ORACLE®

Database In-Memory

The Next Generation

(CMU PDL Talk: May 3rd, 2016)

Shasank Chavan
Architect & Director
In-Memory Data Technologies Group

Tirthankar Lahiri
Vice President
Data Technologies Group

Mike Gleeson
Consulting Member of Technical Staff
Database File System (DBFS) & Secure Files Group

A Brief History of Time

Before 2013

Miscellaneous Events

- Big Bang
- Moon Landing
- Battlestar Galactica
- Relational Databases
- etc.

Row Format Databases vs. Column Format 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

State of the Art: Choose One Format, Enjoy Benefits, Live with Drawbacks

A Brief History of Time

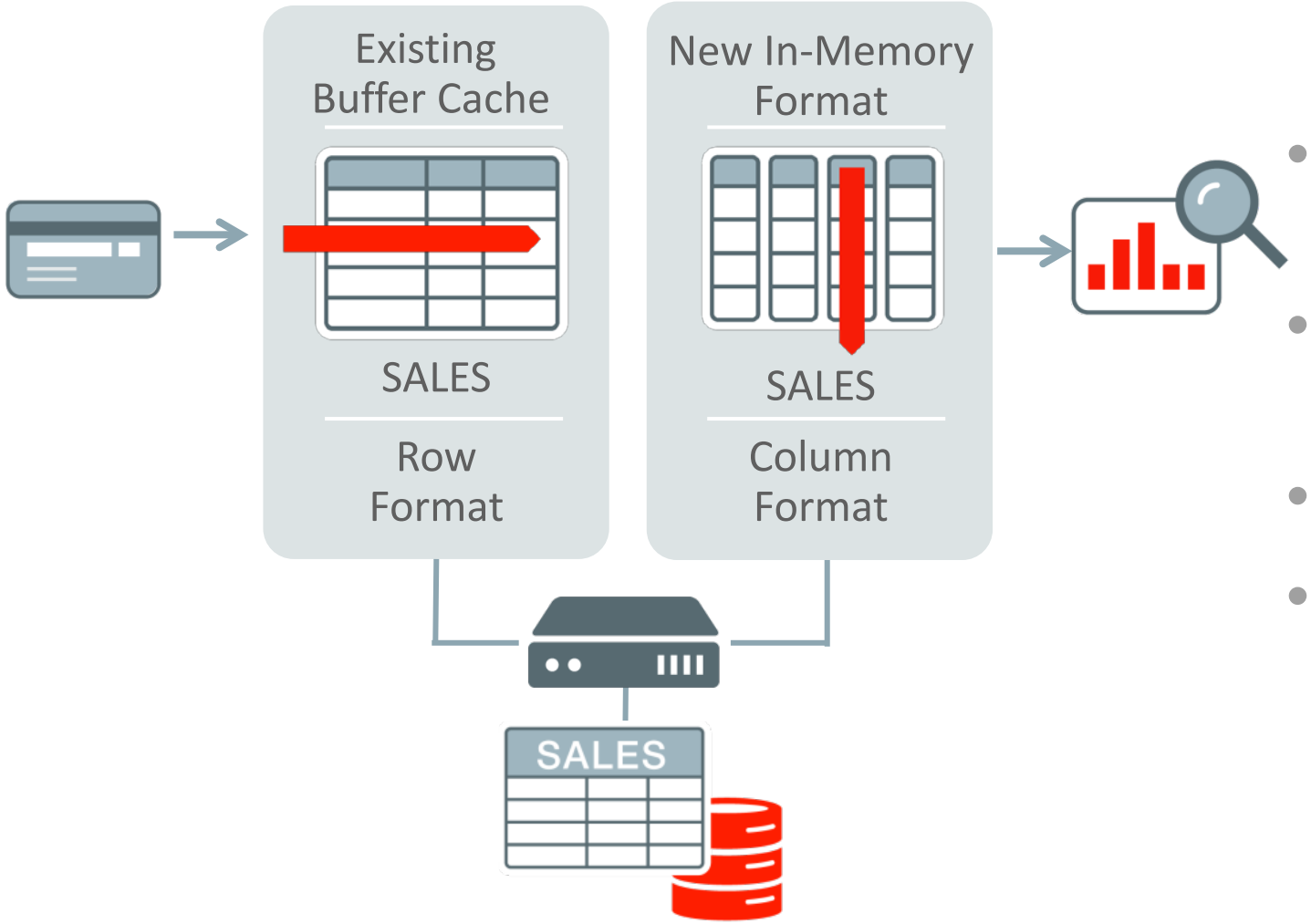
Miscellaneous Events

- Big Bang
- Moon Landing
- Battlestar Galactica
- Relational Databases
- etc.

Database In-Memory Announced

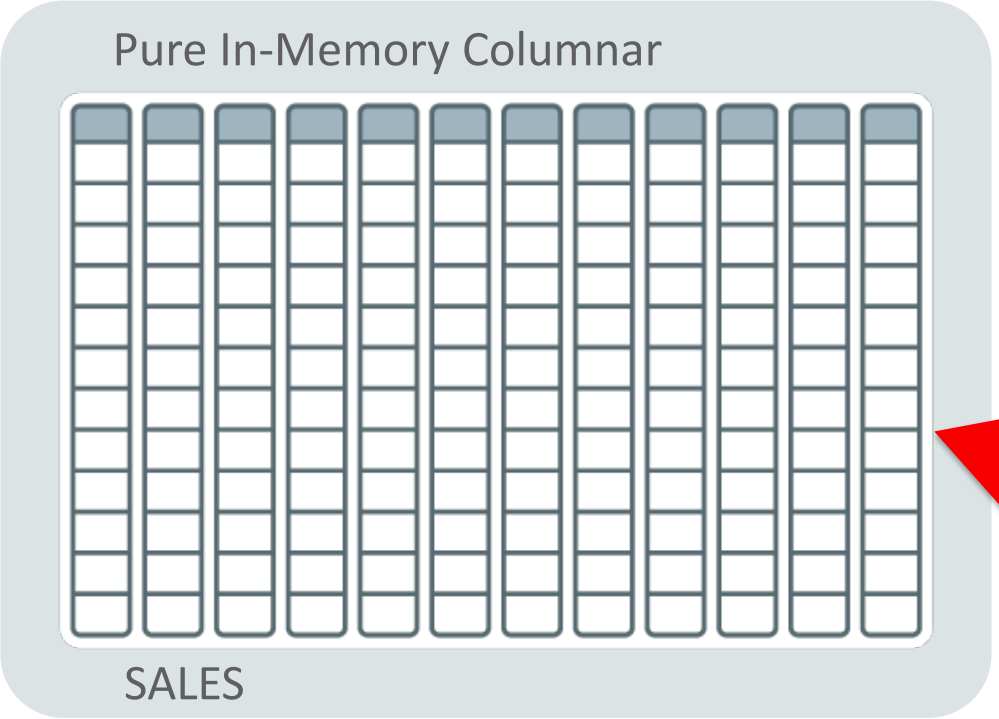
- Industry first dual-format inmemory database
- Transparent to Applications
- Trivial to deploy

Oracle Database In-Memory: Dual Format Architecture



- **BOTH** row and column formats for same table
- Simultaneously active and consistent
- OLTP uses existing row format
- Analytics uses new In-Memory Column format

In-Memory Columnar Format



- Pure in-memory column format
  - Enable for subset of database
  - Cheap to maintain – no logging or IO
  - Allows efficient OLTP
  - No change to disk format
- Built **seamlessly** into Oracle Database
  - Appears as a new storage type
  - **Transparent** to Applications
  - All Enterprise Features work ..
    - Availability – RAC, Flashback, DataGuard, etc.
    - Security – Encryption, Auditing, etc.

A Brief History of Time

Before 2013

Miscellaneous Events

- Big Bang
- Moon Landing
- Battlestar Galactica
- Relational Databases
- etc.

Oct 2013

Database In-Memory Announced

- Industry first dual-format inmemory database
- Transparent to Applications
- Trivial to deploy

