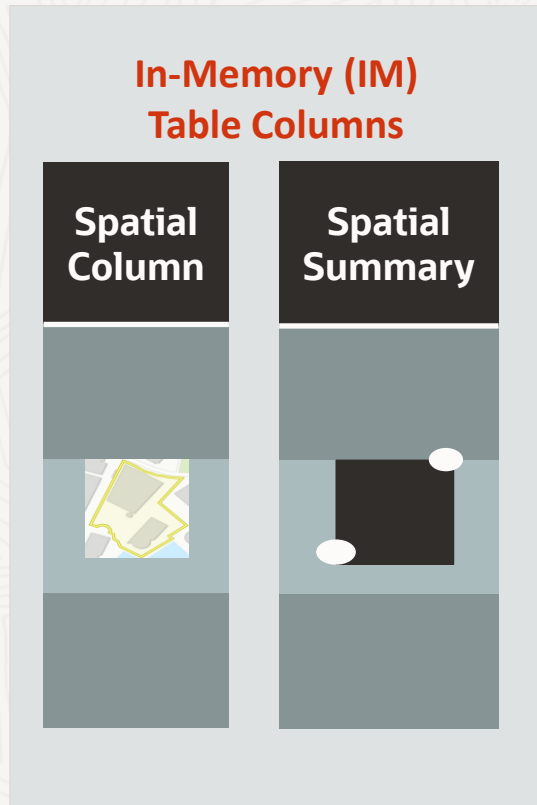
- In Oracle Database 23ai AIM has been enhanced to add the ability to automatically:
 - Enable Join Groups where beneficial
 - Enable In-Memory Optimized Arithmetic for beneficial number columns
 - Enable higher compression levels on specific columns
 - Unpopulate unused columns
- Reduces manual effort required to leverage key performance features
- Enhanced workload analysis to better account for mixed workload environments
 - DML overhead with In-Memory (for example, fetching invalid rows in column store from buffer cache) is factored into analysis
- No application changes required



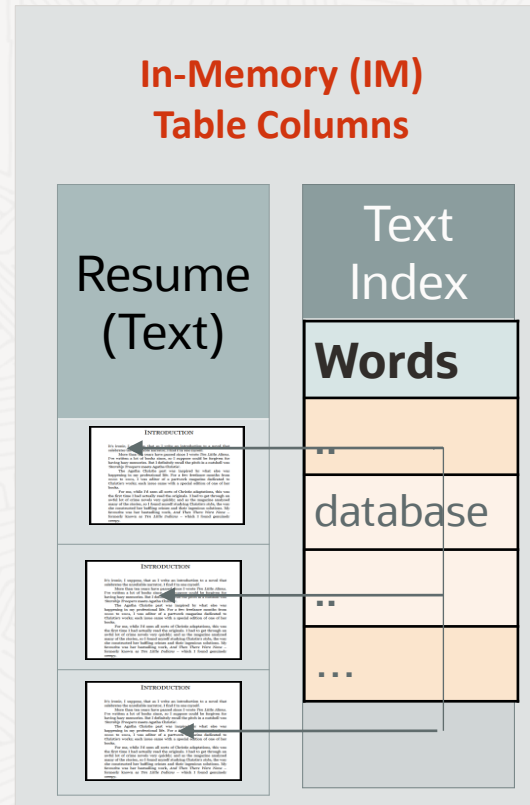
Converged Workloads

In-Memory Analytics on Spatial, Text, and JSON

1. Store **Spatial** Summaries in Column Store for Faster Filtering



2. Store Optimized **Text** Index structure in Column Store for fast searches



3. Store **JSON** in optimized binary representation in Column Store

```
{  
  "Theater": "AMC 15",  
  "Movie": "Rogue One",  
  "Time": "2017-01-09 18:45",  
  "Tickets": {  
    "Adults": 2  
  }  
}
```



00101001...



