ORACLE

Oracle Database In-Memory

AskTOM Office Hours, In-Memory Expressions Update

November 18, 2021

Andy Rivenes
Database In-Memory Product Manager
Twitter: @TheInMemoryGuy
Email: andy.rivenes@oracle.com

Auro Mishra
Director of Development, Data
Email: aurosish.mishra@oracle.com

Previous Database In-Memory Ask TOM Office Hours

Additional Details on Features Covered Today

- In-Memory Expressions
- Database In-Memory Office Hours: Focus on JSON
- Database In-Memory 21c New Features

In-Memory Expressions Review

In-Memory Expressions

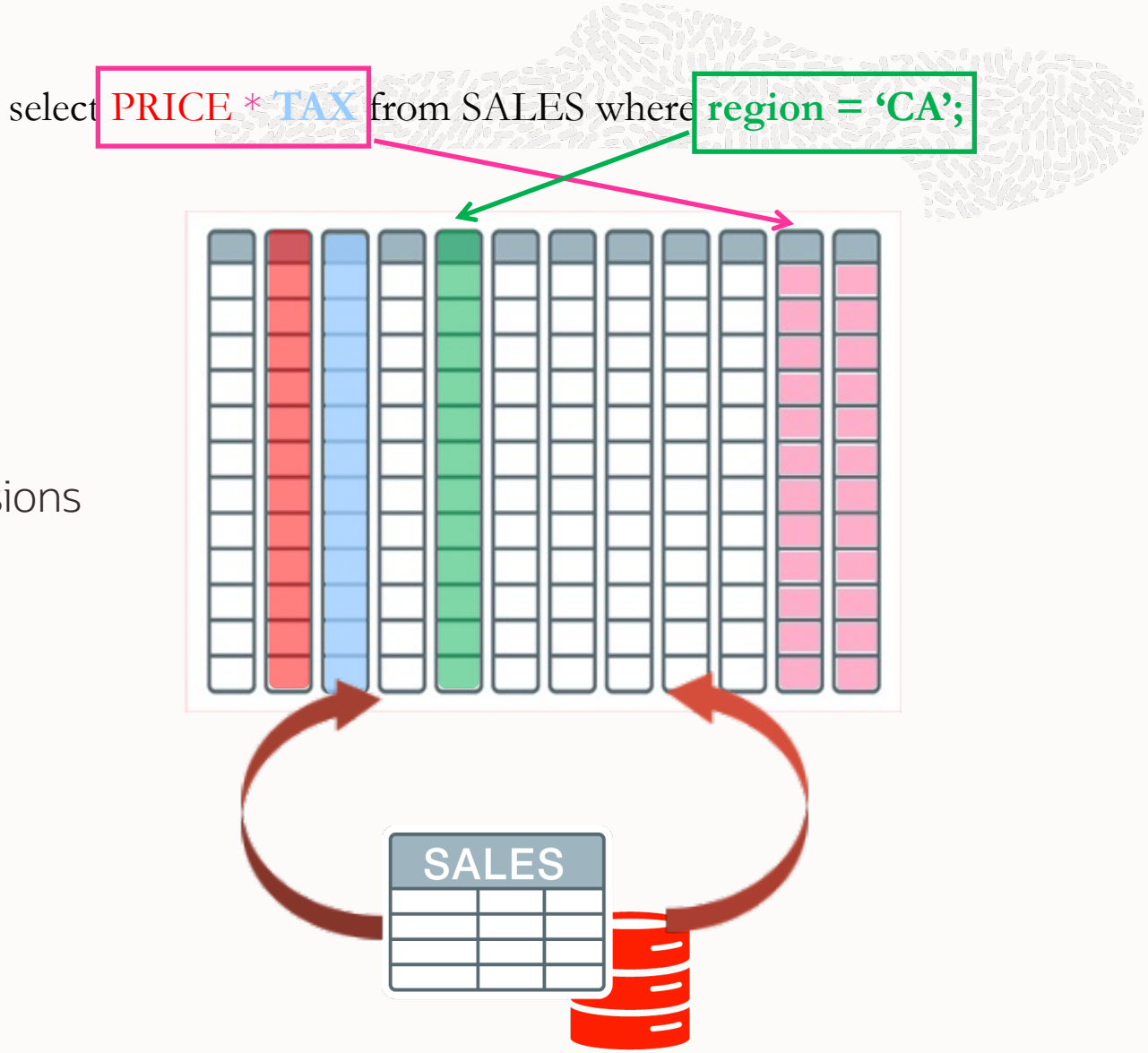
What are they?

