ORACLE

Oracle Database MultiTenant/MultiModel/InMemory/ACO

Workshop

Juan Carlos Diaz
José Vázquez
Francisco Alvarez
Francisco Rivas

Madrid 18 de Enero de 2022

Safe harbor statement

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.

Overview

- Multitenant
- MultiModel - JSON
- InMemory
- ACO

Overview

- Multitenant
- Multimodel - JSON
- InMemory

Oracle 12.1 Oracle 12.2 Oracle 18 Oracle 19

Deprecation of Non-CDB Architecture

The non-CDB architecture was deprecated in Oracle Database 12c. It can be desupported and unavailable in a release after Oracle Database 19c .

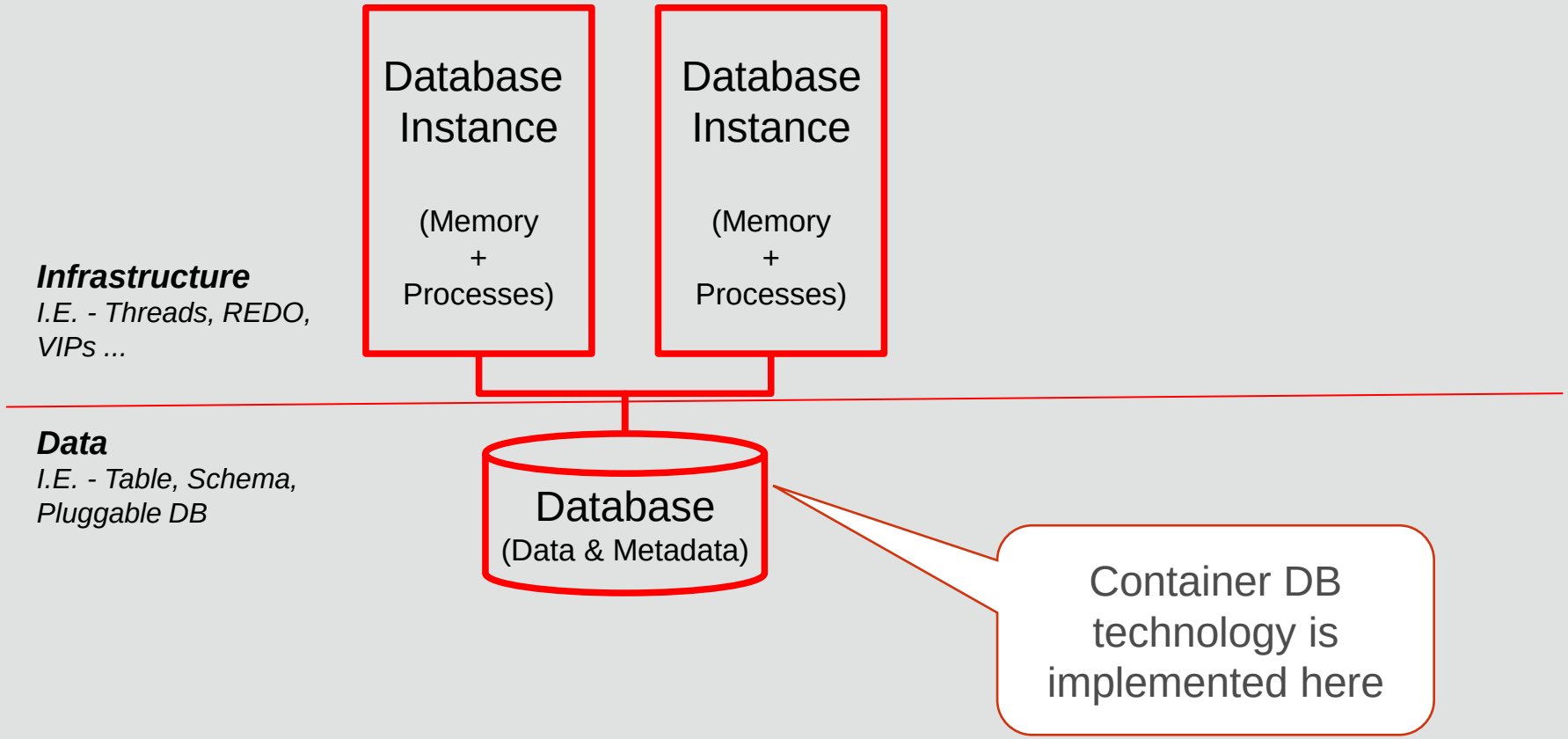
"Oracle recommends use of the CDB architecture"

ORACLE OPEN WORLD

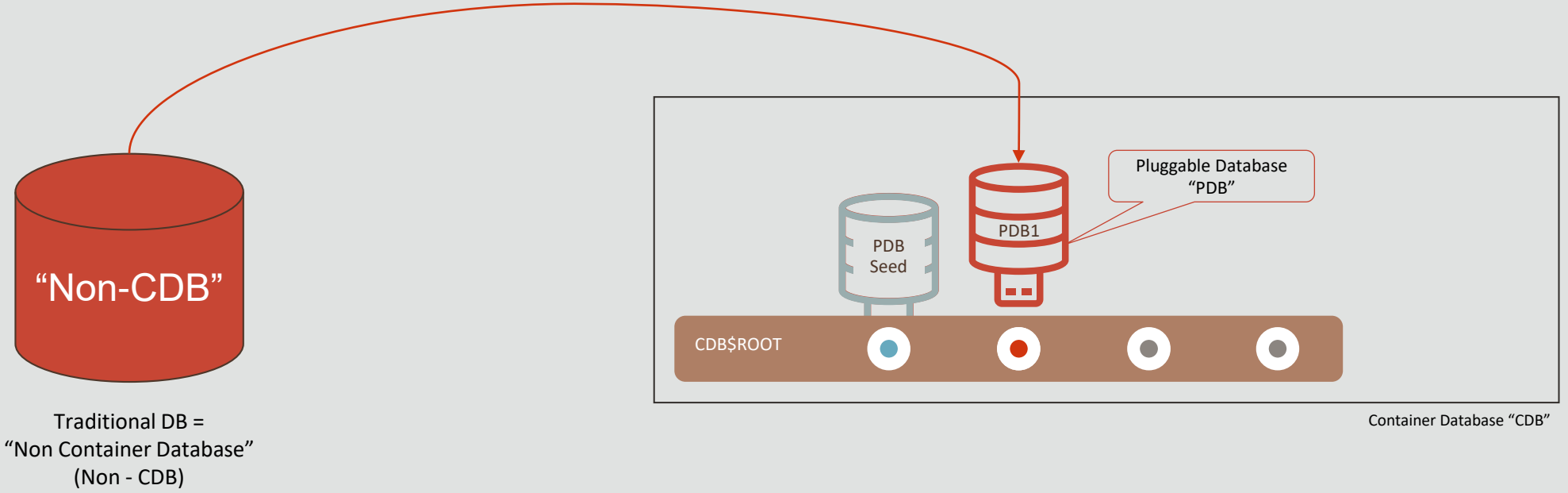
19 - Sept - 2019

*"The Oracle Database non-CDB architecture will be **de-supported** from Oracle Database 20c onwards"*