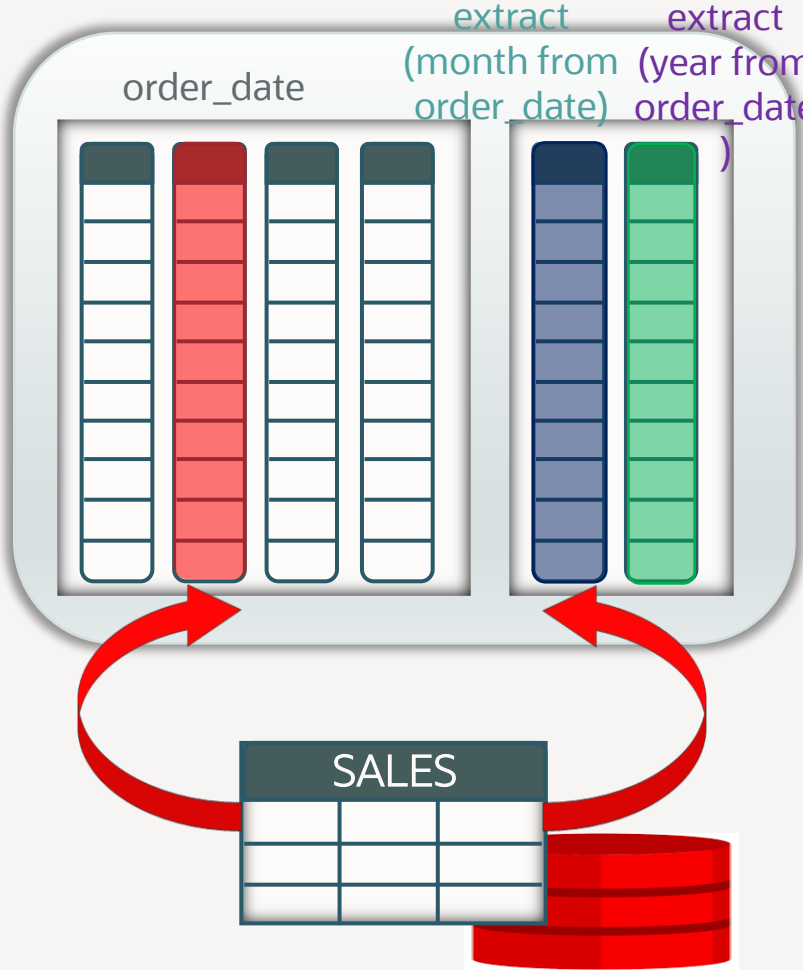
Accelerating DATE Queries

Consider a query to find the *Total sales amount for every month in 2022*

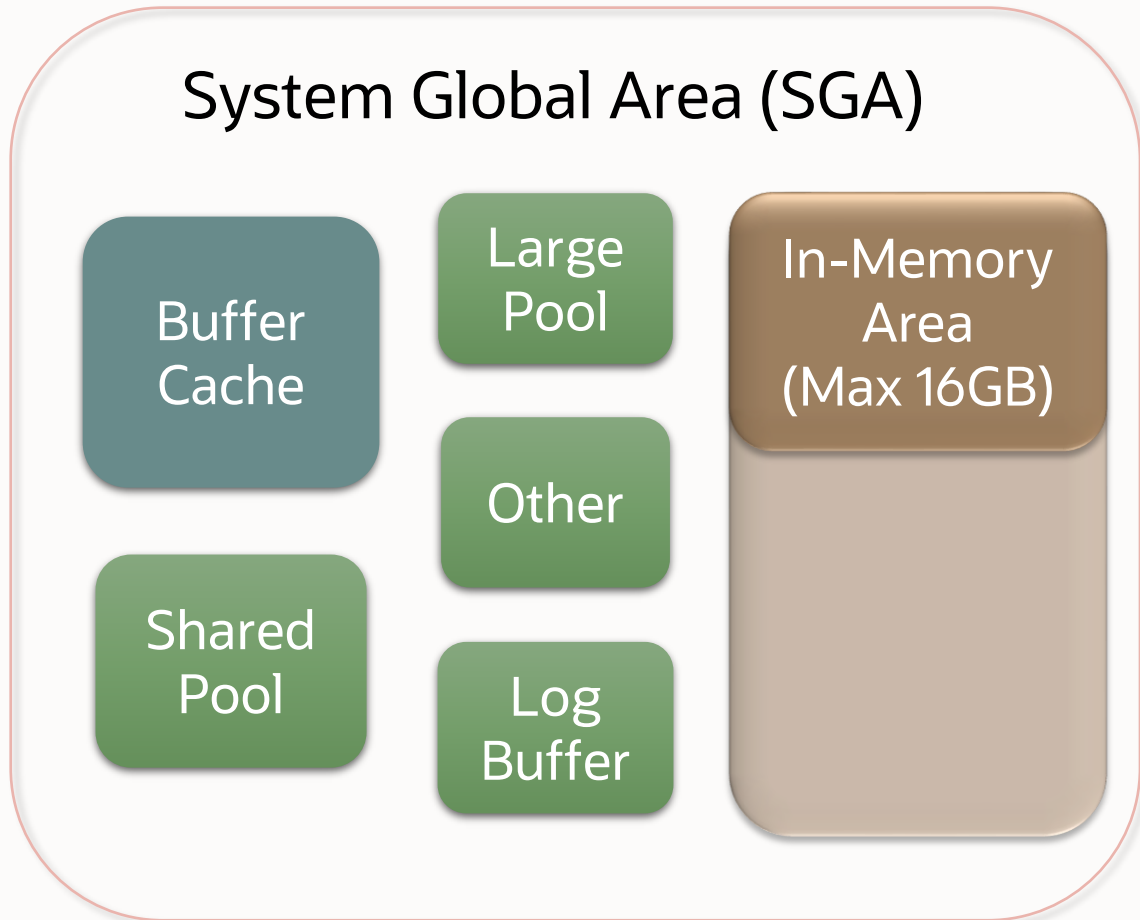
```
select extract(month from order_date)
MONTH, sum(order_amount) TOTAL_SALES
from SALES
where extract(year from order_date) = 2022
group by extract(month from order_date);
```

In-Memory can now run such queries by up-to **6X faster** by leveraging the In-Memory Expressions framework

- Each extracted component (e.g. MONTH) for a DATE column adds only a **1B** per-row in-memory overhead
- User can specify which DATE column component should be stored in-memory through a parameter



In-Memory Base Level Feature



- Introduced in 21c and backported to 19c (starting in 19.8)
- Customers can now use up to a IM column store without having to license the Database In-Memory option
- The purpose of the feature is to allow customers to see the value of Database In-Memory
- Not all Database In-Memory features are available with Base Level enabled
- Enabled with INMEMORY_FORCE parameter
 - Must be set to BASE_LEVEL



Exadata System Software 24ai





AI Smart Scan

Complete support for Oracle Database 23ai Ai Vector Search



Smart Scan

Full support for Oracle Database 23ai with a focus on increased performance and security



Performance

Faster transactions and automatic data loading into Flash Cache and XRMEM Data Accelerator