- “An expression is a combination of one or more values, operators, and SQL or PL/SQL functions (`DETERMINISTIC` only) that resolve to a value”
- In-Memory expressions "pre-compute" frequently evaluated expressions
- In-Memory expressions (IME) can be created for:
  - Virtual columns
  - Automatic capture
    - Frequently evaluated query expressions
    - Other useful internal computations (join hash values and data conversions)
- Repeated expression evaluation can be computationally expensive
- Significant performance increases realized for very large data sets

In-Memory Expressions

How do they work?

- In-Memory only columns
  - User defined virtual columns (static)
  - Automatic capture (dynamic)
    - Captured by the Optimizer (ESS)
    - Top 20 most frequently accessed expressions
- Reduce repeated evaluations
  - Save CPU by only calculating once
- Still supports other In-Memory optimizations (min/max pruning, SIMD, etc.)



In-Memory Expressions

New Statistics

Generally follows the same statistics as for CUs

- IM populate EUs ...
- IM prepopulate EUs ...
- IM repopulate EUs ...
- IM scan EUs ...

In-Memory Expressions Example

```
select lo_shipmode, sum(lo_ordtotalprice),
  sum(lo_ordtotalprice - (lo_ordtotalprice*(lo_discount/100)) + lo_tax)
discount_price
from   LINEORDER
group by
  lo_shipmode;
```

In-Memory Expressions Example

```
SQL> Select lo_shipmode, sum(lo_ordtotalprice),
  2          sum(lo_ordtotalprice - (lo_ordtotalprice*(lo_discount/100)) + lo_tax)
discount_price
  3   From    LINEORDER
  4   group by
  5     lo_shipmode;

LO_SHIPMOD SUM(LO_ORDTOTALPRICE)       DISCOUNT_PRICE
---------- --------------------- --------------------
AIR                 161811429297122   153751580704864.92
Rows Deleted ...
TRUCK               161820421065658   153761833420584.79

7 rows selected.

Elapsed: 00:00:09.88
SQL>
```

In-Memory Expressions Example

```
SQL> -- Create In-Memory Column Expression
SQL>
SQL> alter table lineorder no inmemory;

Table altered.

SQL> alter table lineorder add v1 invisible as
(lo_ordtotalprice - (lo_ordtotalprice*(lo_discount/100)) + lo_tax);

Table altered.

SQL> alter table lineorder inmemory;

Table altered.

SQL>
```

In-Memory Expressions Example

```
select   o.owner, o.object_name, i.column_name, count(*) t_imeu,
         sum(i.length)/1024/1024 space
from     v$im_imecol_cu i, dba_objects o
where    i.objd = o.object_id
group by o.owner, o.object_name, o.subobject_name, i.column_name;
```

Owner	Object	Column Name	Total IMEUs	Used Space(MB)
SSB	LINEORDER	V1	110	690

In-Memory Expressions Example

```
SQL> Select lo_shipmode, sum(lo_ordtotalprice),
  2         sum(lo_ordtotalprice - (lo_ordtotalprice*(lo_discount/100)) + lo_tax)
discount_price
  3  From   LINEORDER
  4  group by
  5    lo_shipmode;

LO_SHIPMOD SUM(LO_ORDTOTALPRICE)       DISCOUNT_PRICE
---------- --------------------- --------------------
AIR                 161811429297122   153751580704864.92
Rows Deleted ...
TRUCK               161820421065658   153761833420584.79

7 rows selected.

Elapsed: 00:00:04.57
SQL>
```