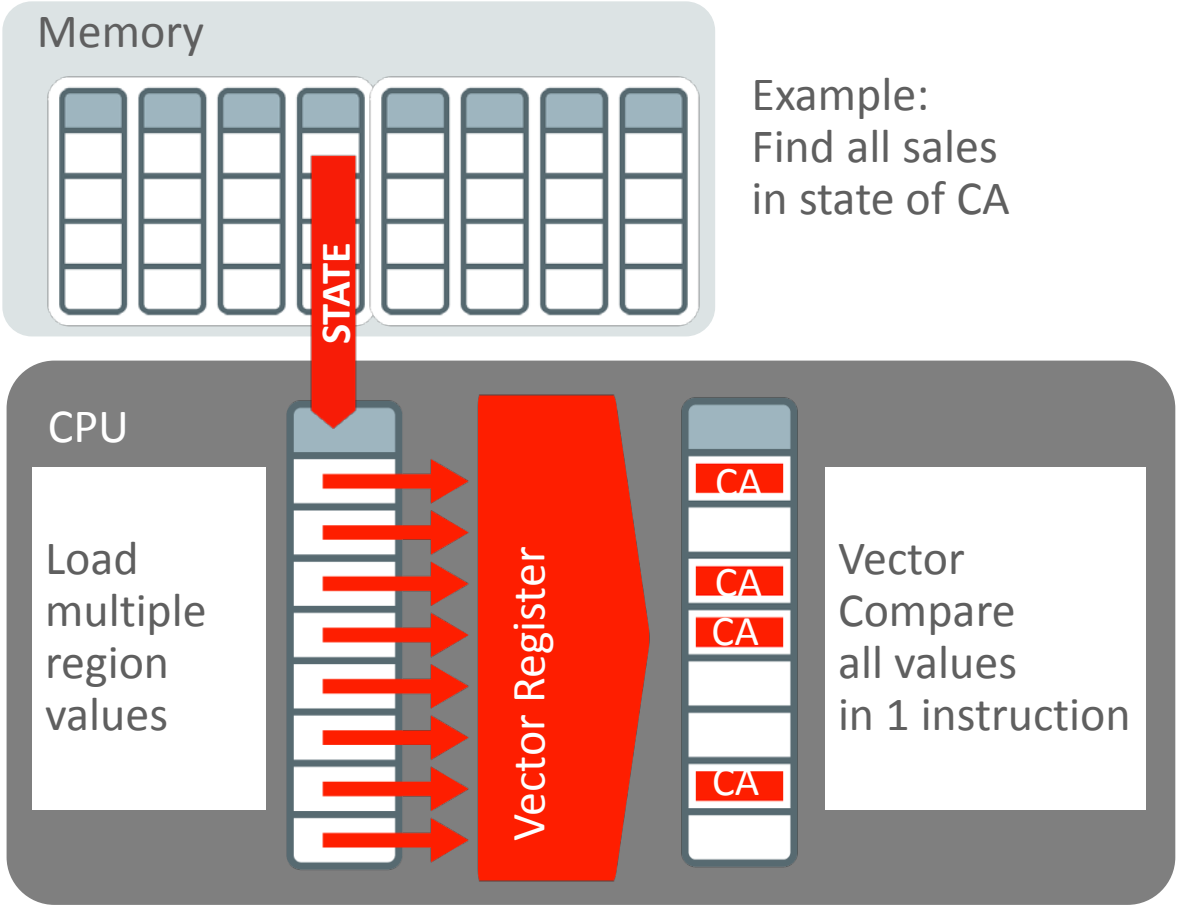
July 2014

Database In-Memory General Availability

- Oracle Database 12.1.0.2
- Orders of magnitude faster analytic workloads
- Faster Enterprise mixed-workload OLTP

ORACLE

In-Memory Scans



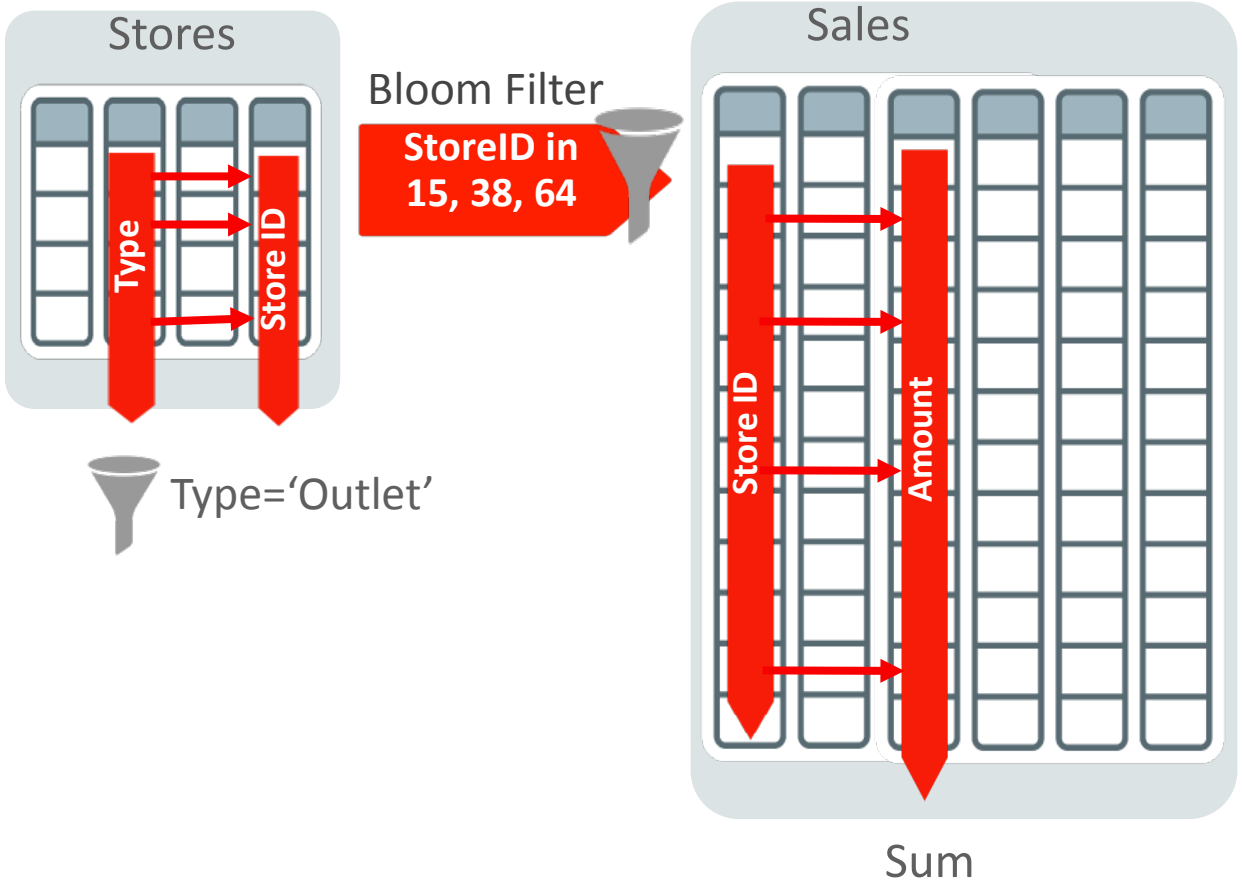
Example:
Find all sales
in state of CA

> 100x Faster

- Each CPU core scans only required columns
- SIMD vector instructions used to process multiple values in each instruction
  - E.g. Intel AVX instructions with 256 bit vector registers
- **Billions of rows/sec** scan rate per CPU core
  - Row format is millions/sec

In-Memory Joins

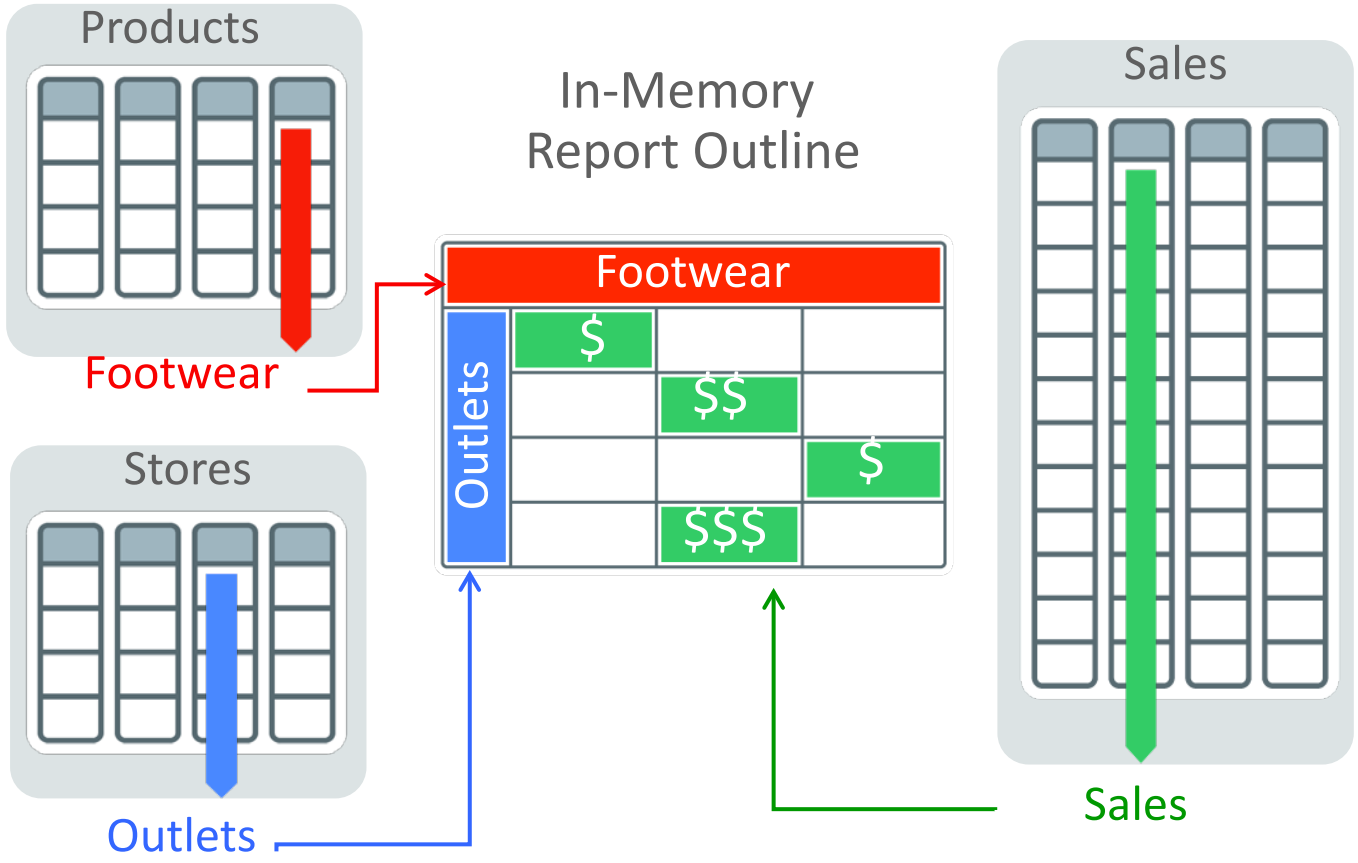
Example: Find total sales in outlet stores



- Bloom filter created on dimension scan
- Bloom filter pushdown:
  - Filtering pushed down to fact scan
  - Returns only rows that are likely to be join candidates
- Joins tables **10x** faster

In-Memory Aggregation

Example: Report sales of footwear in outlet stores



- Create (empty) in-memory report outline during dimension scan
- Push down report outline aggregation to fact scan
- Reduces complex aggregations to series of fast inmemory scans
- Reports run **10x** faster
  - Without predefined cubes

Scale-Out In-Memory Database to Any Size

- Scale-Out across servers to grow memory and CPUs
  - DISTRIBUTE clause: by Partition, Sub-Partition, or Rowid Range
- In-Memory **queries parallelized** across servers to access local column data
- **Direct-to-wire** InfiniBand protocol speeds messaging

A Brief History of Time

Before 2013

Miscellaneous Events

- Big Bang
- Moon Landing
- Battlestar Galactica
- Relational Databases
- etc.