A

AI Smart Scan

Complete support for Oracle Database 23ai Ai Vector Search

ORACLE
Exadata 24^{ai}
System Software

C

Smart Scan

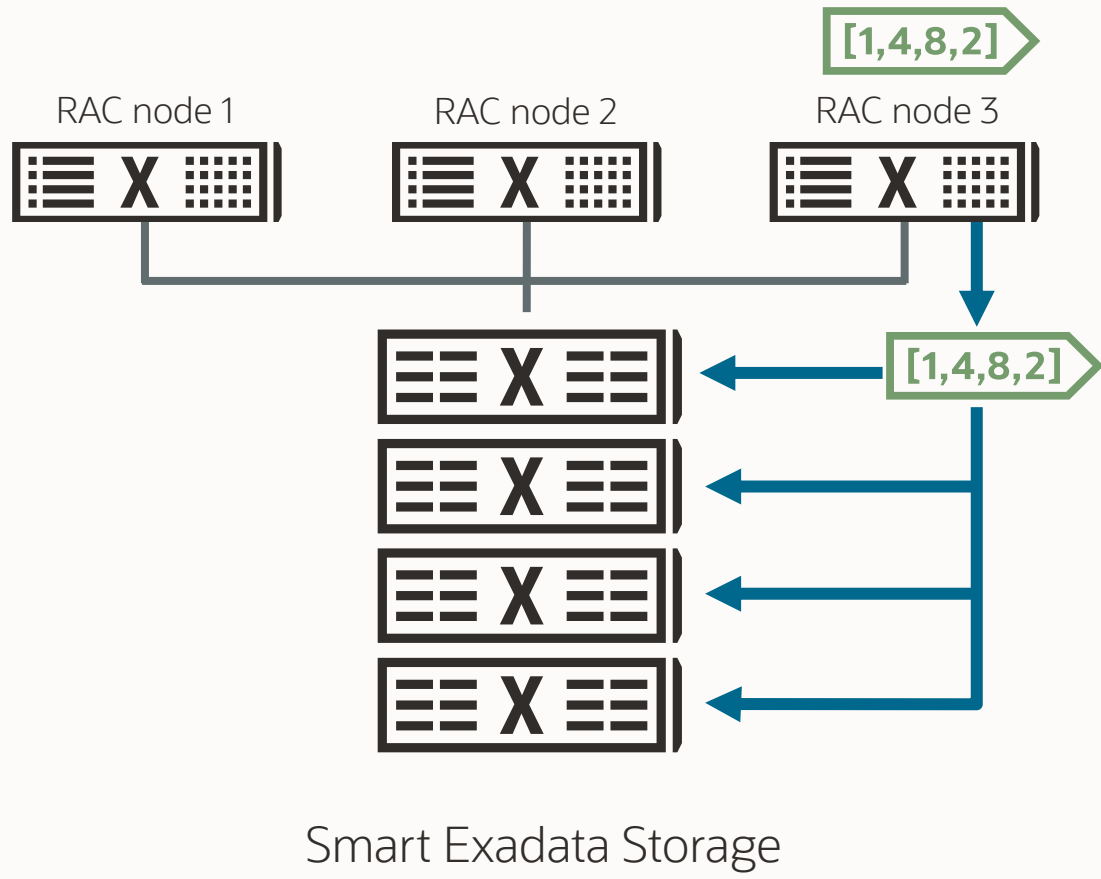
Full support for Oracle Database 23ai with a focus on increased performance and security

D

Performance

Faster transactions and automatic data loading into Flash Cache and XRMEM Data Accelerator

AI Smart Scan



Oracle AI Vector Search can be **transparently offloaded** to smart Exadata storage for faster search

Automatically parallelizes search across all storage servers

Each storage server independently computes top-K matches

- Database merges the results
- Runs up to orders of magnitude faster





AI Smart Scan

Complete support for Oracle Database 23ai Ai Vector Search



Smart Scan

Full support for Oracle Database 23ai with a focus on increased performance and security



Performance

Faster transactions and automatic data loading into Flash Cache and XRMEM Data Accelerator





Smart Scan

Columnar Scan at Memory Speed

JSON In-Memory

Transparent Cross-Tier Scan

Improvements in Smart Scan

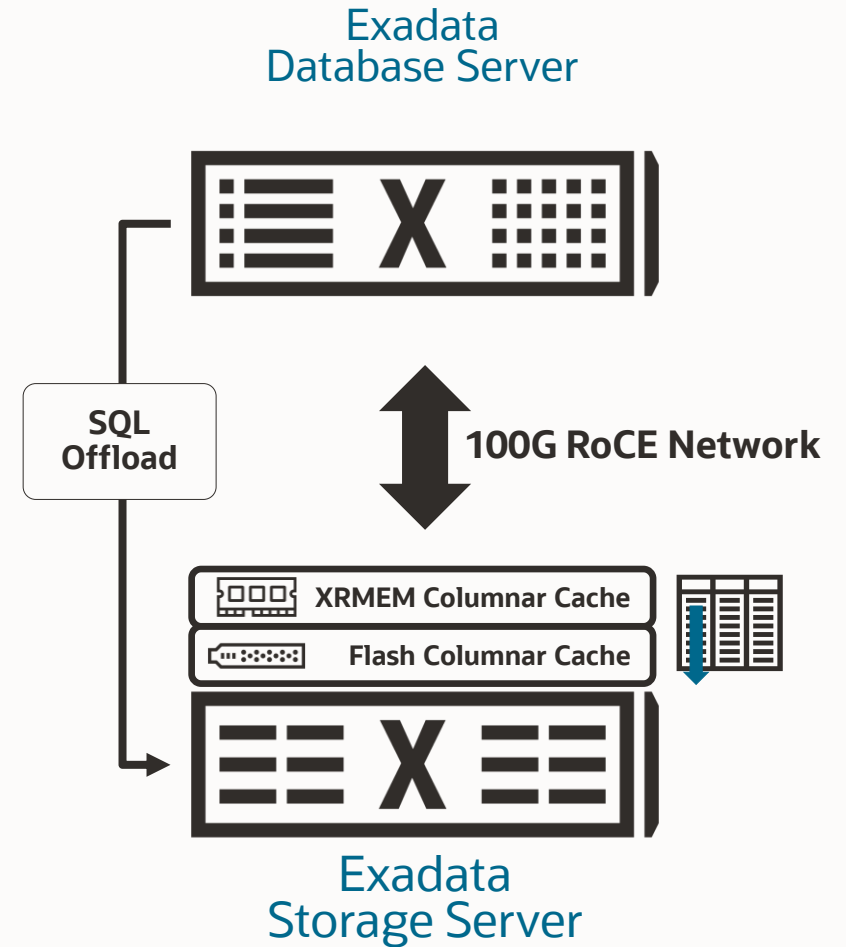
Columnar Smart Scan at Memory Speed

Smart Scans **directly** access columnar data from XRMEM memory on X10M storage servers