CDB architecture

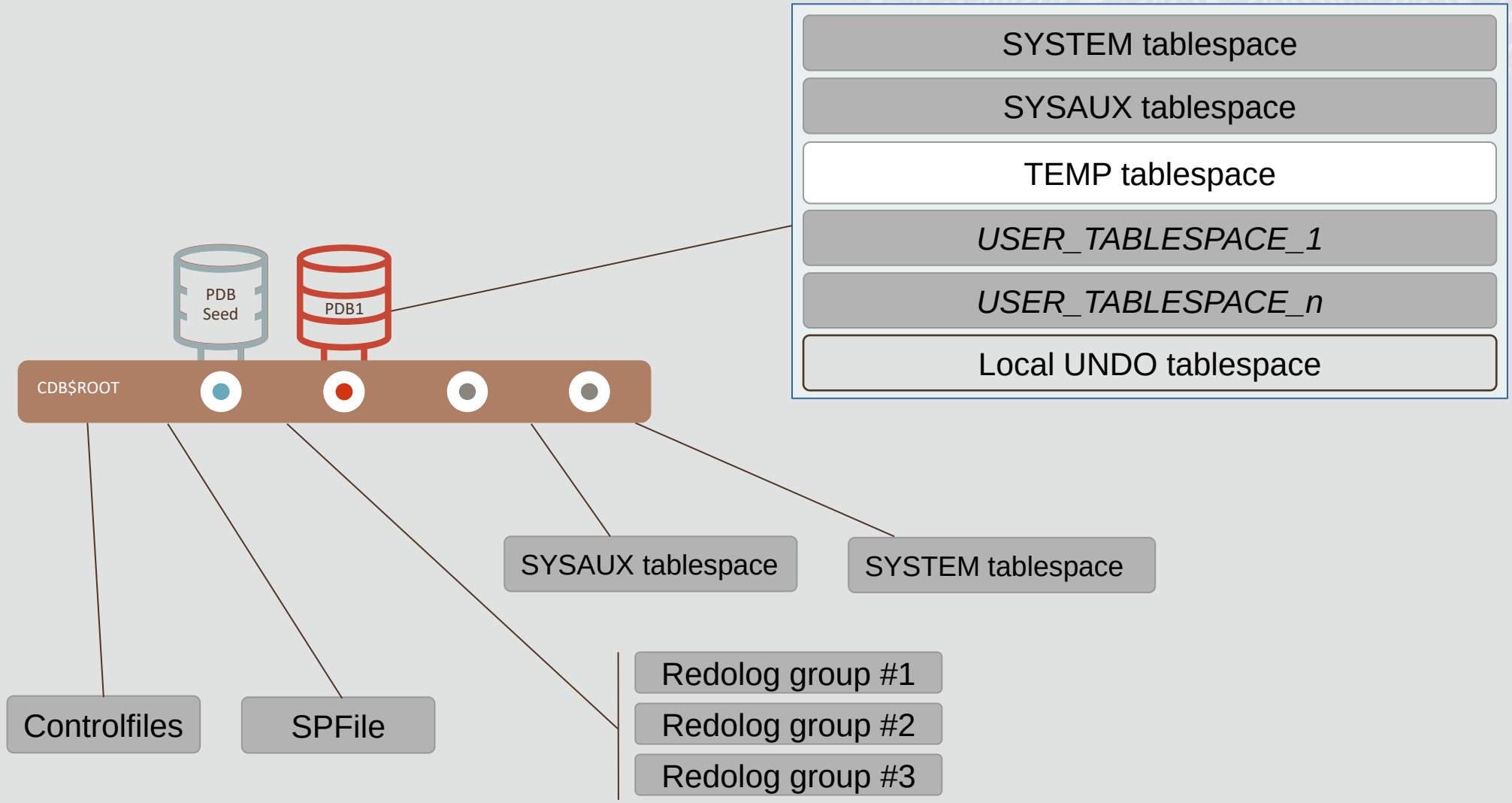


From Non-CDB to CDB architecture

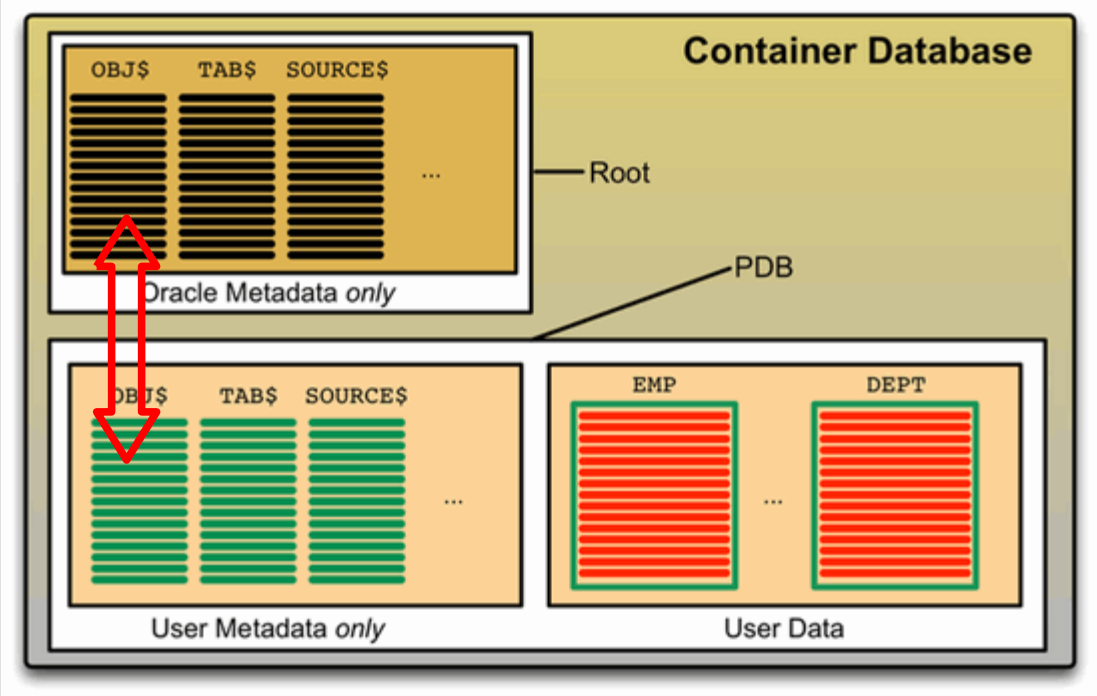


- ✓ 100% Logically Isolated.
- ✓ Same DB as it was. Same identity, same behavior.