Oct 2013

Database In-Memory Announced

- Industry first dual-format inmemory database
- Transparent to Applications
- Trivial to deploy

July 2014

Database In-Memory General Availability

- Oracle Database 12.1.0.2
- Orders of magnitude faster analytic workloads
- Faster Enterprise mixed-workload OLTP

Oct 2015

Database In-Memory Adoption

- Fastest growing option in the history of Oracle Database
- Further enhancements across 13 interim bundle patch releases

ORACLE

Mixed Workloads: Performance Improvement Example*

Metric	12.1.0.2 (GA release)	12.1.0.2 (BP13)	Improvement
CPU Utilization	98%	29%	3.37x
Overall Throughput	1400 ops / sec	4000 ops / sec	2.85x

Synthetic mixed workload:

- Analytic scans (1%)
- OLTP queries (30%)
- Update (30%)
- Insert (24%)
- Delete (15%)

(*) Disclaimer: Results for illustrative purpose only. Your Mileage May Vary (workload, queries, data)

ORACLE

A Brief History of Time

Before 2013

Miscellaneous Events

- Big Bang
- Moon Landing
- Battlestar Galactica
- Relational Databases
- etc.

Oct 2013

Database In-Memory Announced

- Industry first dual-format inmemory database
- Transparent to Applications
- Trival to deploy

July 2014

Database In-Memory General Availability

- Oracle Database 12.1.0.2
- Orders of magnitude faster analytic workloads
- Faster Enterprise mixed-workload OLTP

Oct 2015

Database In-Memory Adoption

- Fastest growing option in the history of Oracle Database
- Further performance enhancements for analytics and mixed workloads across interim releases

2016

Database In-Memory Release 2

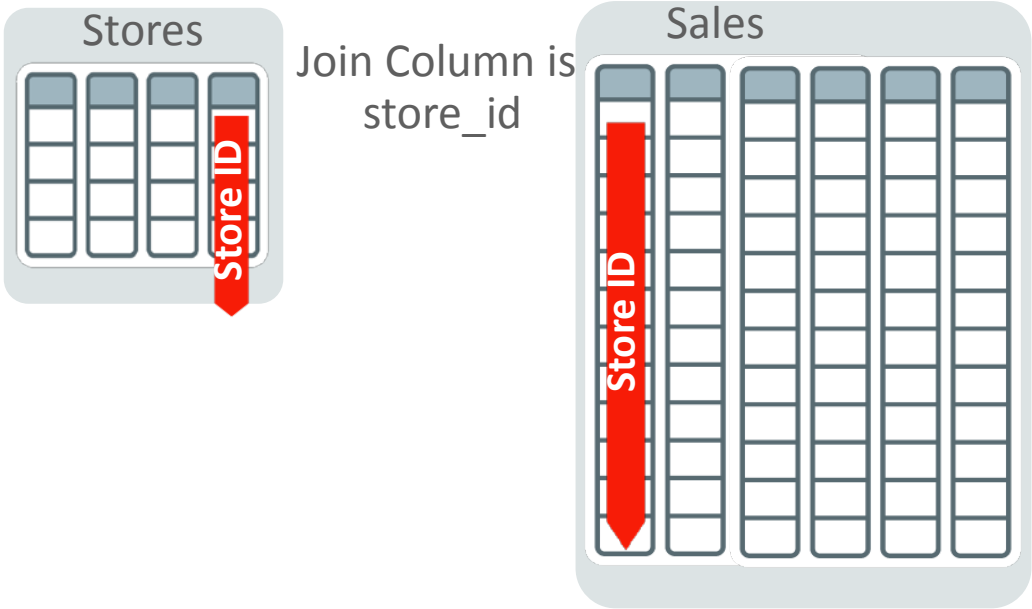
Back to the Future

Database In-Memory Release 2

ORACLE

Faster In-Memory Joins: In-Memory Join Groups

Example:
Find total sales in outlet stores



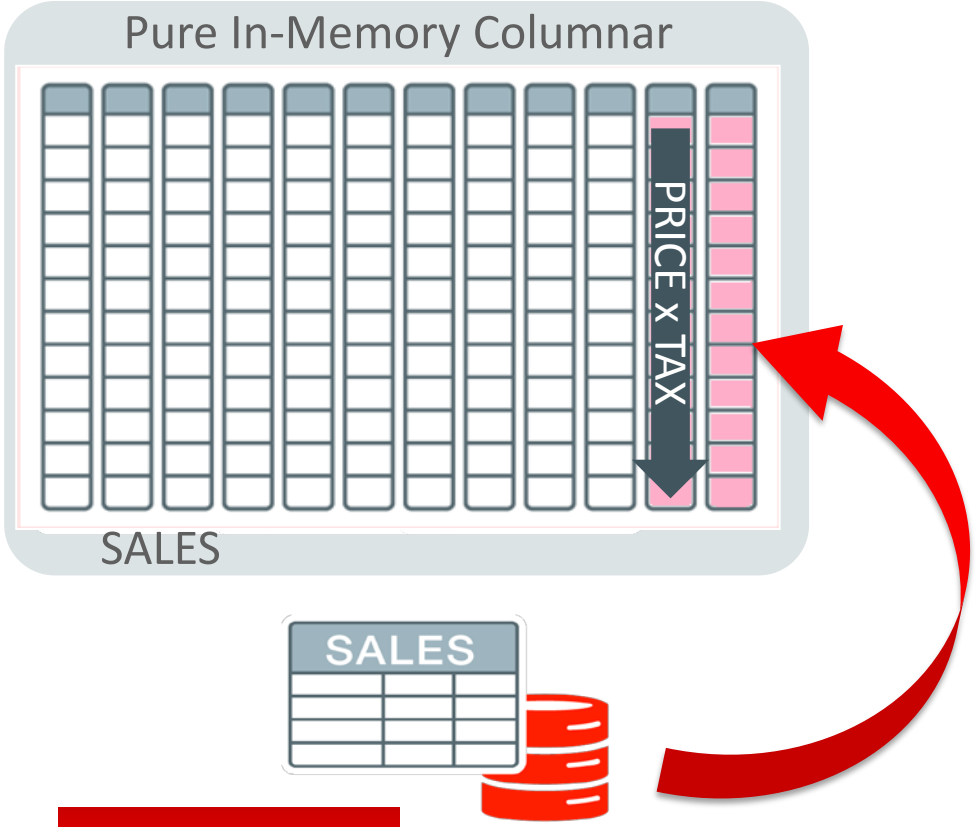

```
Create Join Group store_sales_jg
(STORES, STORE_ID), (SALES, STORE_ID);
```

- **Today:** Joins run on decompressed data
  - Data populated in-memory is compressed
  - All qualifying column data must be decompressed before join
- **12.2:** Joins run on compressed data
  - Users specify which columns used for joins across tables as a **Join Group**
  - Columns specified in a group use same dictionary for encoding
  - Joins occur on symbols rather than data
  - Greatly improves the efficiency of joins

Faster Expression Evaluation: In-Memory Expressions

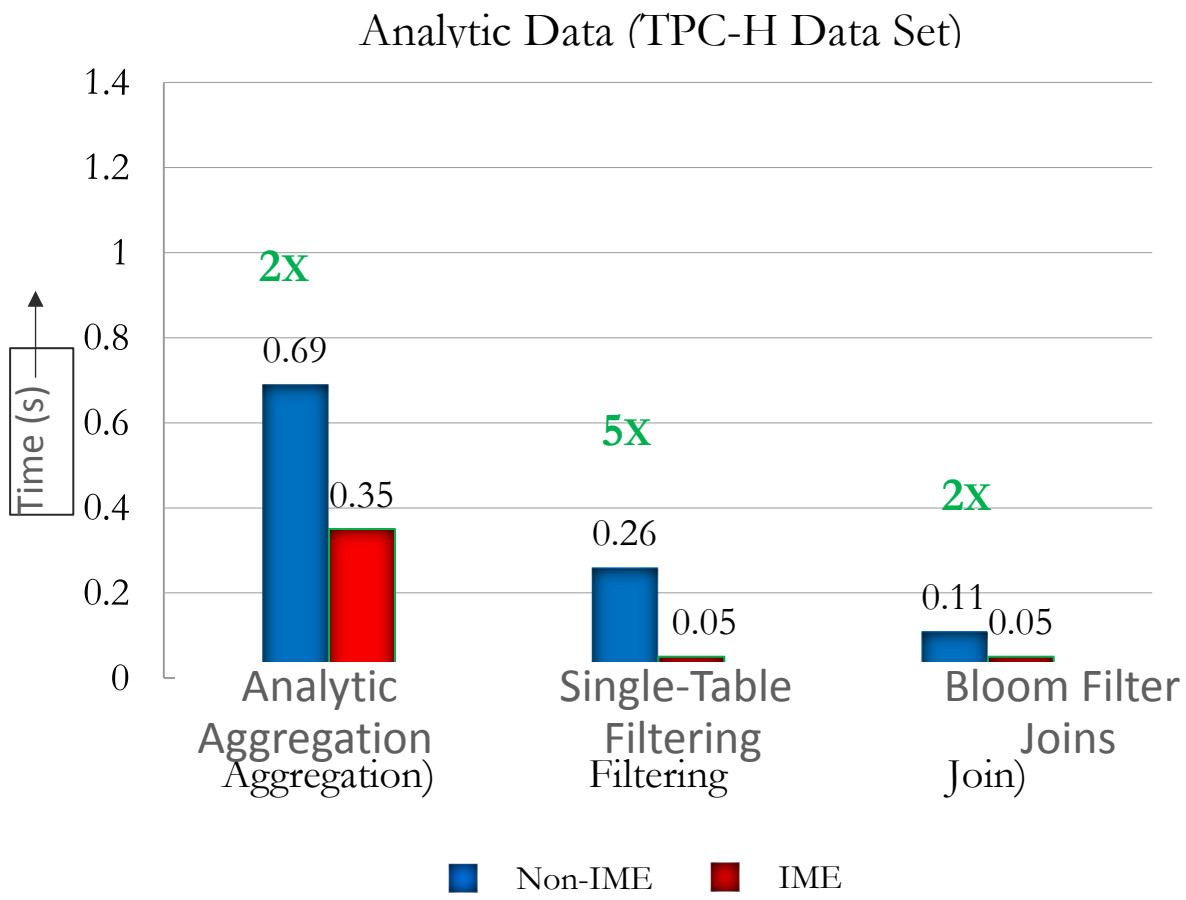
Example:

```
Select PRICE * TAX
From   SALES;
```