- Up to **6.5x** faster query performance
- OLTP-like **sub-second** latency for AI vector scans

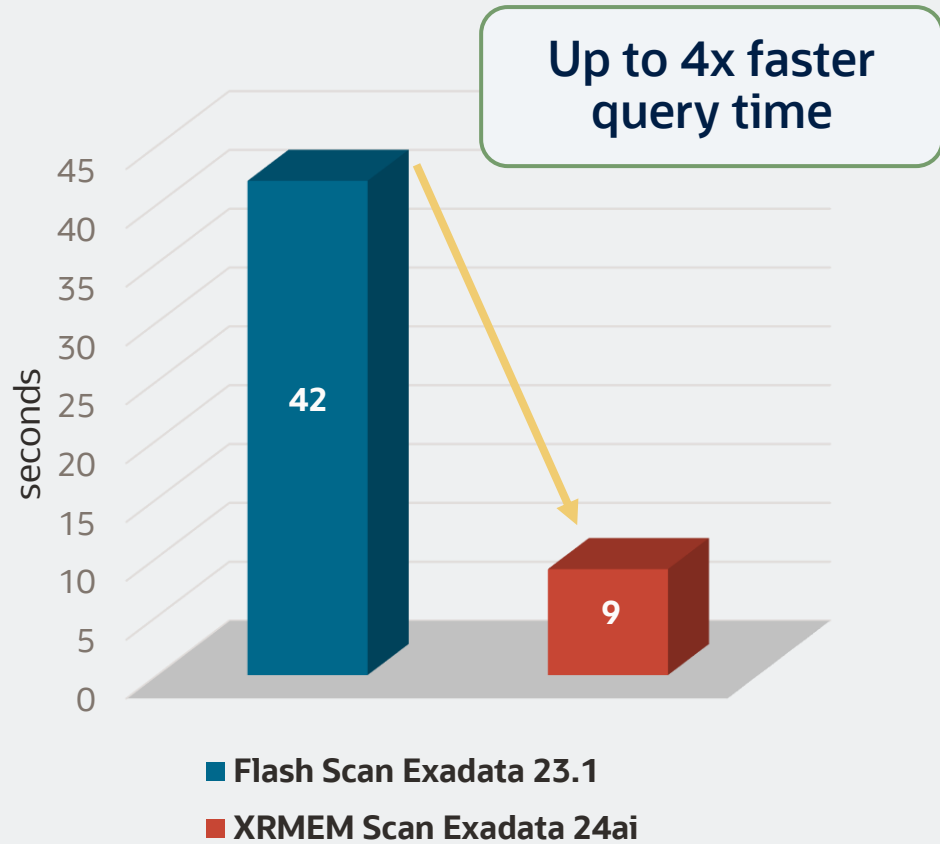
Cell software **automatically promotes** the most frequently accessed columnar data to XRMEM while keeping the rest of the columnar data in flash

Cell software **transparently places** OLTP data and columnar data in XRMEM to accelerate both workload types