CDB architecture



How multiple dictionaries?



Root dictionary and PDBs dictionaries are :

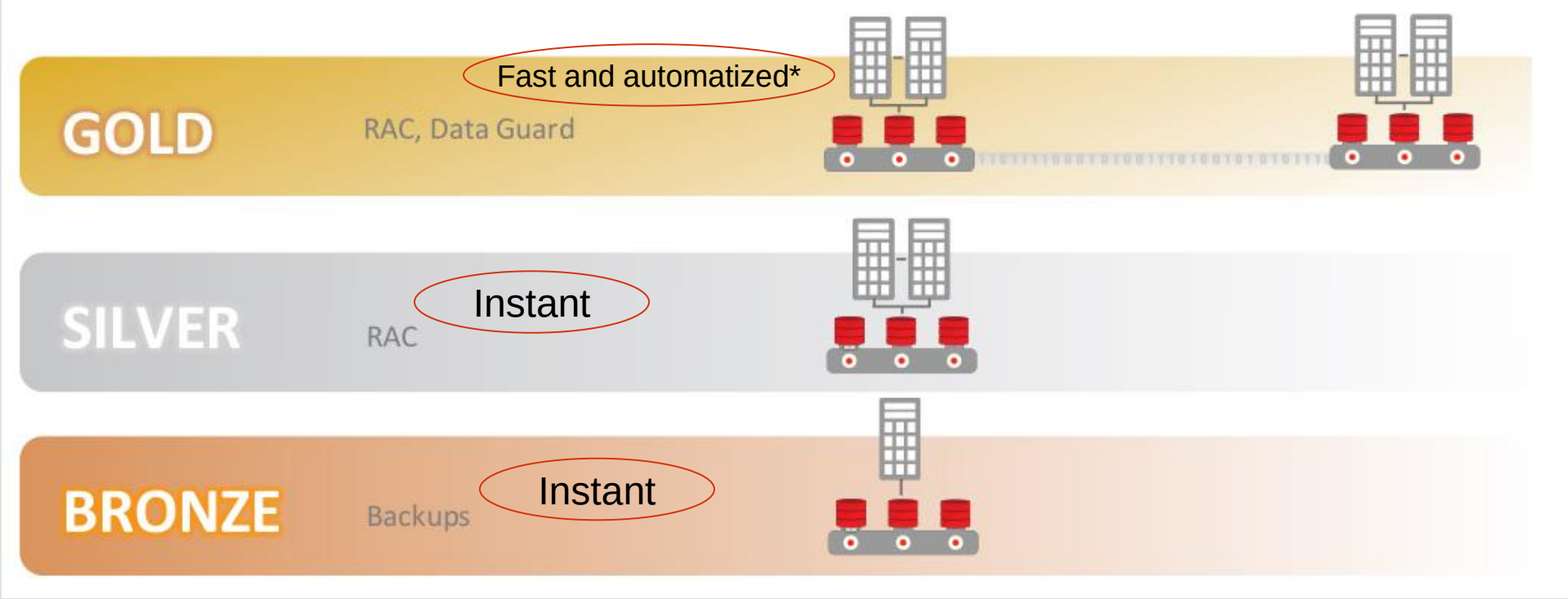
- ✓ Logically independent
- ✓ Internally connected with stubs for common data
- ✓ Same structures on each dictionary
- ✓ Different rows on each dictionary, thus “horizontal partitioned”.

PDB Means mobility

Multiple SYSTEM & SYSAUX tablespaces
Where is What?

Feature	Stored @Root	Stored @PDB
ASH*	✓	✓
AWR	✓ Root perf. info	✓ PDB perf. Info
ADDM*	✓	✓
Segment Advisor		✓
Optimizer statistics	✓	✓
DBReplay and/or SPA	✓	✓

PDB Means mobility

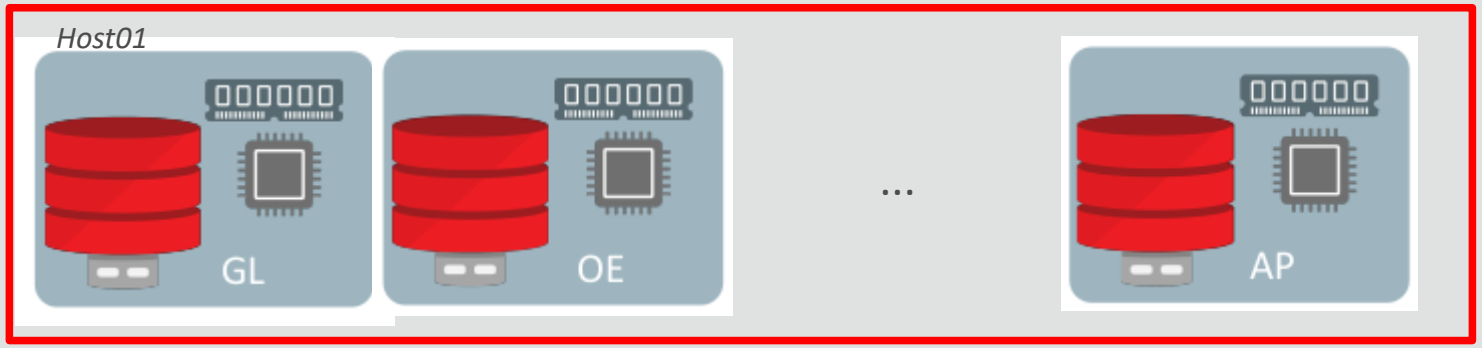


Main Use cases

1. More power under the same hardware (multitenant)
2. Resources control (multitenant & singletenant)
3. Faster, easier and new maintenances (multitenant & singletenant)