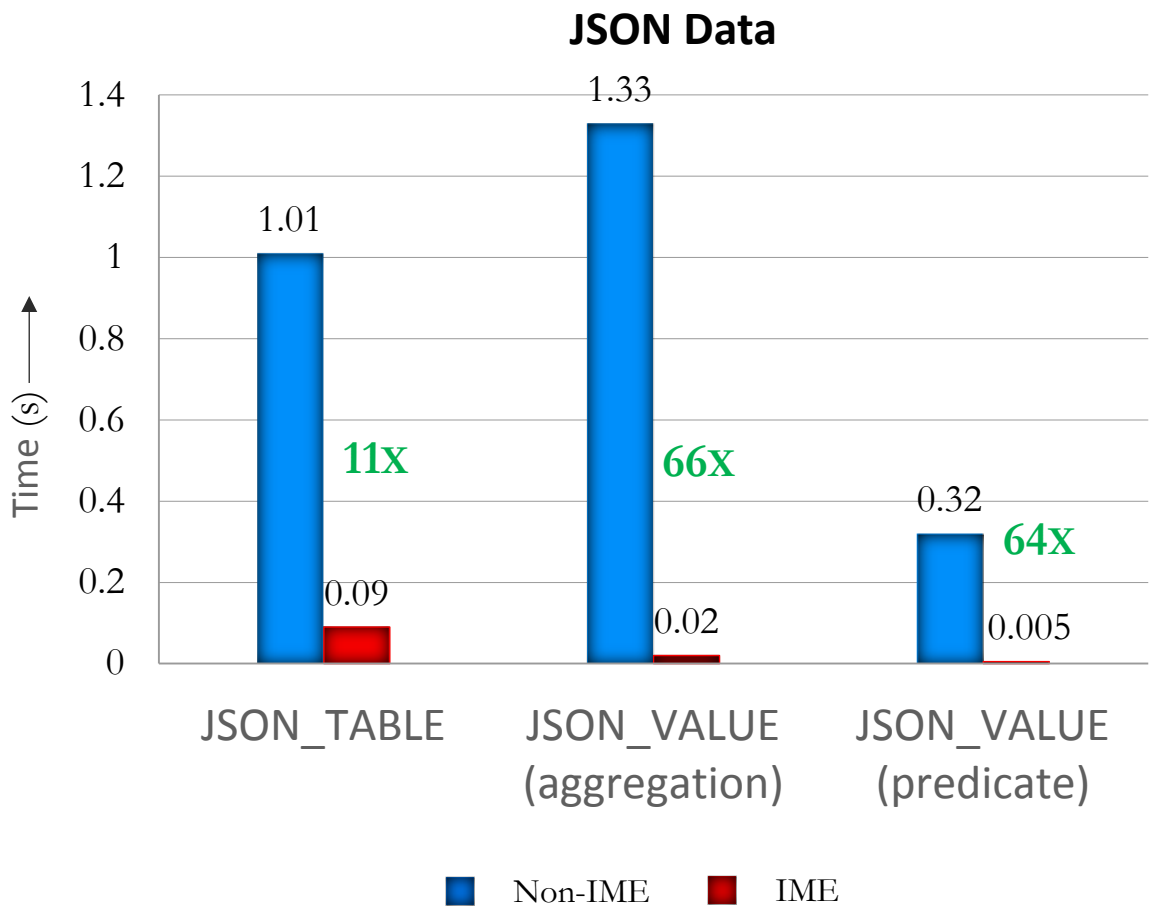
- **Today:** Commonly used expressions are recomputed every time
- **12.2:** In-Memory Expressions
  - Two modes:
    - Manual: User defined virtual columns
    - Automatic: Frequently-evaluated expressions
  - Many uses – e.g. Arithmetic Expressions, Type Conversions, In-Memory optimized JSON format
  - Maintained consistently with source columns
  - All In-Memory query performance optimizations apply - Vector Processing, min-max pruning etc.

In-Memory Expressions: Performance Gain Example



- Analytic Queries on star schema
- Explicitly declared in-memory expressions
  - price * (1 – discount)
  - price * (1 – discount) * (1 + tax)
- Shows major performance gains for analytic queries

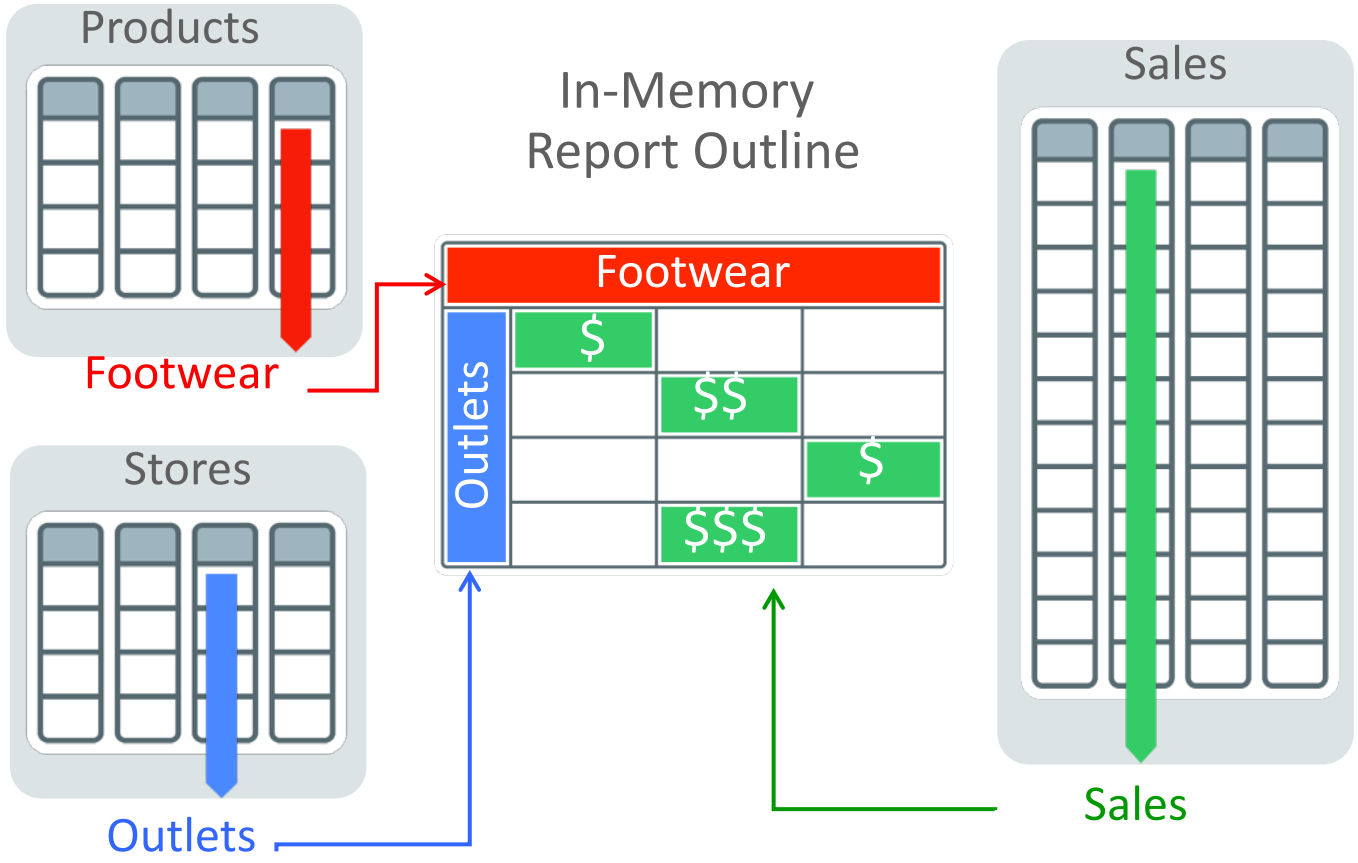
In-Memory Expressions: Performance Gain Example



- JSON automatically stored in memory optimized format
  - Allows much faster processing of JSON_TABLE operations
- Explicitly created IME on JSON_VALUE provides massive speedup for extracting scalar values from JSON fields