In-Memory Expressions Example

NAME	VALUE
IM scan CUs columns accessed	444
IM scan CUs memcompress for query low	111
IM scan CUs pcode aggregation pushdown	222
IM scan EU rows	59441176
IM scan EUs columns accessed	110
IM scan EUs memcompress for query low	110
IM scan rows	59986052
IM scan rows pcode aggregated	59986052
IM scan rows projected	777
IM scan rows valid	59986052
parse time cpu	3
parse time elapsed	3
redo size	636
session logical reads	451051
session logical reads - IM	450834

Other Features That Use In-Memory Expressions

JSON Support With Database In-Memory

JSON in Oracle Database

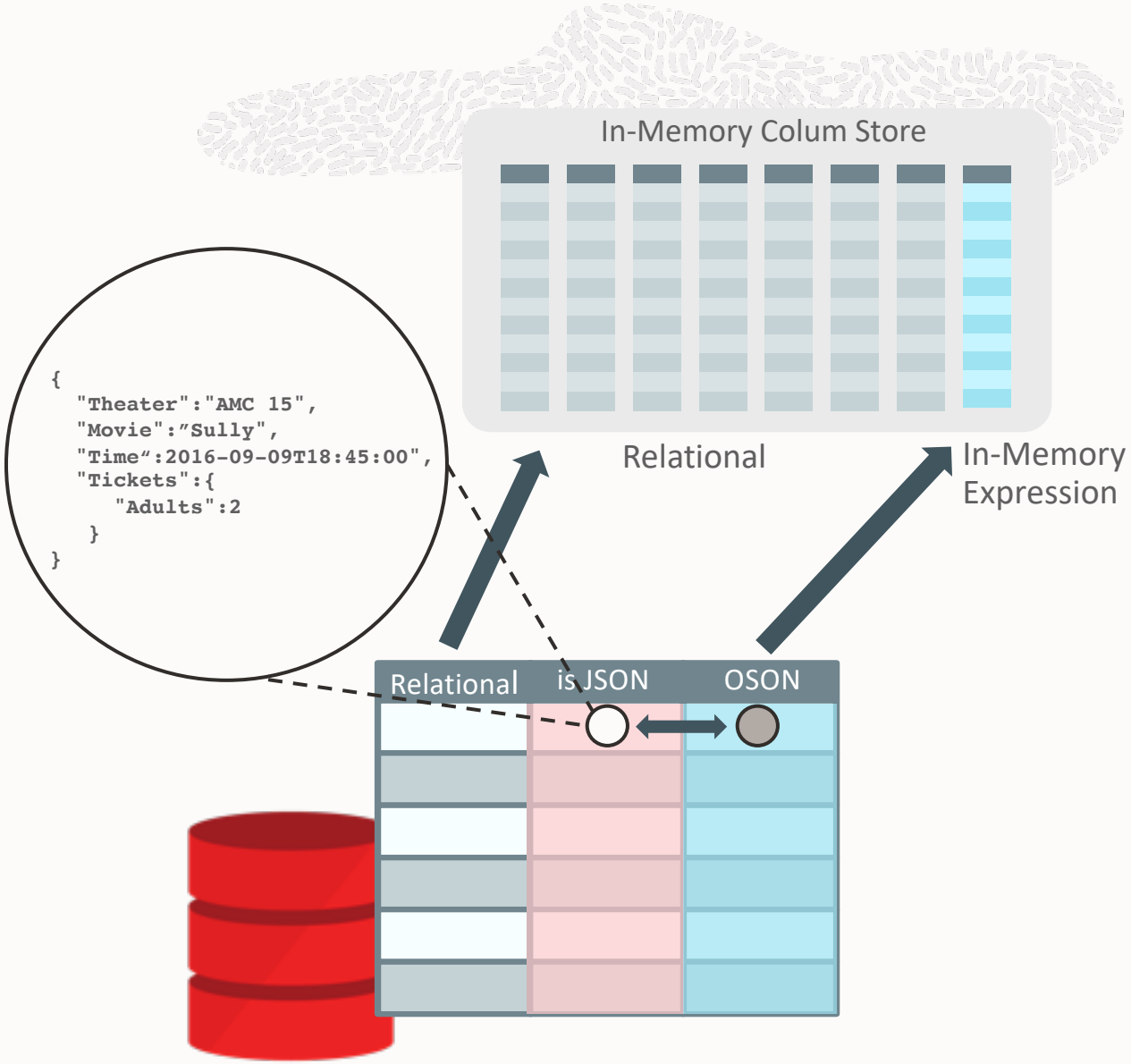
- JSON can be stored as a BLOB, VARCHAR2 or CLOB
- Multiple functions (like: json_value) and conditions (like: json_exists)
- Indexing of JSON data is supported
- Dot notation is available: po_document.**ShippingInstructions.Phone.type**
- A check constraint is used to tell Oracle the data is JSON:
  - **CONSTRAINT** ensure_json **CHECK** (po_document **IS JSON**)