Resource isolation with the deprecated architecture

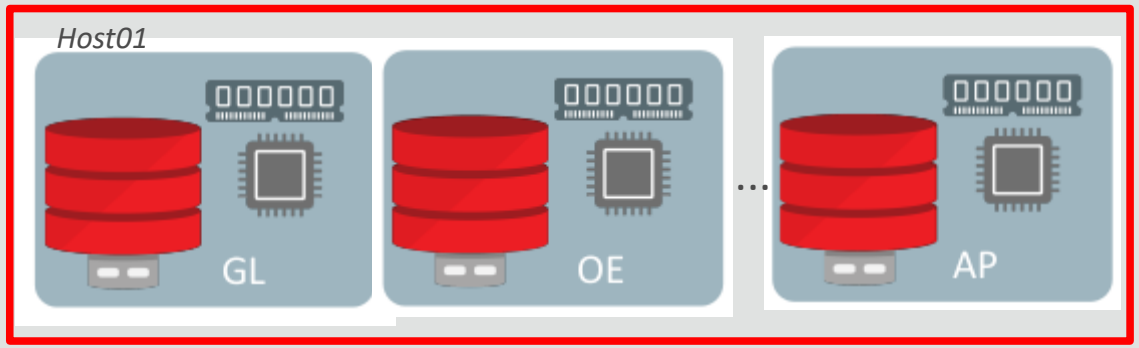
Non-CBD
19c - **30** DB



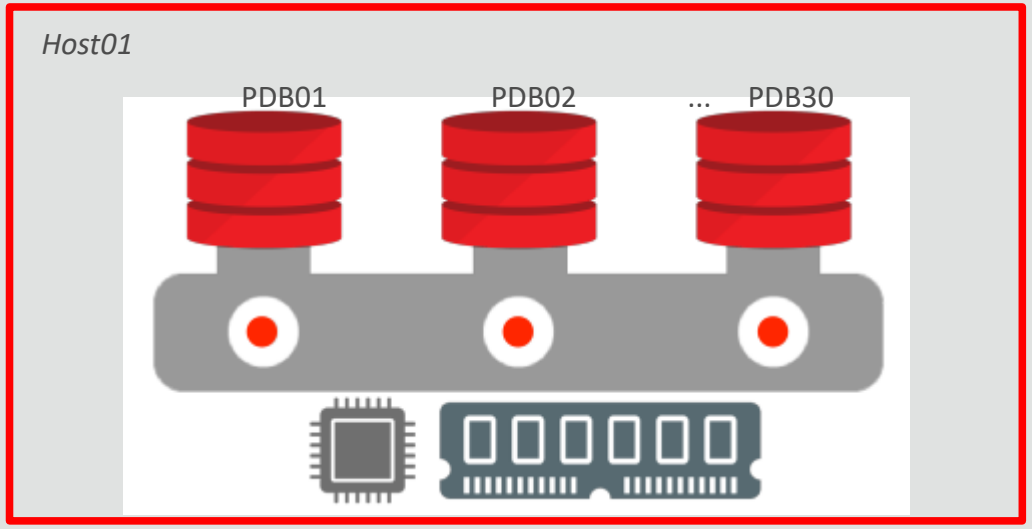
- Competing use of CPUs (full cpu per instance) or underperforming use of CPU (caging)
- Unused & unshared SGA
- Unused & unshared PGA
- *N* sets of background processes

More power under the same hardware

Non-CBD
19c - **30** DB



- Competing use of CPUs (full cpu per instance) or underperforming use of CPU (caging)
- Unused & unshared SGA
- Unused & unshared PGA
- *N* sets of background processes



- ✓ Online and immediate CPU rebalance
- ✓ Online and immediate SGA rebalance
- ✓ Shared PGA free space
- ✓ 1 set of background processes