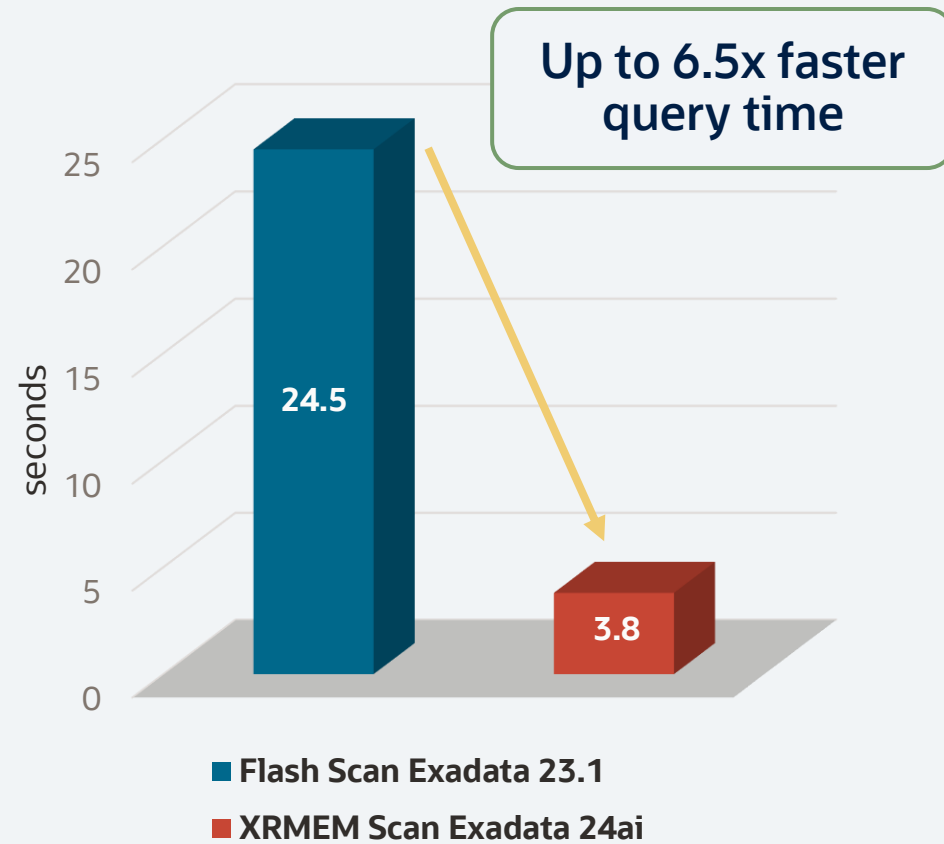


Columnar Smart Scan at Memory Speed

Analytic Scan Query on Exadata X10M



JSON Analytics on Nested Array



In-Memory Columnar Speed JSON Query

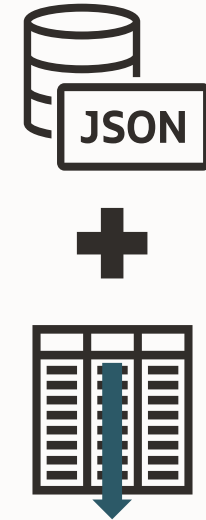
Smart Scan supports many JSON operators

- JSON_EXISTS, JSON_VALUE, JSON_QUERY, IS JSON, IS NOT JSON

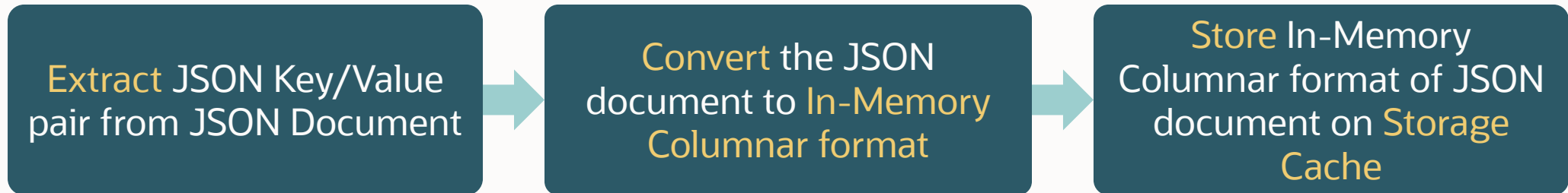
Exadata 24ai **automatically** and **transparently detects** JSON documents during Smart Scans and **stores** them in an In-Memory Columnar format on Exadata Flash Cache and XRMEM

- Oracle 23ai's JSON data type
- Oracle 19c's OSON format

Smart Scan of JSON document in In-Memory Columnar format benefits from SIMD Processing, Data Pruning, Dictionary encoding, Compression



Columnar scans on Flash and XRMEM



Up to 4.4x Higher Throughput

Up to 4.4x Lower Latency

Up to 7x Lower Storage CPU usage



```

{
  'name'      : 'John',
  'profession': 'Artist',
  'address'  :
  {
    'city'    : 'Redwood City',
    'state'   : 'CA'
  }
}
{
  'name'      : 'Alan',
  'profession': 'Carpenter',
  'address'  :
  {
    'city'    : 'New York',
    'state'   : 'NY'
  }
}
{
  'name'      : 'Clara',
  'profession': 'Teacher',
  'address'  :
  {
    'city'    : 'Dallas',
    'state'   : 'TX'
  }
}

```



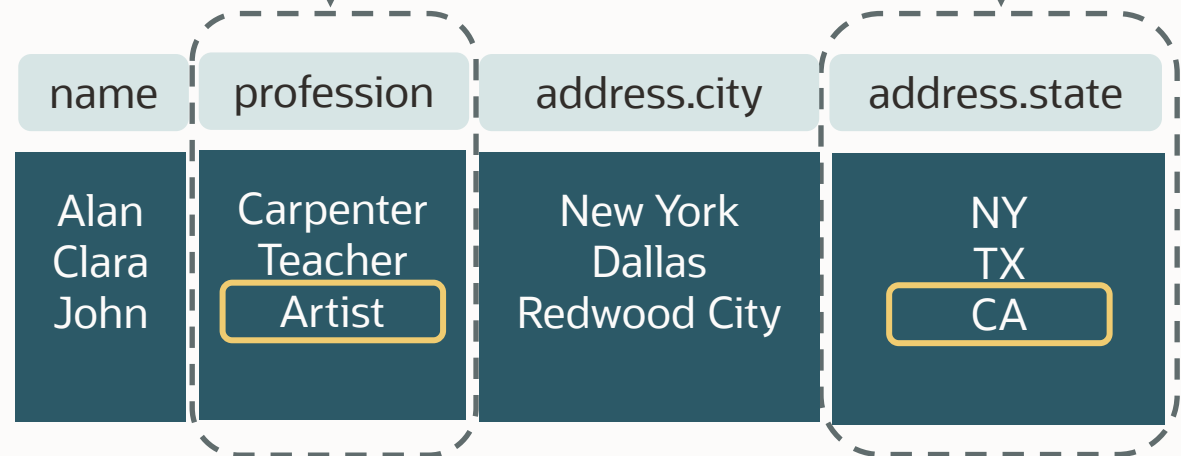
{JSON}

```

-- Find Artists in California
SELECT COUNT(*)
FROM   club_members
WHERE  JSON_EXISTS(jdoc,
                  '$.person?(@.profession = 'Artist' && @.address.state='CA')');

```

Columnar
Representation
In Flash Cache



Smart Scan Index Organized Table (IOT)