Using JSON with Database In-Memory

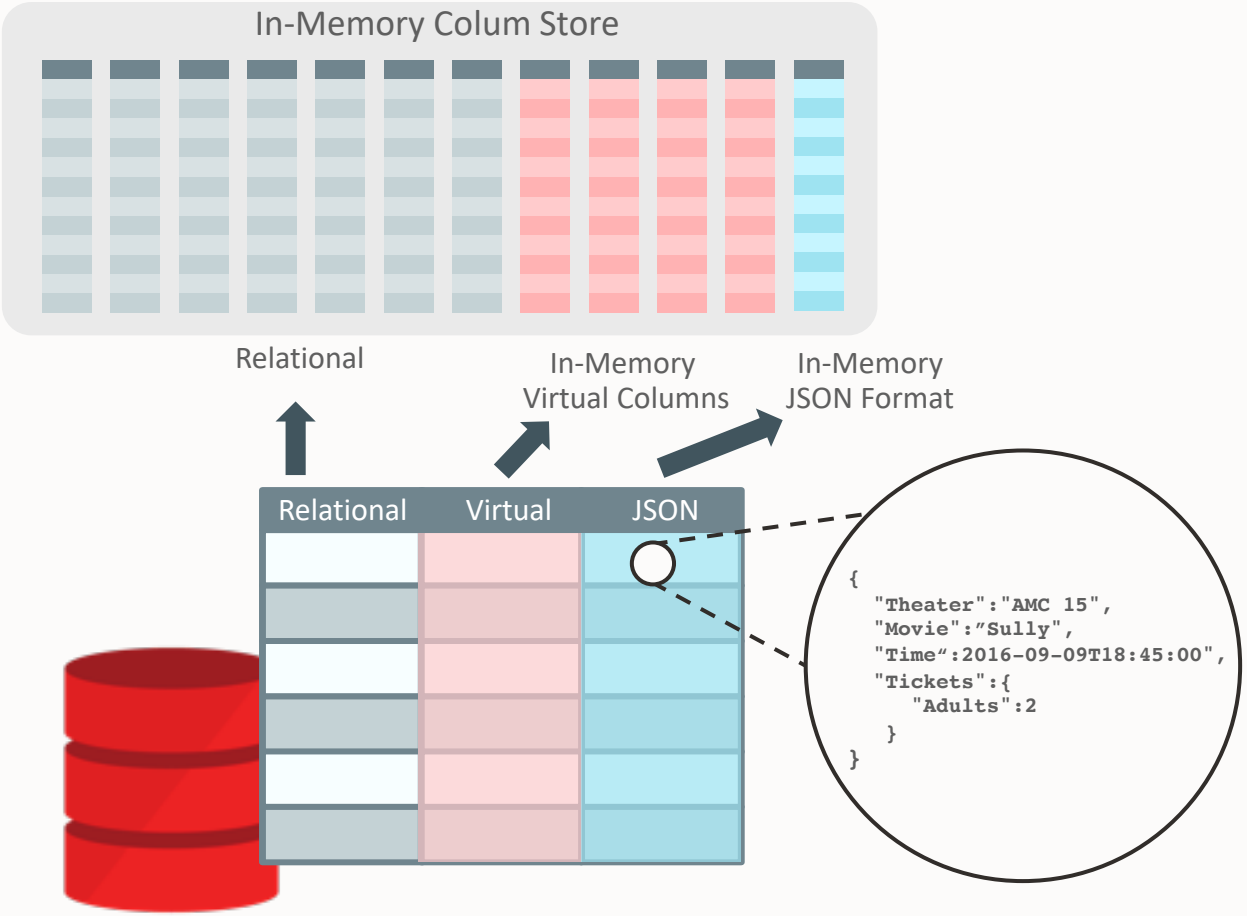
- Database compatibility set to 12.2.0.0 or higher
- max_string_size must be set to 'extended'
  - 32K VARCHAR2 columns can be populated in the IM column store
  - See blog post: Storing Values Up To 32KB In Size In The In-Memory Column Store
- inmemory_expressions_usage set to STATIC_ONLY or ENABLE
- inmemory_virtual_columns set to ENABLE (to enable population with the table)
- JSON data columns must have 'is json' check constraints
- Database In-Memory supports a special "binary" JSON format called **OSON** that performs better than row-oriented JSON