More power under the same hardware

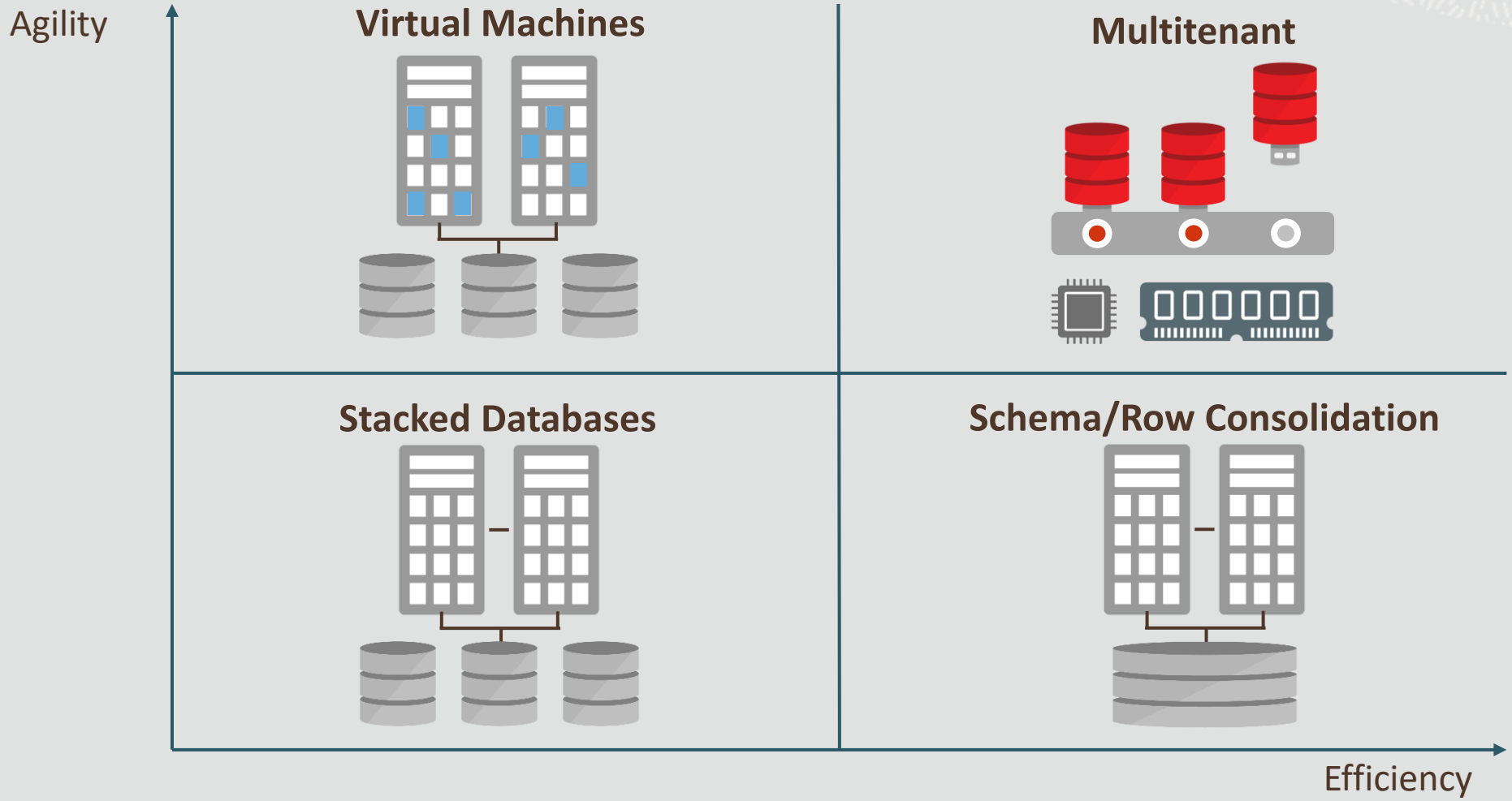
Unconsolidated

- Overheads replicated for each workload
- Competing workloads interfere with each other
- Net throughput is limited and gridlock is possible

PDB Consolidation

- Single, shared set of overheads
- Resource sharing vs isolation
- Coordination and cooperation vs competition