Faster Aggregation: Aggregation Push-Down

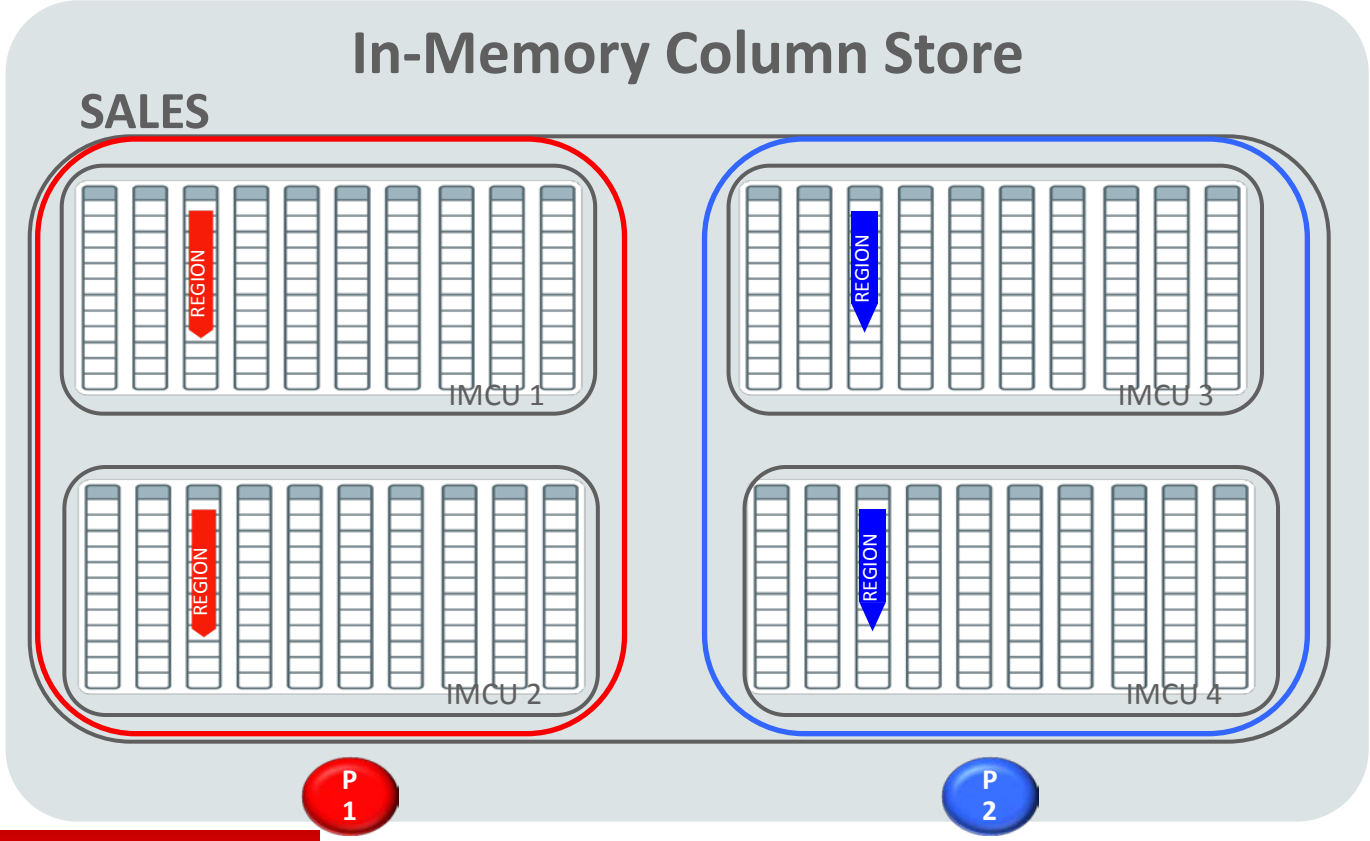
Example: Report sales of footwear in outlet stores



- Aggregation on compressed global dictionary codes.
  - Algorithm changes for late materialization.
- Vectorized aggregation using SIMD instructions.
- More aggressive benefit estimations
  - VGBY kicks in more often.

Faster Scans: In-Memory Dynamic Scans

```
SELECT count(*) FROM SALES
WHERE region='CA';
```

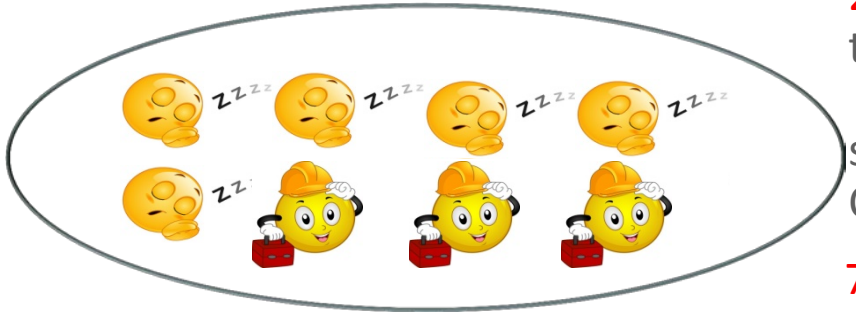


Dynamic Scan & Parallel Execution:

- Each Parallel server process gets multiple threads at a time
- Each threads scan 1 IMCU at a time
- Number of threads active controlled by Resource Manager

Dynamic Scans: How do they work?

1. Query is executed against a table In-Memory



2. A pool of light-weight threads is spawned
All threads are initially sleeping – consuming no CPU

7. The pool of light-weight threads is killed off after they are idle of a given period

4. Resource Manager determines how many threads can participate

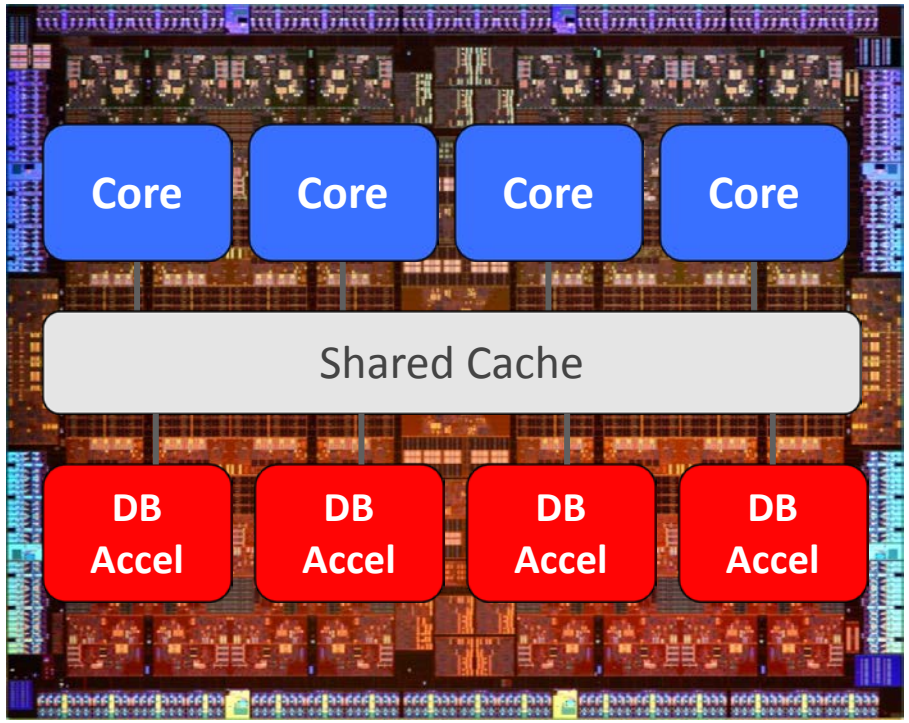
3. A separate task is created & queued for each IMCU that needs to be scanned

5. Results of each task are queued to be consumed by foreground process

6. If system is too busy & Resource Manager prevents threads from running, foreground process will execute remaining task(s)

SQL in Silicon: **10x** Acceleration of Database In-Memory

SPARC M7

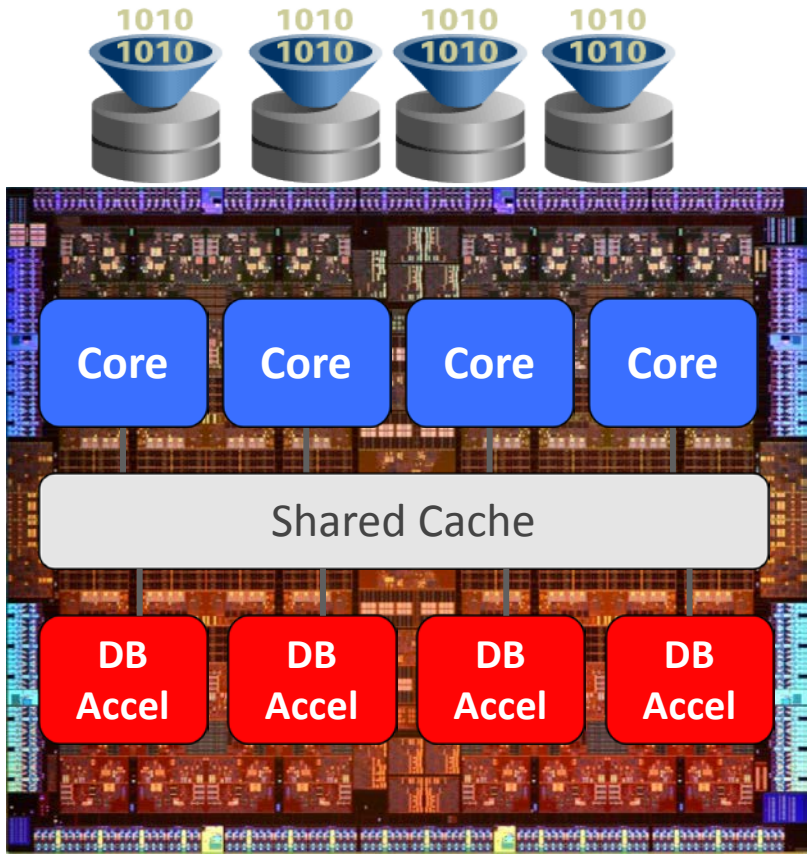


32 Database Accelerators (DAX)

- **Today:** We use standard SIMD vector instructions designed for Graphics and HPC, not for Databases
  - Translating database query operations to SIMD vector instructions is complex and expensive
- **SPARC M7:** New M7 chip has 32 Database Acceleration Engines (DAX), like having 32 specialized cores for Database In-Memory
  - Directly runs basic database query primitives
    - E.g. find all values that match 'California'
  - **2-10x** speedup – up to 170 Billion Rows per second