In-Memory JSON

- Use the **is JSON** check constraint to ensure well formed JSON
  - Creates a hidden virtual column
  - Data is stored in a special binary format - OSON
- 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 'Jurassic'
- **2X to 30X** performance gains observed



In-Memory JSON Data Type (21c)



- New JSON data type columns populated up to 8KB inline using an optimized binary format
- IM queries on JSON content supported for `JSON_TABLE`, `JSON_VALUE`, `JSON_EXISTS` and `JSON_TEXTCONTAINS`
  - e.g. Find movies where movie.name contains "Jurassic"
- Additional expressions can be created on JSON columns (e.g. JSON_VALUE) & stored in column store
- 2X to 30X performance gains observed versus non-IM JSON queries

JSON Datatype Example: Using Database In-Memory

```
SQL> desc J_PURCHASEORDER
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL VARCHAR2(32)
 DATE_LOADED                                        TIMESTAMP(6) WITH TIME ZONE
 PO_DOCUMENT                                        JSON
```

```
SELECT COUNT(*)
FROM
  j_purchaseorder po
WHERE
  json_exists(po.po_document,'$.ShippingInstructions?(@.Address.zipCode == 99236)');
```

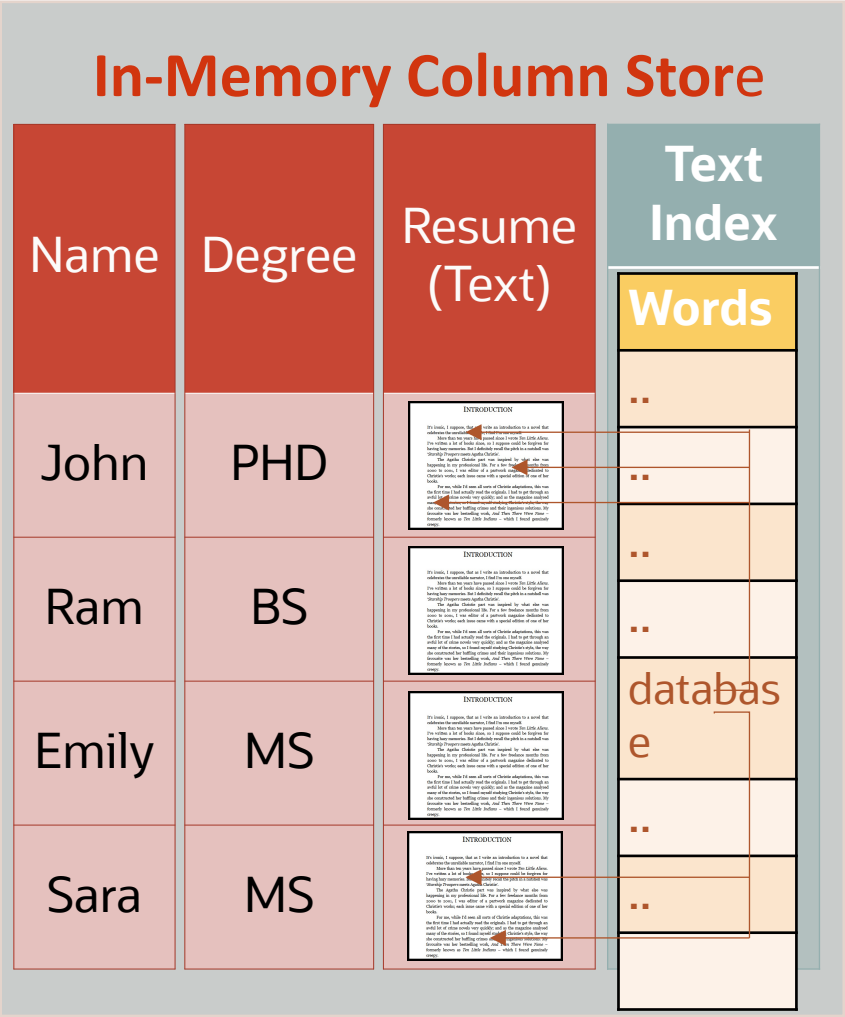
JSON Datatype Example: Execution Plan

```
-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                  |       |       |     1 (100)|          |
|   1 |  SORT AGGREGATE              |                  |     1 |  4102 |            |          |
|*  2 |   TABLE ACCESS INMEMORY FULL| J_PURCHASEORDER  |     6 | 24612 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - inmemory(JSON_EXISTS2("PO_DOCUMENT" /*+ LOB_BY_VALUE */  FORMAT OSON ,
              '$.ShippingInstructions?(@.Address.zipCode == 99236)' FALSE ON ERROR)=1)
       filter(JSON_EXISTS2("PO_DOCUMENT" /*+ LOB_BY_VALUE */  FORMAT OSON ,
              '$.ShippingInstructions?(@.Address.zipCode == 99236)' FALSE ON ERROR)=1)
```