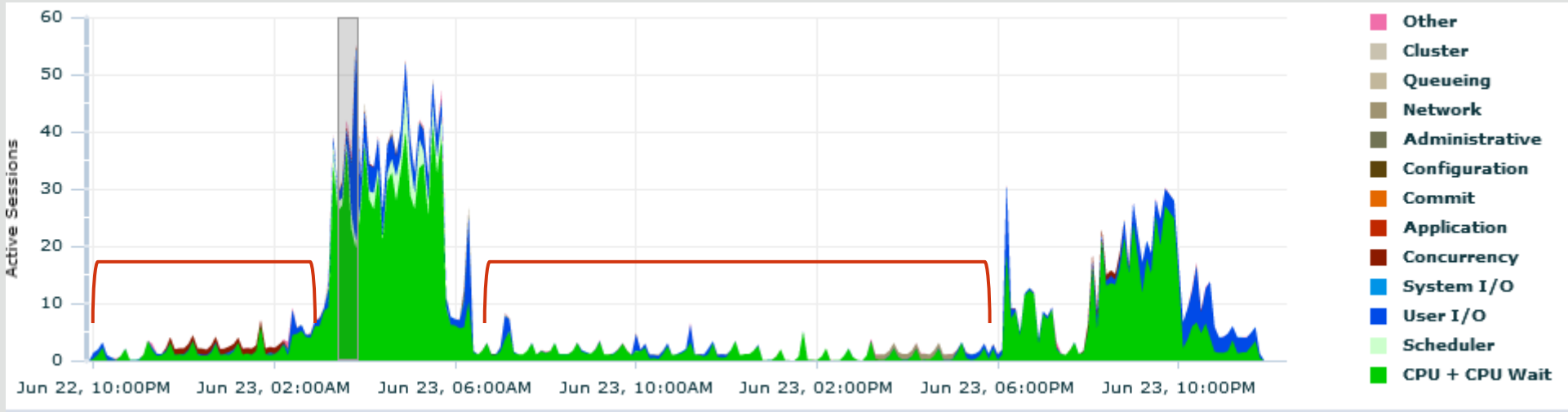
More power under the same hardware



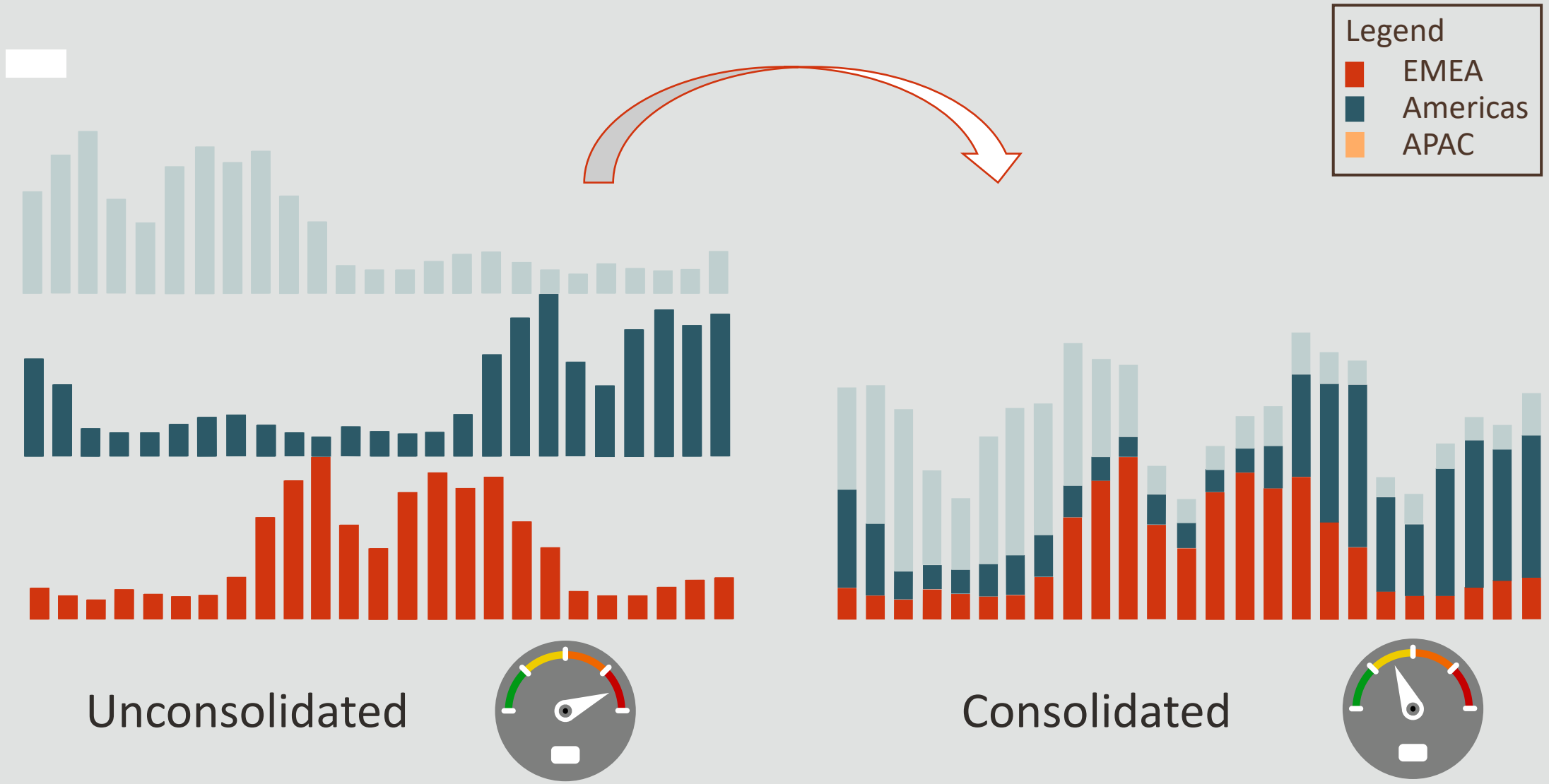
CPU Management

Why PDB online resource sharing is so capable

1. Datawarehouse reports executed at specific time-frames.
2. Reports in time for the CEO = buy HW for peak loads
3. CPU and memory wasted during idle time.



CPU Management : Greater flexibility as consolidation increases



1. More power under the same hardware (multitenant)
2. **Resources control** (multitenant & singletenant)
3. Faster, easier and new maintenances (multitenant & singletenant)

Real online control of your resources

- **CPU** : No more fixed CPU values per database; defined dynamically by priorities. Can set minimums. Can set maximums. Modifiable online. Modifiable by a schedule.
- **Memory** : All the memory accessible by all databases. SGA and PGA can be set with minimums, maximums and/or fixed values.
- **I/O** : Can define IO rate limits (both IOPS and/or MBPS) for each PDB. Can be dynamically set though priorities with Exadata *(I.O.R.M at PDB level)*
- **Parallel Execution Servers** : Parallelism capacity can be controled dynamically between PDBs by priorities.

1. More power under the same hardware (multitenant)
2. Power *with* control (multitenant & singletenant)
3. **Faster, easier and *new* maintenances** (multitenant & singletenant)

New features brought by CDB architecture

- **Fast database provisioning** : ~1 minute vs 45 minutes with old architecture
- ***Online* PDB relocation :** 2 simple SQL commands. Powerful migration option.
- ***Online* PDB clone (local or remote):** 1 or 2 simple SQL commands. RMAN not needed
- **PDB Snapshot clone:** 1 simple SQL command. Immediate. Space saver.
- **ASM Split Mirror:** 2 simple SQL commands. Immediate read/write clone.
- **Many B&R features** (PDB level recovery, PDB level flashback, etc)

Workshop

MultiTenant + JSON + InMemory + ACO

https://github.com/OracleDataManagementSpain/ConvergedDatabase/tree/master/MTJSONIMACO

18 Enero
WORKSHOP_Multitenant_Multimodel_InMemory_HOL_v3_part1of3.pdf

19 Enero
WORKSHOP_Multitenant_Multimodel_InMemory_HOL_v3_part2of3.pdf

20 Enero
WORKSHOP_Multitenant_Multimodel_InMemory_HOL_v3_part3of3.pdf

ORACLE

Oracle Database MultiTenant/MultiModel/InMemory/ACO

Workshop

Juan Carlos Diaz
José Vázquez
Francisco Alvarez
Francisco Rivas

Madrid 19 de Enero de 2022

Overview

- Multitenant
- MultiModel - JSON
- InMemory
- ACO

New features brought by CDB architecture

- **Fast database provisioning** : ~1 minute vs 45 minutes with old architecture
- ***Online* PDB relocation :** 2 simple SQL commands. Powerful migration option.
- ***Online* PDB clone (local or remote):** 1 or 2 simple SQL commands. RMAN not needed
- **PDB Snapshot clone:** 1 simple SQL command. Immediate. Space saver.
- **ASM Split Mirror:** 2 simple SQL commands. Immediate read/write clone.
- **Many B&R features** (PDB level recovery, PDB level flashback, etc)

Overview

- MultiTenant
- Multimodel - JSON
- InMemory

What is JSON?

JSON stands for **JavaScript Object Notation**

It’s a "lightweight", readable data interchange format that’s language independent

Most popular data format for new web applications

Instead of creating an entity relationship model to define all of the data the application needs and then mapping it to a set of relational tables

Storing JSON documents in the database greatly simplifies application development as the same schema-less data representation can be use in Application and the Database

What is JSON?


```
{"id":1,
 "name":"Century 16",
 "location":{"street":"Main St",
             "city":"Redwood",
             "zipCode":"94607",
             "state":"CA",
            "phoneNumber":null
             },
"ticketPrice":{"adultPrice":14.95,
               "childPrice":9.95,
               "seniorPrice":9.95
              }
}
```

- A data format that consists of one or more **name value pairs** enclosed in curly brackets
- The **name** is always a string and is separated from the value by a colon
- A **value** can be a number, string, true, false null, an object or array
  - E.g. location is an **object** as it has random set of name value pairs nested inside , enclosed in { }
  - An **array** is an ordered list of related items which could be JSON objects and is enclosed in [ ]
- Each pair is separated by a comma

Storing JSON in the Oracle Database

Table containing JSON documents

```
CREATE TABLE theater
 (
   theater_id     VARCHAR2(255),
   json_document BLOB


 );
```

- Oracle stores JSON in table columns
  - No special data type
  - Can be VARCHAR2, BLOB or CLOB
- JSON supported by all Oracle features
  - Analytics, Encryption, In-Memory, RAC, Replication, Parallel SQL, …
  - Plus can index any JSON element

Storing JSON in the Oracle Database

Table containing JSON documents

```
CREATE TABLE theater
 (
   theater_id    VARCHAR2(255),
   json_document BLOB
   CONSTRAINT is_json CHECK
    (json_document IS JSON)
 );
```

```
{id:1,
 name:"Century 16",
}
```

```
{"id":1,
 "name":"Century 16",
}
```

- Oracle stores JSON in table columns
  - No special data type
  - Can be VARCHAR2, BLOB or CLOB
- JSON supported by all Oracle features
  - Analytics, Encryption, In-Memory, RAC, Replication, Parallel SQL, …
  - Plus can index any JSON element
- **IS JSON** check constraint enforces lax JSON syntax by default
  - Does not require NAME attributes to be in double quotes

Storing JSON in the Oracle Database

Which data type to pick?