Index Organized Tables store data in a B-Tree Index structure

- Both **key and non-key** columns are **stored in leaf blocks**

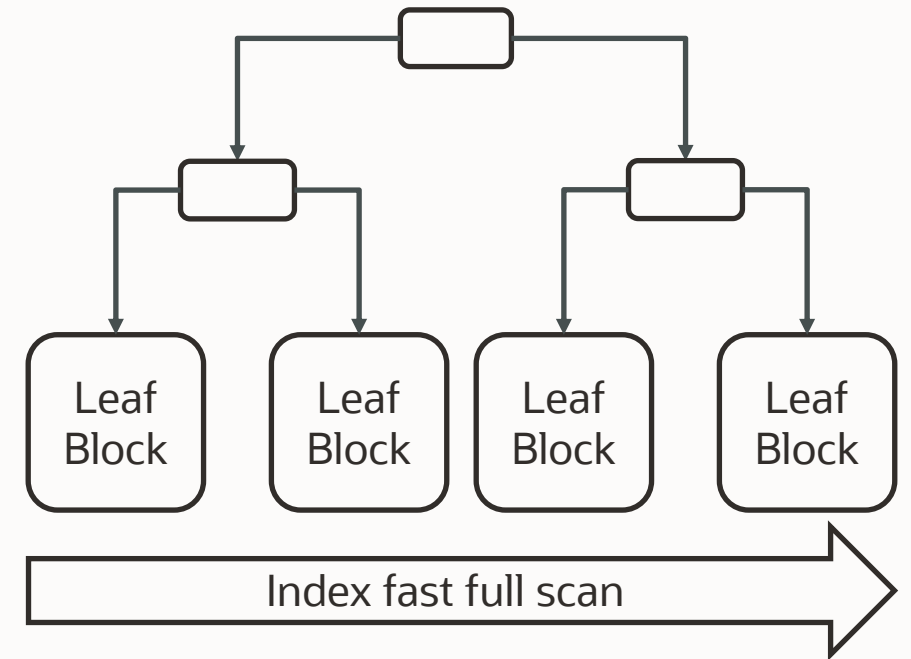
Smart Scan is supported with ALL compression formats used to store IOTs

Transparently works with all Smart Scan capabilities

- Storage Index, In-Memory Columnar Cache, etc.

Explain plan indicates Smart Scan with STORAGE operation

- INDEX STORAGE FAST FULL SCAN
- INDEX STORAGE FULL SCAN



Up to **13x** faster queries





Smart Scan Index Organized Table



```
SQL> explain plan for select count(*) from orders_iot where total_price > 1000;
```

Explained.

```
SQL> select * from dbms_xplan.display();
```

PLAN_TABLE_OUTPUT

Plan hash value: 2190295915

Id	Operation	Name	Rows	Bytes	Cost	(%CPU)	Time
0	SELECT STATEMENT		1	6	3900	0 (1)	00:00:02
1	SORT AGGREGATE		1	6			
* 2	INDEX STORAGE FAST FULL SCAN	SYS_IOT_TOP_27239	20M.	114M	3900	0 (1)	00:00:02

Predicate Information (identified by operation id):

```
2 - storage("TOTAL_PRICE">1000)
    filter("TOTOAL_PRICE">1000)
```

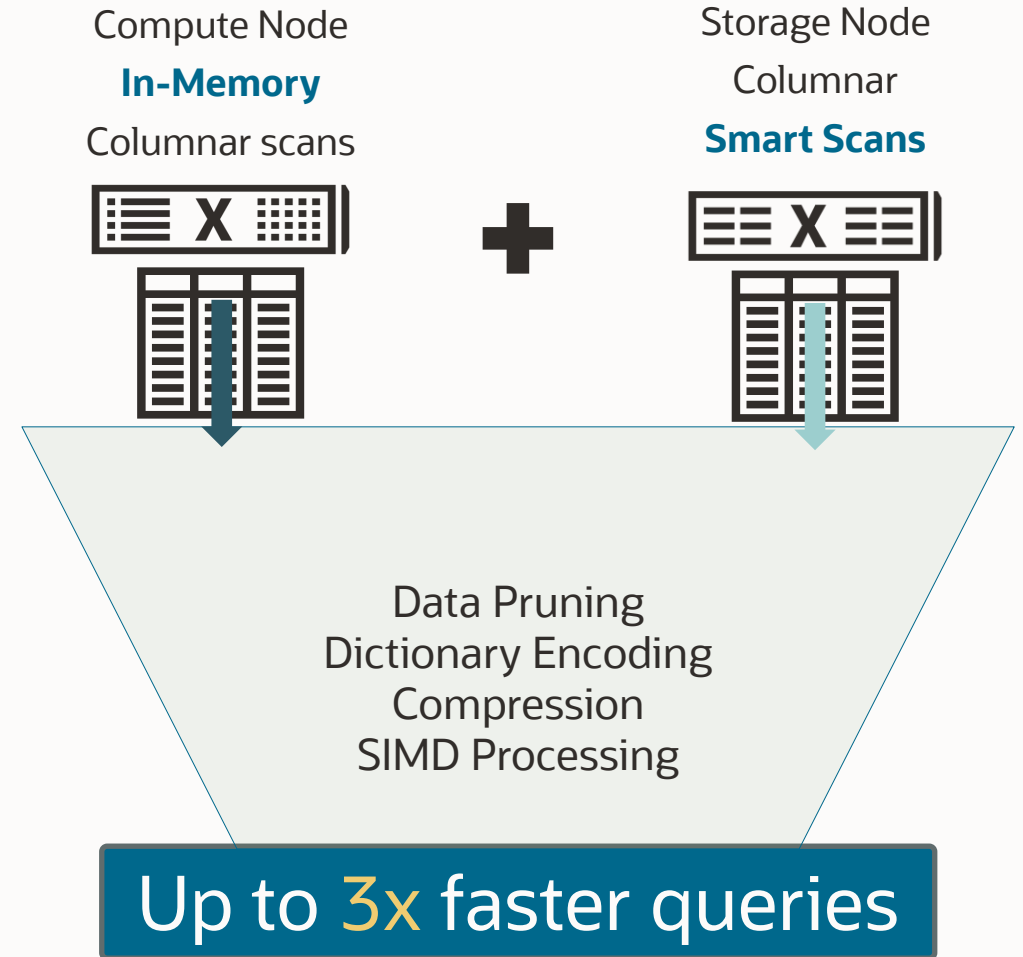
15 rows selected.