In-Memory Full Text Columns

In-Memory Full Text Columns

- New In-Memory only Inverted Index for each text column
- Inverted Index maps words to documents containing those words
- Converged queries (*relational* + *text*) faster since executed as IM table scans
- Replaces on-disk text index for analytic workloads
- Text Queries run 3x faster



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

Query with In-Memory Full Text Enabled

ALTER TABLE chicago_text INMEMORY **TEXT (description USING 'desc_search_policy')**;

```
TABLE_NAME           COLUMN_NAME                    DATA_TYPE            DATA_LENGTH DATA_DEFAULT
-------------------- ------------------------------ -------------------- ----------- ------------------------------
CHICAGO_TEXT         SYS_IME_IVDX_83D6E07444044F7AB RAW                        32767 SYS_CTX_MKIVIDX("DESCRIPTION"
                     F0C01D713A46260                                                 RETURNING RAW(32767) USING
                                                                                     'desc_search_policy' HAVING
                                                                                     1240)
```

Query with In-Memory Full Text Enabled

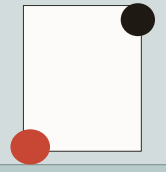
select count(*) from chicago_text where district = '009' and **CONTAINS**(description, 'BATTERY', 1) > 0;

```
-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows   | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |              |        |       | 11236 (100)|          |
|   1 |  SORT AGGREGATE              |              |      1 | 18399 |            |          |
|*  2 |   TABLE ACCESS INMEMORY FULL| CHICAGO_TEXT |  26937 |   472M| 11236  (75)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - inmemory(("DISTRICT"='009' AND SYS_CTX_CONTAINS2("DESCRIPTION" /*+ LOB_BY_VALUE */  USING '"CHICAGO"."DESC_SEARCH_POLICY"' ,
              'BATTERY' , SYS_CTX_MKIVIDX("DESCRIPTION" /*+ LOB_BY_VALUE */  RETURNING RAW(32767)))>0))
       filter(("DISTRICT"='009' AND SYS_CTX_CONTAINS2("DESCRIPTION" /*+ LOB_BY_VALUE */  USING '"CHICAGO"."DESC_SEARCH_POLICY"' ,
              'BATTERY' , SYS_CTX_MKIVIDX("DESCRIPTION" /*+ LOB_BY_VALUE */  RETURNING RAW(32767)))>0))
```