VARCHAR2

- Best performance, easy to retrieve via SELECT
- Limited max size of 32k only (with MAX_STRING_SIZE=EXTENDED)

BLOB

- Best LOB performance but not as fast as VARCHAR2
- Unlimited size, not as easy to retrieve via SELECT
- No potential characterset conversion

CLOB

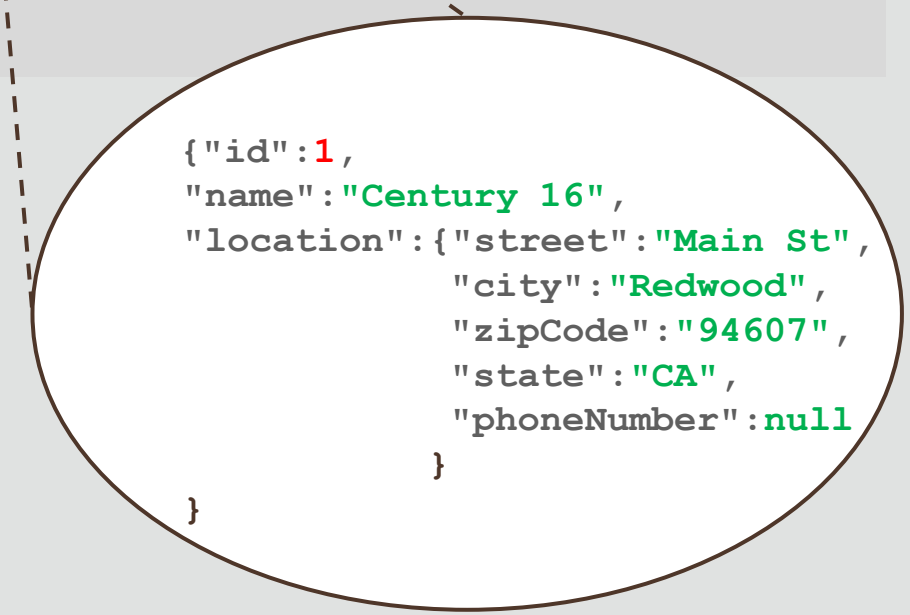
- Unlimited size, easy to retrieve via SELECT
- Potential characterset conversion (from 1 byte UTF-8 to 2 byte USC-2, double space)
- Potential bigger size on disk

Native SQL Support for JSON

12.2 JSON

Table containing JSON documents

```
SQL> desc THEATER
NAME                          TYPE
-----------------------------  ----------
JSON_DOCUMENT                  BLOB
```



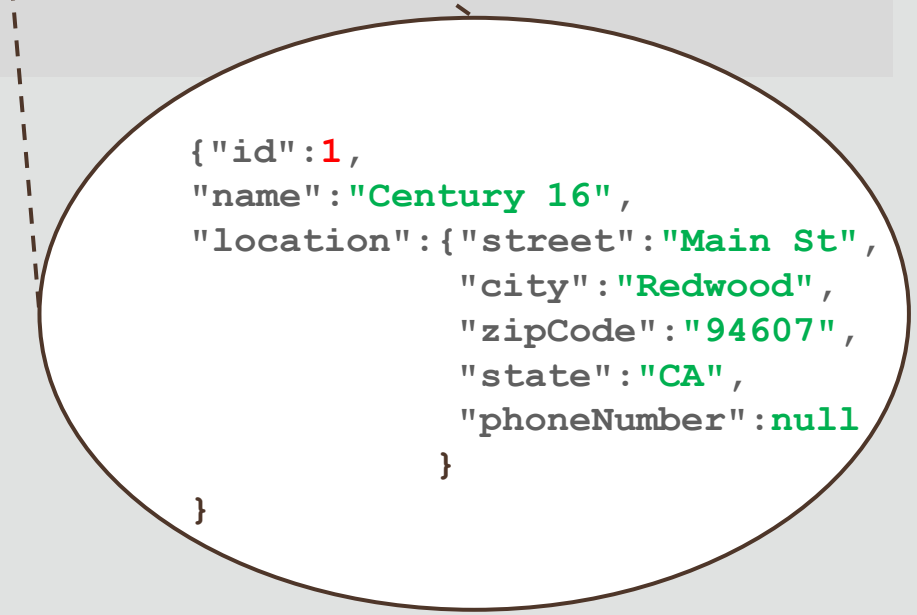
- JSON can be queried using simple SQL dot notation, requires **`IS JSON`** **check constraint** and **table alias**

```
SELECT
   t.json_document.location.city
FROM   theater t;

Location
----------------------
Los Angeles
New York
San Francisco
Redwood
```

Alternative Mechanisms for Querying JSON

Table containing JSON documents

```
SQL> desc THEATER
NAME                          TYPE
----------------------------  -----------
JSON_DOCUMENT                 BLOB
```



- Without **IS JSON** constraint you need to use the **JSON_VALUE** function to query JSON

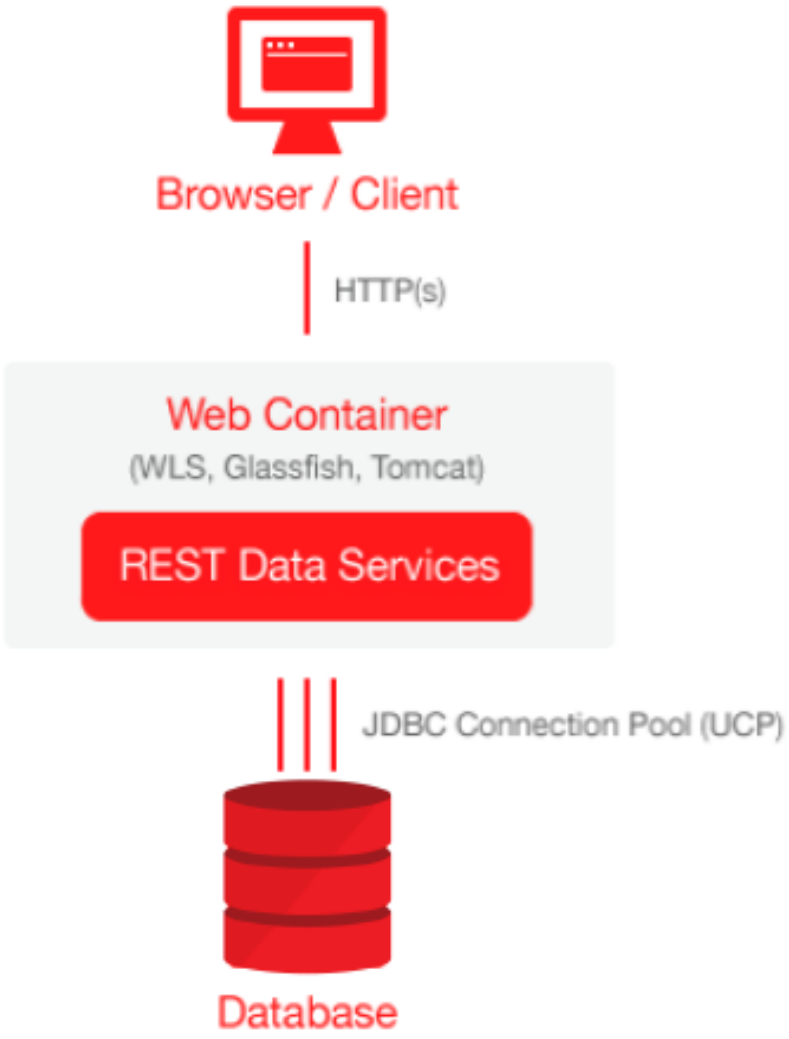
```
SELECT
JSON_VALUE(t.json_document,
                '$.location.city') City
FROM    theater t;

City
--------------------
Los Angeles
New York
San Francisco
Redwood
```