Transparent Cross-Tier Scan – In-Memory + Columnar Smart Scan

Analytic workload tables are large and data is spread across compute and storage nodes in Columnar format

Single query scans **In-Memory** Columnar data on **compute** and **storage**

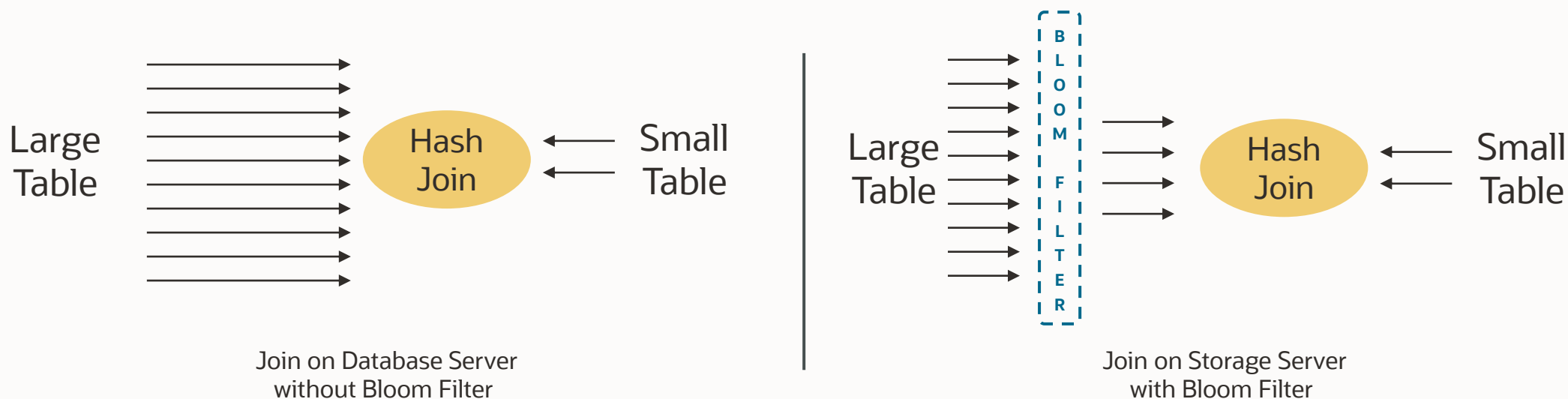
- **Serial** and **Parallel** queries can perform Transparent Cross-Tier Scan
- Both tiers benefit from **SIMD Processing, Compression, Dictionary Encoding and Data Pruning**



Faster Hash Joins on Large Tables

Database **automatically and transparently** summarizes smaller tables in hash join
Applying a **bloom filter on the large table reduces the amount of data, making joins faster**

- Bloom filter is distributed to all storage servers and is applied on the large table in parallel
- **Database 23ai can offload larger bloom filters** reducing data from large tables in the join



Up to **2X faster** analytic query performance



Smart Scan – Additional Features

Wide Tables

Smart Scan transparently supports Wide Tables

Database 23ai supports 4096 columns per table

COMPATIBLE=23.0.0

MAX_COLUMNS=STANDARD (default) or EXTENDED (4096)



Time Zone Upgrade

Smart Scan is enabled for queries on tables with no `TIMESTAMP WITH TIMEZONE` columns during time zone upgrade

Time zone upgrades needed to correctly process time calculations and daylight-saving time transitions

- Potentially alters data stored in `TIMESTAMP WITH TIMEZONE` columns



Smart Scan operators and functions

Date

CEIL and FLOOR functions

Interval

CEIL, FLOOR, ROUND and TRUNC functions

Boolean

TO_BOOLEAN function
IS FALSE, IS NOT FALSE, IS TRUE, IS NOT TRUE predicate operators



AI Smart Scan

Complete support for Oracle Database 23ai Ai Vector Search

ORACLE
Exadata **24^{ai}**
System Software



Smart Scan

Full support for Oracle Database 23ai with a focus on increased performance and security



Performance

Faster transactions and automatic data loading into Flash Cache and XRMEM Data Accelerator



Performance

Pipelined Log Writes

Automatic KEEP Object Load into
Exadata Flash Cache

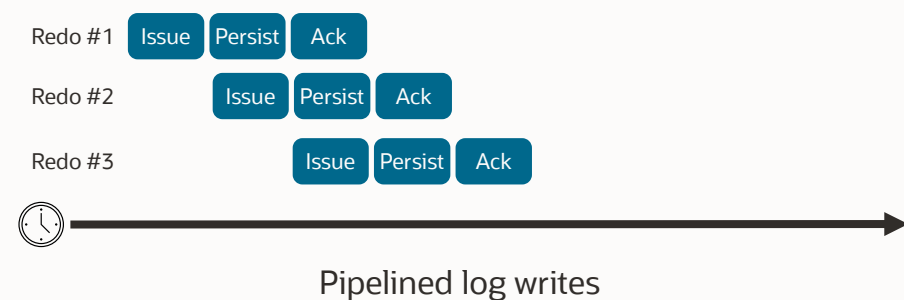
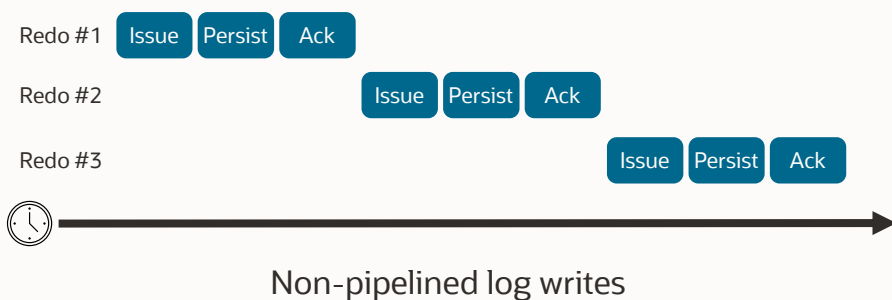
Pipelined Log Writes