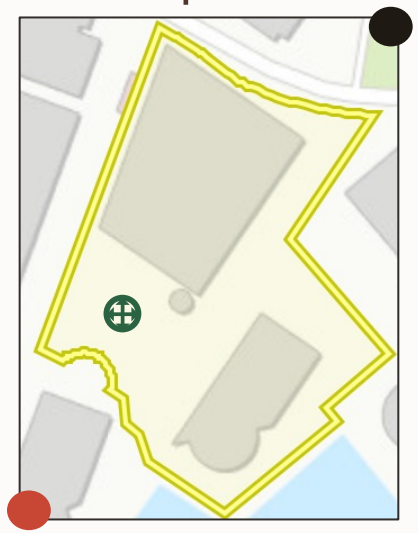
In-Memory Spatial Support

In-Memory Spatial

- New In-Memory only Spatial Summary column for each spatial column
  - Spatial Summary - compact approximation of spatial details
- Spatial Summaries stored in In-Memory formats
  - Filter values using SIMD vector scans
  - Replace R-Tree Indexes for searches
- 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		

Search 140 Million US land parcels



Which parcel is the utility valve in?

Query with In-Memory Spatial Enabled

ALTER TABLE city_points INMEMORY PRIORITY high **INMEMORY SPATIAL** (shape);

```
TABLE_NAME    COLUMN_NAME                        DATA_TYPE DATA_LENGTH DATA_DEFAULT
------------- ------------------------------  ------------- ----------- ----------------------------------------
CITY_POINTS   SYS_IME_SDO_3ACD632341514FA8B  BINARY_DOUBLE           8 SDO_GEOM_MIN_X(SYS_OP_NOEXPAND("SHAPE"))
              FD1244C9A4C375C
CITY_POINTS   SYS_IME_SDO_E484EF9DF2DB4FB4BF BINARY_DOUBLE           8 SDO_GEOM_MAX_X(SYS_OP_NOEXPAND("SHAPE"))
              F7AC9AF6081A95
CITY_POINTS   SYS_IME_SDO_DFF187A1C0C14F44BF BINARY_DOUBLE           8 SDO_GEOM_MIN_Y(SYS_OP_NOEXPAND("SHAPE"))
              F52AE162B78676
CITY_POINTS   SYS_IME_SDO_31CF528563884F3BBF BINARY_DOUBLE           8 SDO_GEOM_MAX_Y(SYS_OP_NOEXPAND("SHAPE"))
              AEF9C1A68F56A5
CITY_POINTS   SYS_IME_SDO_416B0F8BBC6A4F24BF BINARY_DOUBLE           8 SDO_GEOM_MIN_Z(SYS_OP_NOEXPAND("SHAPE"))
              BE1720E0E03AA3
CITY_POINTS   SYS_IME_SDO_5F0B63CD88274F0ABF BINARY_DOUBLE           8 SDO_GEOM_MAX_Z(SYS_OP_NOEXPAND("SHAPE"))
              AD935DA59FA9DD
```

Query with In-Memory Spatial Enabled

SELECT city_name FROM city_points c where sdo_filter(c.shape,
  sdo_geometry(2001,8307,sdo_point_type(-122.453613,37.661791,null),null,null)) = 'TRUE';

```
-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |       |       |   334 (100)|          |
|*  1 |  TABLE ACCESS INMEMORY FULL| CITY_POINTS |     1 |  3849 |   334 (100)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter((SDO_GEOM_MAX_X("SHAPE")>=SDO_GEOM_MIN_X("MDSYS"."SDO_GEOMETRY"(2001
              ,8307,"SDO_POINT_TYPE"((-122.453613),37.661791,NULL),NULL,NULL))-7.848052667402416
              6E-008D AND SDO_GEOM_MIN_X("SHAPE")<=SDO_GEOM_MAX_X("MDSYS"."SDO_GEOMETRY"(2001,83
              07,"SDO_POINT_TYPE"((-122.453613),37.661791,NULL),NULL,NULL))+7.8480526674024166E-
              008D AND SDO_GEOM_MAX_Y("SHAPE")>=SDO_GEOM_MIN_Y. <== Rest deleted for space
```

Where Can You Get More Information?

Documentation

Additional Details on Features Covered Today

- Database In-Memory Guide
- Spatial and Graph Developer's Guide
- Text Application Developer's Guide
- JSON Developer's Guide

https://blogs.oracle.com/in-memory/dbim-resources