Oracle Database In-Memory

The Next Big Thing

Maria Colgan
Master Product Manager

#DBIM12c
Why is Oracle do this
Oracle Database In-Memory Goals

- Real Time Analytics
- Accelerate Mixed Workload OLTP
- No Changes to Applications
- Trivial to Implement

100x
What is an analytic query?

- Which products give us our highest margins?
- Who are the top 10 sales reps in the north west region this month?
- If I get a 20% discount on widget A, how much will our margins improve?
What is an analytic query?

- Scans a large amount of data
- Selects a subset of columns from wide tables
- Uses filter predicates in the form of =, >, <, between, in-list
- Uses selective join predicates that reduce the amount of data returned
- Contain complex calculations or aggregations
Row Format Databases vs. Column Format Databases

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

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

Row

- Query a single sales order in row format
  - One contiguous row accessed = FAST

Column

- Query a sales order in Column Format
  - Many column accessed = S L O W

Until Now Must Choose One Format and Suffer Tradeoffs
What is it
Breakthrough: Dual Format Database

- BOTH row and column formats for same table
- Simultaneously active and transactionally consistent
- Analytics & reporting use new in-memory Column format
- OLTP uses proven row format
How Does it work
Oracle In-Memory Columnar Technology

- Pure in-memory column format
  - Not persistent, and no logging
  - Quick to change data: fast OLTP

- Enabled at table or partition
  - Only active data in-memory

- 2x to 20x compression typical

- Available on all hardware platforms
Early User - Schneider Electric

• Global Specialist in Energy Management™
• 25 billion € revenue
• 160,000+ employees in 100+ countries
Schneider In-Memory Compression

Schneider General Ledger Compression Factors

- Over 2 billion General Ledger Entries
- 900 GB on disk
Why is an In-Memory scan faster than the buffer cache?

```
SELECT COL4 FROM MYTABLE;
```

Row Format

Buffer Cache
Why is an In-Memory scan faster than the buffer cache?

IM Column Store

<table>
<thead>
<tr>
<th>COL1</th>
<th>COL2</th>
<th>COL3</th>
<th>COL4</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

SELECT COL4 FROM MYTABLE;

RESULT
Oracle In-Memory Column Store Storage Index

Example: Find all sales from stores with a store_id of 8

- Each column is made up of multiple column units
- Min / max value is recorded for each column unit in a storage index
- Storage index provides partition pruning like performance for **ALL** queries

**Memory**

**SALES**

<table>
<thead>
<tr>
<th>Column Format</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>15</td>
</tr>
</tbody>
</table>

Example: Find all sales from stores with a store_id of 8?
Orders of Magnitude Faster Analytic Data Scans

- Each CPU core scans local in-memory columns
- Scans use super fast SIMD vector instructions
  - Originally designed for graphics & science
  - **Billions of rows/sec** scan rate per CPU core
  - Row format is millions/sec

Example: Find sales in region of CAリフォNIA

> 100x Faster
Joining and Combining Data Also Dramatically Faster

**Example:** Find total sales in outlet stores

- Converts joins of data in multiple tables into fast column scans
- Joins tables 10x faster
Generates Reports Instantly

Example: Report sales of footwear in outlet stores

- Dynamically creates in-memory report outline
- Then report outline filled-in during fast fact scan
- Reports run much faster
  - Without predefined cubes
- Also offloads report filtering to Exadata Storage servers
Schneider Speedup Across 1545 Queries

7x to 128x faster

Seconds per Query

- 2 billion General Ledger Entries
- 1545 queries
  - Currently take 34 hours to complete
  - Combination of filter queries, aggregations and summations

Buffer Cache

IN-MEMORY

Million rows returned by query
Schneider Speedup vs. Disk

From 62x to 3259x faster

Seconds per Query

<table>
<thead>
<tr>
<th>Million rows returned</th>
<th>Disk</th>
<th>IN-MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000M</td>
<td>800</td>
<td>0</td>
</tr>
<tr>
<td>300M</td>
<td>800</td>
<td>0</td>
</tr>
<tr>
<td>30M</td>
<td>800</td>
<td>0</td>
</tr>
<tr>
<td>5M</td>
<td>800</td>
<td>0</td>
</tr>
<tr>
<td>0.5M</td>
<td>800</td>
<td>0</td>
</tr>
</tbody>
</table>
How does it impact OLTP environments
Complex OLTP is Slowed by Analytic Indexes

- Most Indexes in complex OLTP (e.g. ERP) databases are only used for analytic queries.
- Inserting one row into a table requires updating 10-20 analytic indexes: Slow!
- Indexes only speed up predictable queries & reports.