Low redo log write latency is vital for OLTP database performance

- Transaction processing waits for the commit redo to persist
- Inter-RAC instance block transfer is allowed after dependent redo has persisted

Oracle Database 23ai and 19c, database log writers [pipeline](#) log writes on Exadata X10M

- Leverages the ultra-fast RoCE network and Smart Flash Log on Exadata storage



Up to **1.45x** faster transaction processing

Automatic KEEP Object Load into Exadata Flash Cache

ALTER TABLE with the **CELL_FLASH_CACHE KEEP** clause

- **Automatically** and **asynchronously** loads table data and pins them in the Flash Cache
- Avoids need to execute an explicit full table scan
- Reduce checkpoint writes from the buffer cache
- Improve network efficiency by not returning query data to the DB nodes



```
SQL> ALTER TABLE LINEITEM STORAGE (CELL_FLASH_CACHE KEEP);
```

```
Table altered.
```

```
Elapsed: 00:00:08.59
```

```
CellCLI> LIST FLASHCACHECONTENT WHERE OBJECTNUMBER=10000 DETAIL
```

```
cachedKeepSize:          487711162368
```

```
cachedSize:              487711162368
```



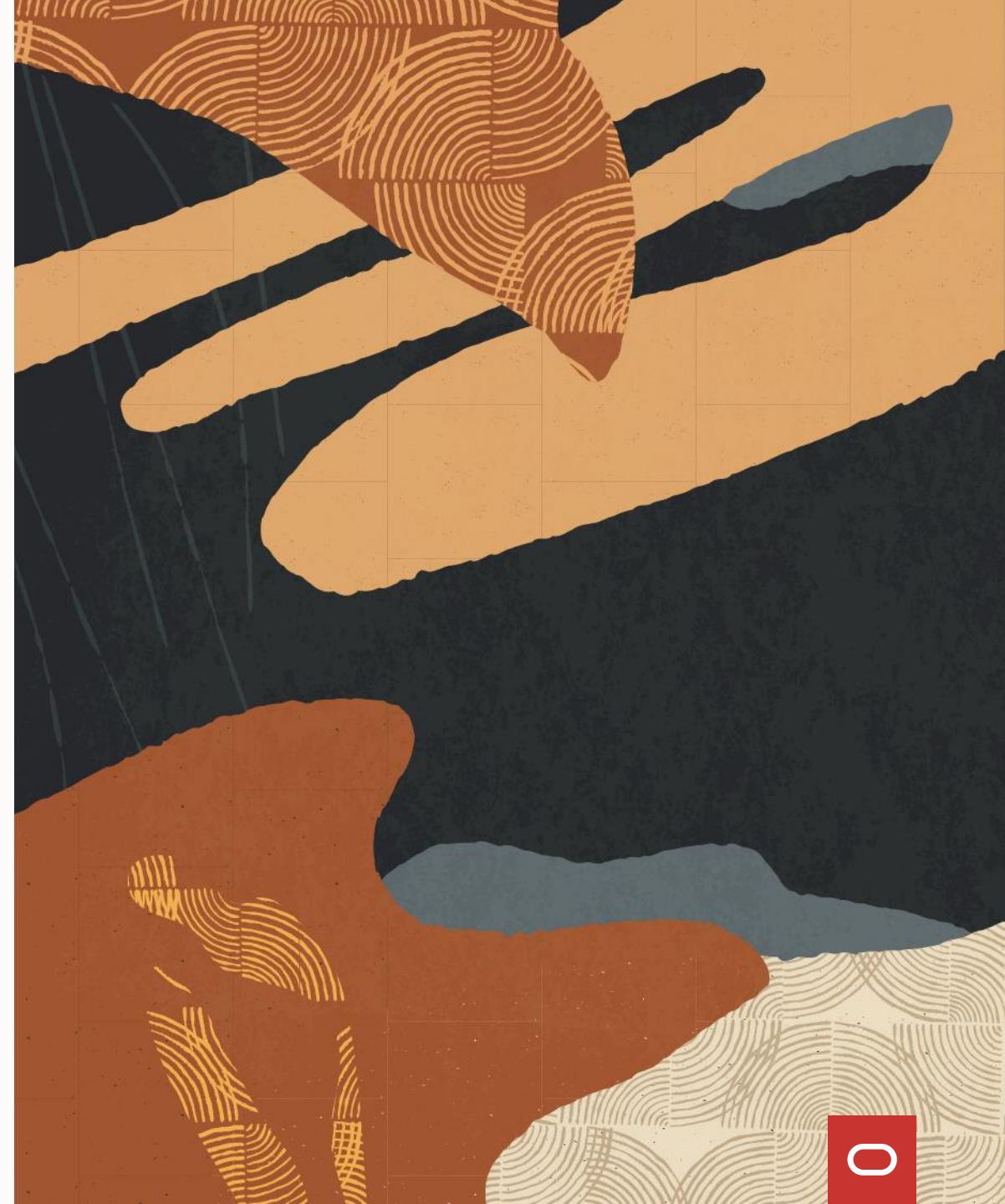
Database 23ai and Exadata 24ai are faster than ever for AI and all other kinds of applications on any data

Key Takeaways

Thank you!



Q&A



Our mission is to help people see
data in new ways, discover insights,
unlock endless possibilities.