ORACLE

Compression in Silicon: **Double** In-Memory Capacity



32 OZIP Decompressors

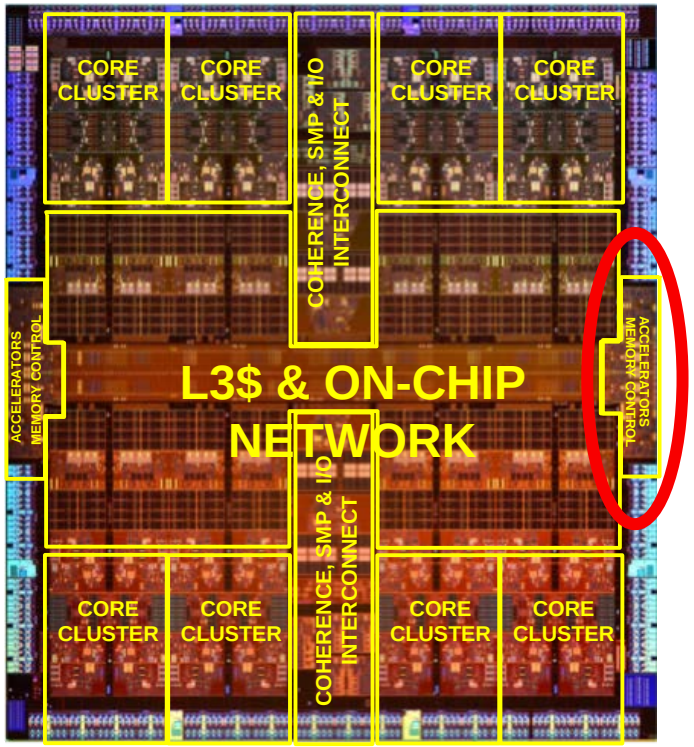
- **TODAY:** Compress FOR QUERY uses value compression
  - Fast since queries run directly on compressed data
  - Lightweight compression, lower compression ratio

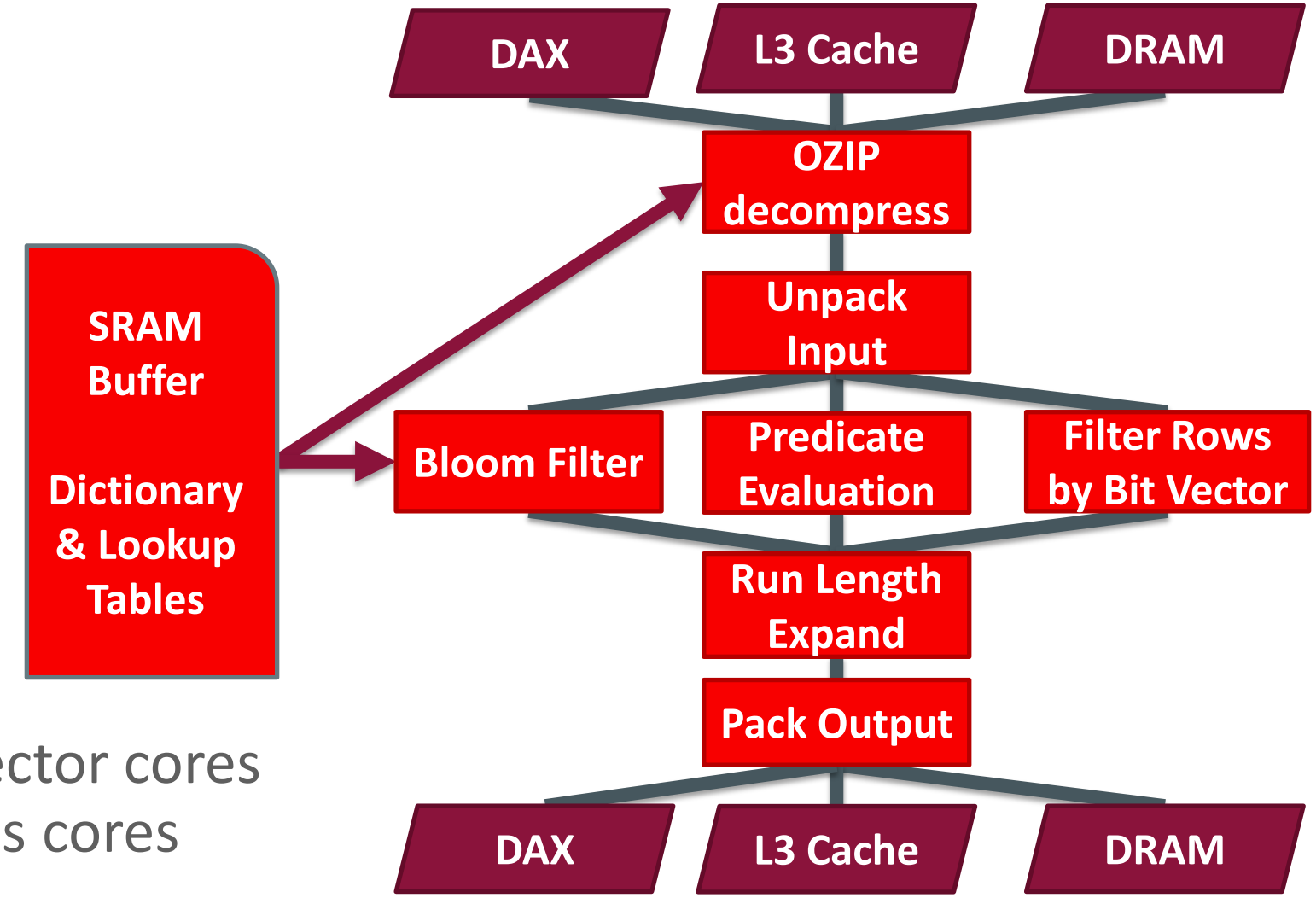
  Compress for CAPACITY uses bit pattern compression
  - Uses Oracle Zip (OZIP): 2-3x better compression than QUERY
  - Slower since data must be decompressed prior to access

- **SPARC M7:** Compress FOR QUERY HIGH uses OZIP on SPARC M7
  - DAX includes specialized OZIP decompression engine
  - Runs OZIP decompress at full memory speed, > 120 GB/sec
  - Pipelines decompression and data processing in hardware
  - Doubles memory capacity with neglible performance penalty