<table>
<thead>
<tr>
<th>Table</th>
<th>1 – 3 OLTP Indexes</th>
<th>10 – 20 Analytic Indexes</th>
</tr>
</thead>
</table>
OLTP is Slowed Down by Analytic Indexes

Insert rate decreases as number of indexes increases

Rows Inserted per Second

Inserts/Second

# of Fully Cached Indexes (Disk Indexes are much slower)

Statistic

- Inserts/second
Column Store Replaces Analytic Indexes

- Fast analytics on any columns
- Better for unpredictable analytics
- Less tuning & administration
- Column Store not persistent so update cost is much lower
  - OLTP & batch run faster
Schneider Update Transactions Speedup

From 5x to 9x faster

- Data – Sales Accounts
- Main table has 1 Primary Key + 21 secondary indexes
- Test - 303 million transactions
  – Currently takes 21 hours

- **Primary Index Only**
- **Primary Index Plus In-Memory Columns**
- **All Indexes**
Schneider Storage Reduction

Over 70% reduction in storage usage due to analytic index removal

![Graph showing reduction in storage usage](image)
How can I scale this solution
**Scale-Out** In-Memory Database to Any Size

- Scale-Out across servers to grow memory and CPUs
- In-Memory *queries parallelized* across servers to access local column data
- **Direct-to-wire** InfiniBand protocol speeds messaging on Engineered Systems
In-Memory Speed + Capacity of Low Cost Disk

- Size not limited by memory
- Data transparently accessed across tiers
- Each tier has specialized algorithms & compression

Simultaneously Achieve:
- **Speed** of DRAM
- **I/Os** of Flash
- **Cost** of Disk
Scale-Up for Maximum In-Memory Performance

M6-32
Big Memory Machine

32 TB DRAM
32 Socket
3 Terabyte/sec Bandwidth

- Scale-Up on large SMPs
- Algorithms NUMA optimized
- SMP scaling removes overhead of distributing queries across servers
- Memory interconnect far faster than any network
Oracle In-Memory: Industrial Strength Availability

- Pure In-Memory format does not change Oracle’s storage format, logging, backup, recovery, etc.
- All Oracle’s proven availability technologies work transparently
- Protection from all failures
  - Node, site, corruption, human error, etc.
Oracle Database In-Memory: Unique Fault Tolerance

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

Only Available on Engineered Systems
How easy is it to get started
Oracle In-Memory: Simple to Implement

1. Configure Memory Capacity
   • `inmemory_size = XXX GB`

2. Configure tables or partitions to be in memory
   • `alter table | partition ... inmemory;`

3. Later drop analytic indexes to speed up OLTP
“In terms of how easy the in-memory option was to use, it was actually almost boring. It just worked – just turn it on, select the tables, nothing else to do.”

– Mark Rittman
Chief Technical Officer
Rittman Mead
Oracle In-Memory Requires Zero Application Changes

**Full Functionality**
- **ZERO restrictions** on SQL

**Easy to Implement**
- No migration of data

**Fully Compatible**
- All existing applications run unchanged

**Fully Multitenant**
- Oracle In-Memory is Cloud Ready

**Uniquely Achieves All In-Memory Benefits With No Application Changes**
“Oracle Database In-Memory made our slowest financial queries faster out-of-the-box; then we dropped indexes and things just got faster.”

– Evan Goldberg
Co-Founder, Chairman, CTO
NetSuite Inc.
“We see clear benefit from the Oracle In Memory for our users. Our existing applications were transparently able to take advantage of them and no application code changes were required”

– Scott VanValkenburgh, SAS
# Oracle Applications In-Memory Examples

<table>
<thead>
<tr>
<th>Oracle Application Module</th>
<th>Improvement</th>
<th>Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-Memory Cost Management</td>
<td>1003x Faster</td>
<td>58 hours to 3.5 mins</td>
</tr>
<tr>
<td>In-Memory - Financial Analyzer</td>
<td>1,354x Faster</td>
<td>4.3 hours to 11 seconds</td>
</tr>
<tr>
<td>In-Memory Sales Order Analysis</td>
<td>1,762x Faster</td>
<td>22.5 minutes to &lt; 1 sec</td>
</tr>
<tr>
<td>Subledger Period Close</td>
<td>200x Faster</td>
<td>600 seconds to 3 secs</td>
</tr>
<tr>
<td>Call Center Ad-hoc query pattern</td>
<td>1247x Faster</td>
<td>129 seconds to &lt; 1 secs</td>
</tr>
</tbody>
</table>
What’s the catch
Getting The Most From In-Memory

Understand Where it Helps

• Fast cars speed up travel, not meetings
• In-Memory speeds up analytic data access, not:
  – Network round trips, logon/logoff
  – Parsing, PL/SQL, complex functions
  – Data processing (as opposed to access)
    • Complex joins or aggregations where not much data is filtered before processing
  – Load and select once – Staging tables, ETL, temp tables

Know your bottleneck!
Getting The Most From In-Memory

The Driver Matters

• Avoid stop and go traffic
  – Process data in sets of rows in the Database
  – Not one row at a time in the application

• Plan ahead, take shortest route
  – Help the optimizer help you: Gather representative set of statistics using DBMS_STATS

• Use all your cylinders
  – Enable parallel execution
  – In-Memory removes storage bottlenecks allowing more parallelism
In-Memory Use Cases

**OLTP**
- Real-time reporting directly on OLTP source data
- Removes need for separate ODS
- Speeds data extraction

**Data Warehouse**
- Staging/ETL/Temp not a candidate
  - Write once, read once
- All or a subset of Foundation Layer
  - For time sensitive analytics
- Potential to replace Access Layer
Where can I get more information
Additional Resources

Join the Conversation
- https://twitter.com/db_inmemory
- https://www.facebook.com/OracleDatabase

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
- Database Industry Experts Discuss Oracle Database In-Memory (11:10)
- Software on Silicon

Any Additional Questions
- Oracle Database In-Memory Blog
- My email: maria.colgan@oracle.com