ORACLE

Database Accelerators (DAX): Pipelined Streaming Engines

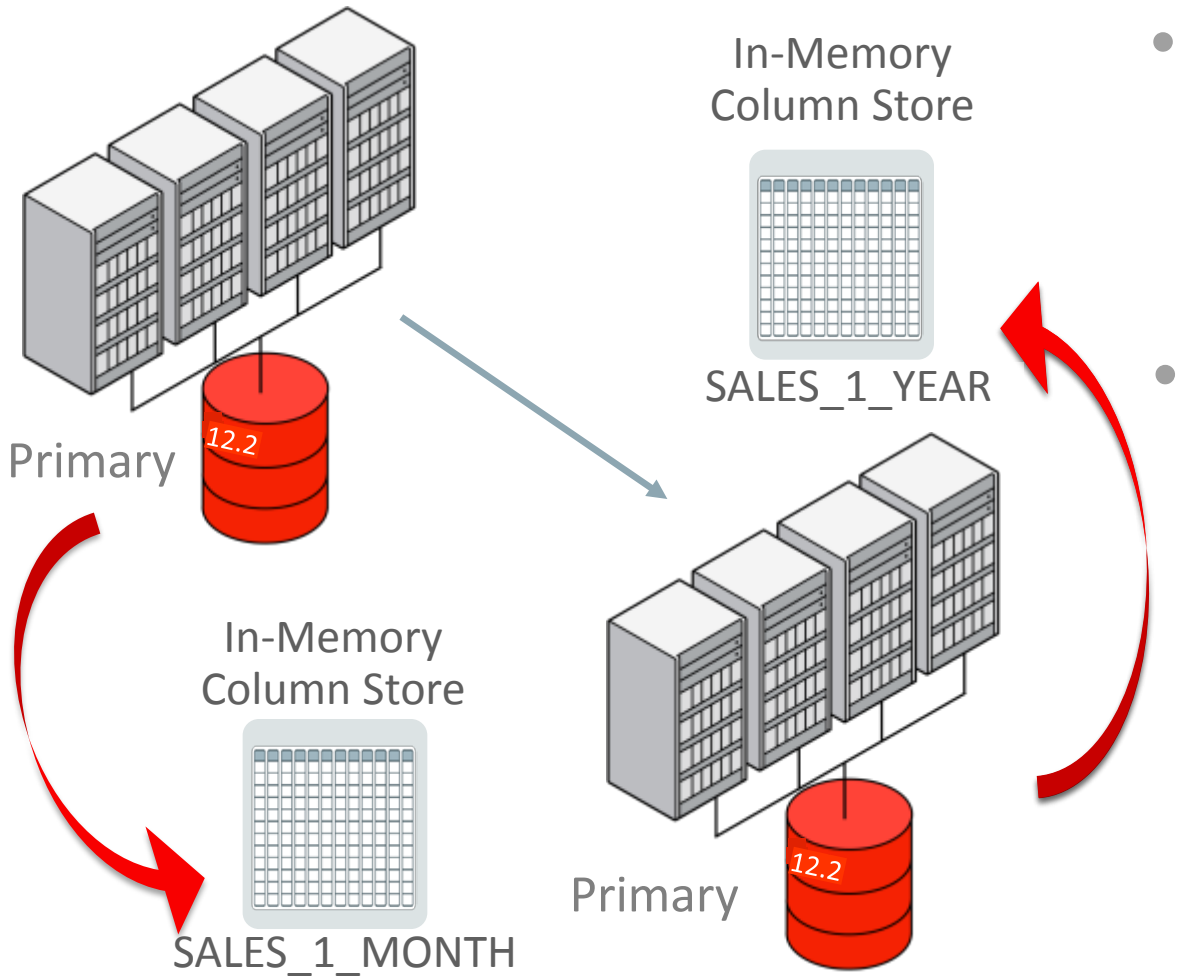




Equivalent of 32 extra vector cores plus 64 extra decompress cores

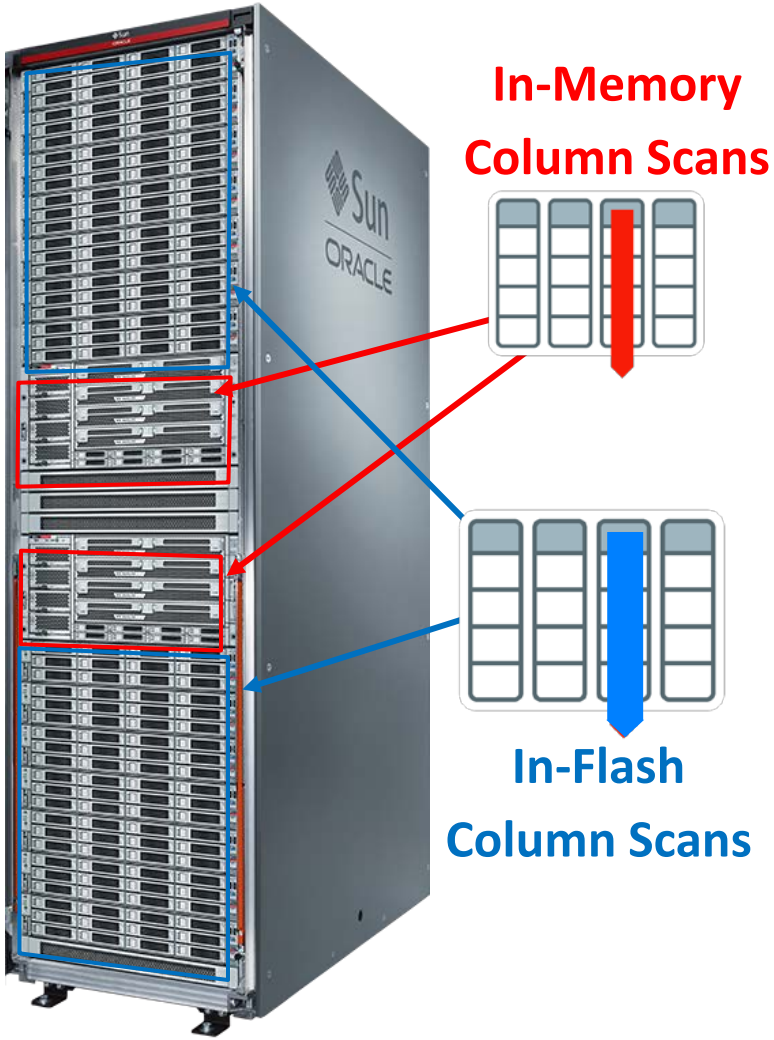
ORACLE

In-Memory Column Store on Active Data Guard



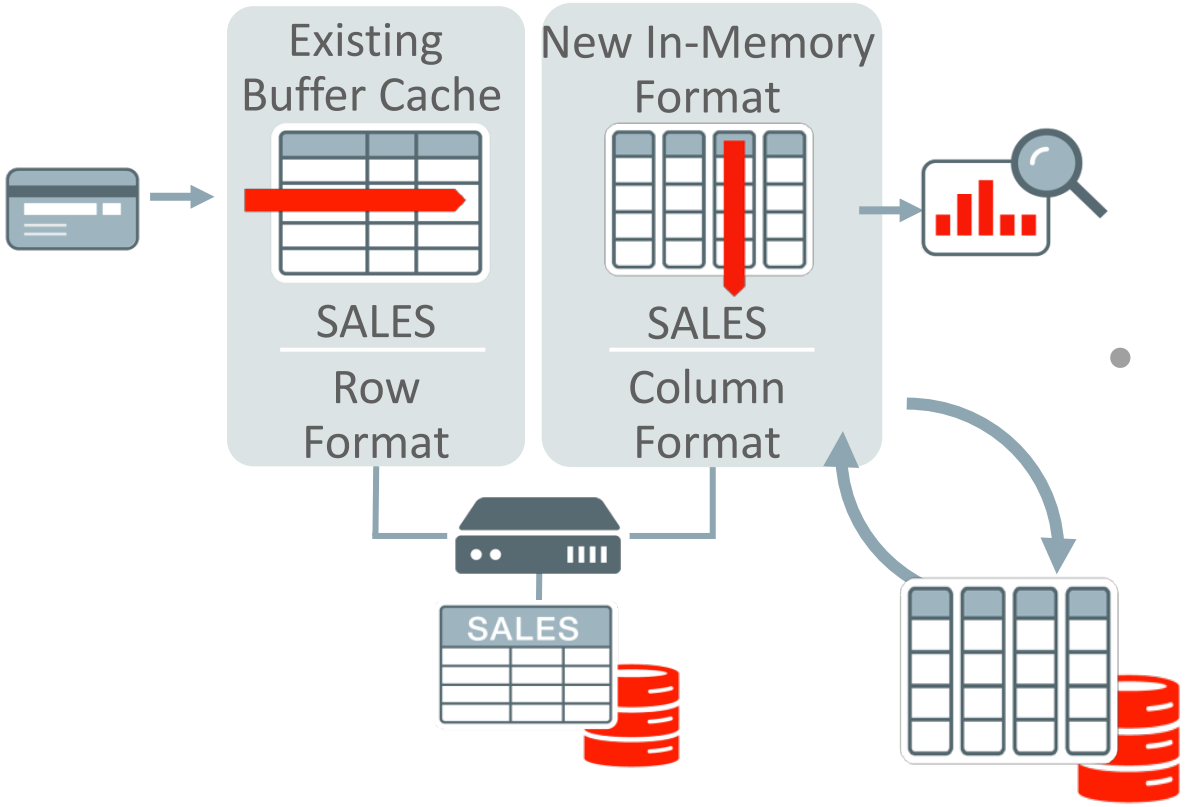
- **Today:** In-Memory queries possible only on Primary Database
  - Also on Logical Standby
- **12.2:** In-Memory queries also possible on Active Data Guard (Physical Standby)
  - Analytic reporting can be offloaded to standby
  - Completely different data can be populated into IM column store on standby
  - Different standbys can have different data in their IM column stores
  - Increases capacity and improves availability for inmemory column store

Columnar Flash Cache: In-Memory Columnar on Flash



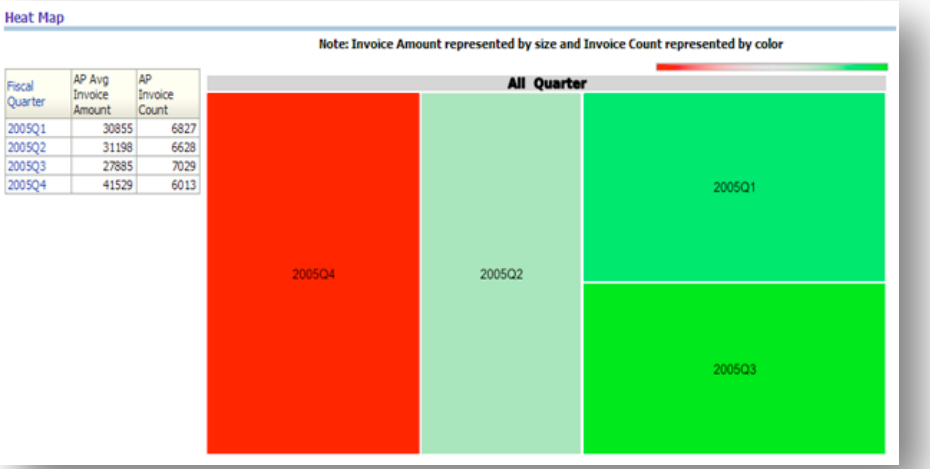
- **Today:** IM column format only in DRAM of Compute Node
  - Each database node has its own IM column store
  - Data can be distributed or duplicated across nodes
- **12.2:** IM column format on Flash Cache of Storage Node
  - Populate Flash Cache with IM columnar format
  - Smart Scans leverages all IM optimizations:
    - SIMD vector processing
    - Storage index pruning
    - Predicate / Aggregate processing optimizations
  - Multiplies effective Columnar Capacity by **10-100x**

Faster Restore of In-Memory Column Store: Fast-Start

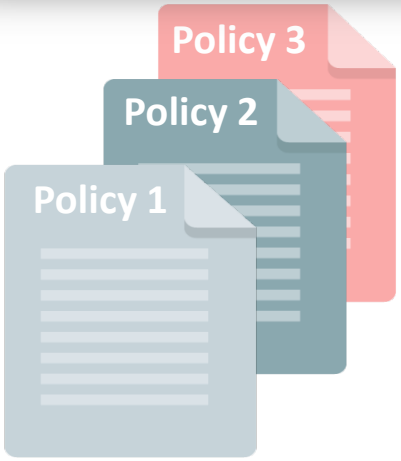


- **Today:** IM column store is always rebuilt on startup
  - Recreated from row format (populate), may take time
- **12.2:** IM column format persisted to storage
  - IM column store contents checkpointed to Secure File Lob on populate
  - When DB restarts, population is faster as population process reads the column format directly from storage
  - Faster restore (3-5x) of column store since no need to reformat data

Data Access Heatmap in 12.1: Recap

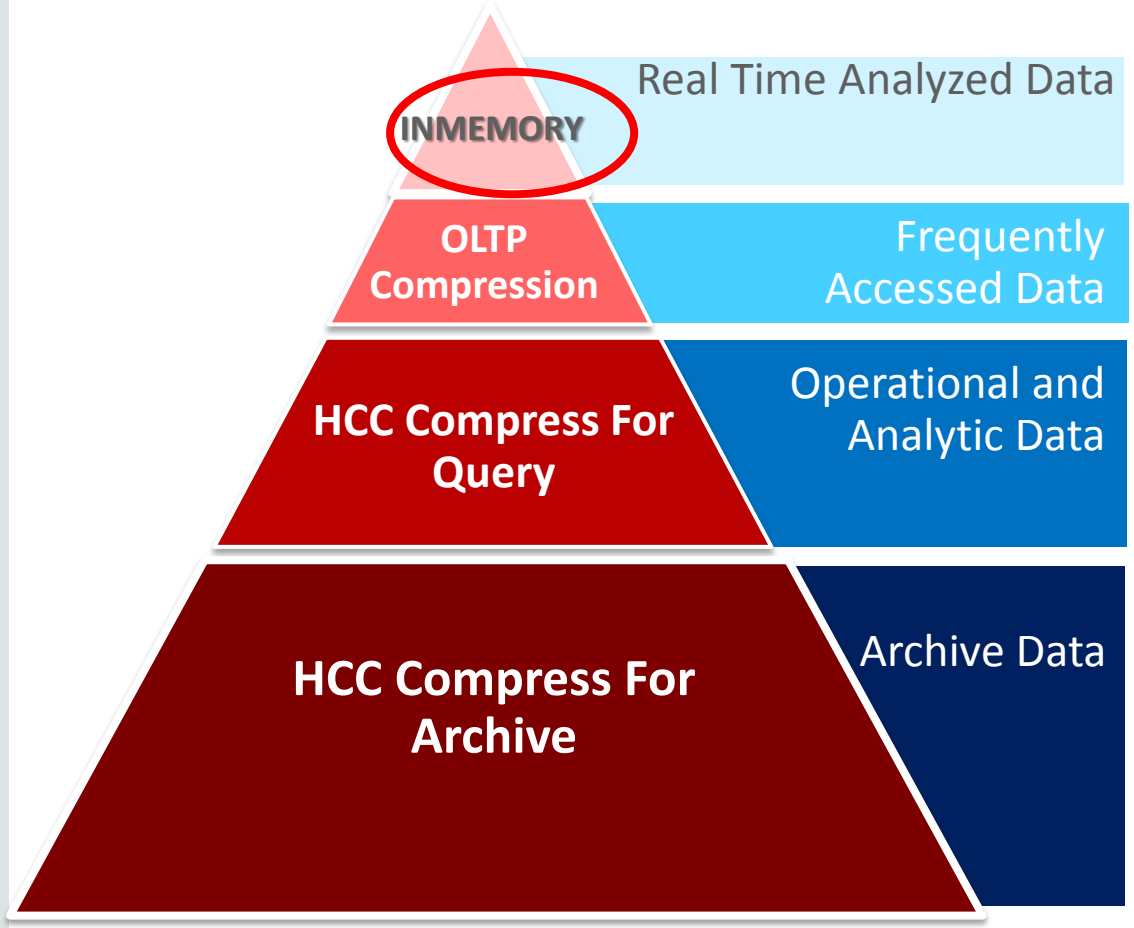


Fiscal Quarter	AP Avg Invoice Amount	AP Invoice Count
2005Q1	30855	6827
2005Q2	31198	6628
2005Q3	27885	7029
2005Q4	41529	6013



- An in-memory heat map tracks disk based block and segment access
  - Heat map is periodically written to storage
  - Data is accessible by views or stored procedures
- Users can attach policies to tables to compress or tier data based on access
  - Tables, Partitions or Sub-partitions can be moved between storage tiers and compression levels
  - Online, no impact to data availability
  - Allows automatic data tiering

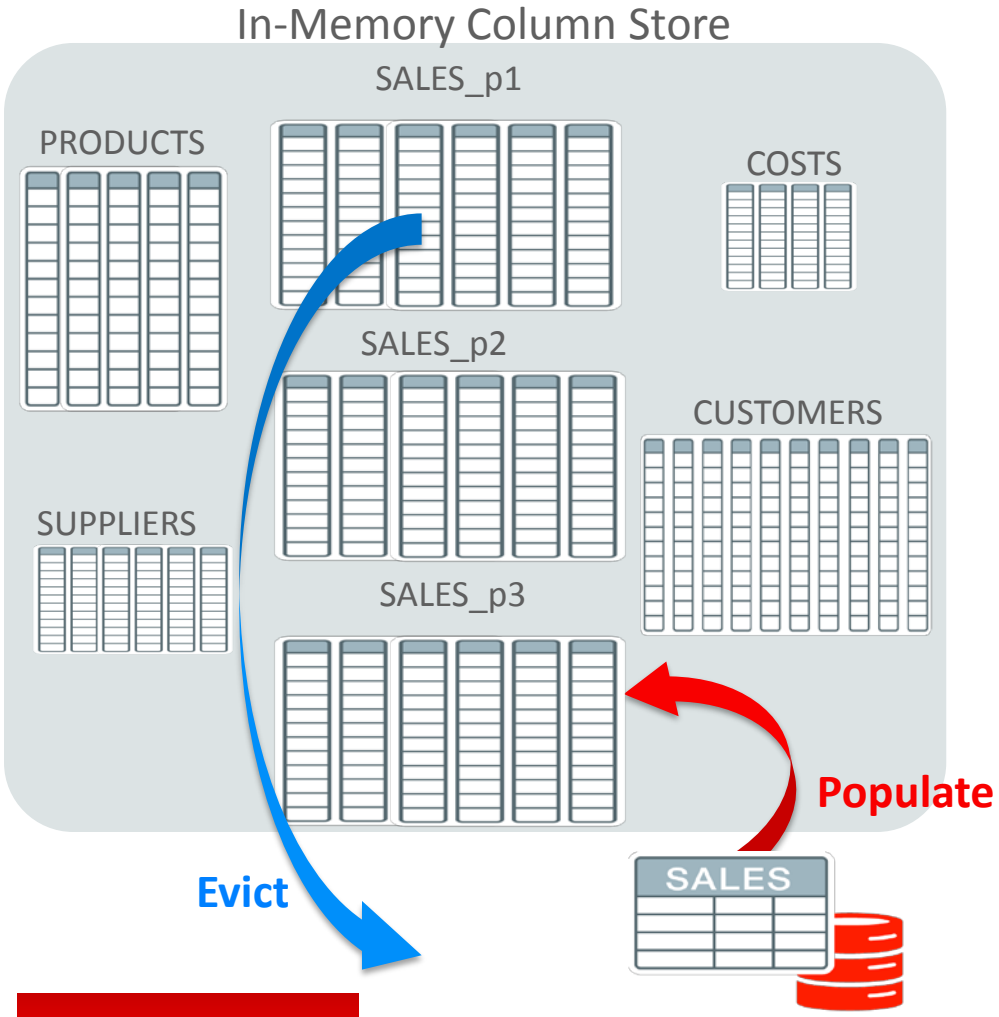
Automatic Data Tiering with Database In-Memory



- **Today:** The in-memory column store can contain a subset of database tables and even a subset of the partitions for a given table. The user must choose the subset (the in-memory advisor can help with this)
- **12.2:** IM column store is managed automatically as a new data tier
  - *Two possible levels* of automation
    - **Policy Mode** - Supports user policies to populate and evict segments
    - **Fully Automatic Mode** - Segment heat map used to add & evict segments based on memory pressure

ORACLE

Automatic Data Tiering with Database In-Memory



- **Today:** The in-memory column store can contain a subset of database tables and even a subset of the partitions for a given table. The user must choose the subset (the in-memory advisor can help with this)

- **Future:** IM column store is managed automatically as a new data tier
  - *Two possible levels* of automation
    - **Policy Mode** - Supports user policies to populate and evict segments
    - **Fully Automatic Mode** - Segment heat map used to add & evict segments based on memory pressure

Database In-Memory Release 2: **Summary**

Faster Performance

- In-Memory Expressions
- In-Memory Join Groups
- In-Memory Ordering
- SQL In Silicon

Greater Capacity

- Compression in Silicon
- In-Memory Columnar Flash Cache
- In-Memory on Active Data Guard

Easier to Manage

- Enhanced IM Advisor
- Automatic In-Memory Data Tiering

Improved High Availability

- In-Memory on Active Data Guard
- In-Memory Fast Start

ORACLE

Additional Resources

Join the Conversation

- https://twitter.com/db_inmemory
- https://blogs.oracle.com/in-memory/
- https://www.facebook.com/OracleDatabase
- http://www.oracle.com/goto/dbim.html

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

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

Any Additional Questions

- Oracle Database In-Memory Blog

Q & A

If you have more questions later, feel free to ask

ORACLE®