Oracle REST Data Service

Oracle Database Cloud Service

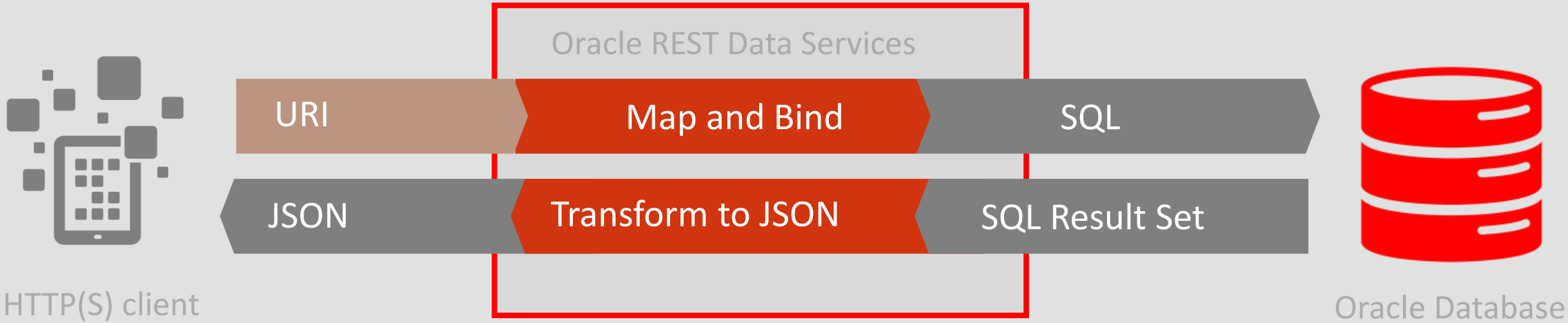
Oracle REST Data Services

- Available Standalone (Jetty), Weblogic, Tomcat & Glassfish
- Turns Database Service into an RESTFul API service
- Fully provisioned and functional in all cloud editions
- Available in 11g, 12c and 18c, no extra cost
- Allows publishing of URI based access to Oracle database over REST
- Results in JSON or CSV
- Mapping of URI to SQL or PL/SQL
- All HTML methods GET, PUT, POST, DELETE, PATCH
- Oauth2 integration
- Highly scalable, can use all feature of database



Oracle REST Data Services

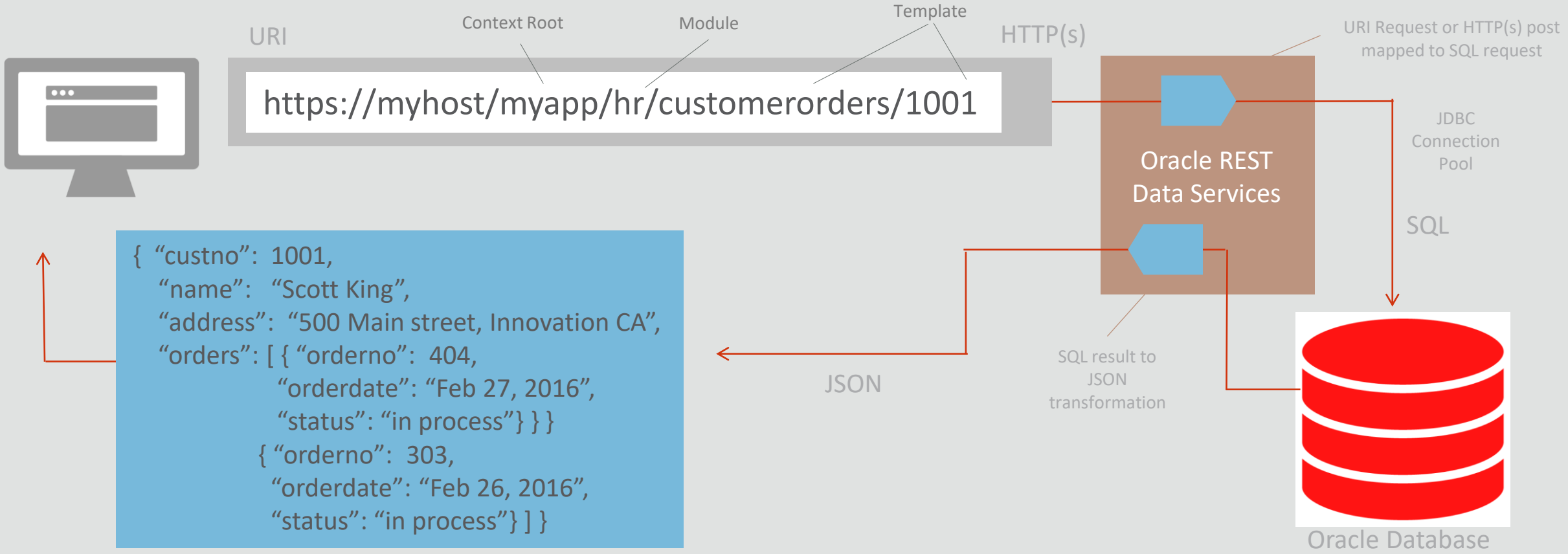
Serving JSON results from relational data



- Data stored in standard relational tables and columns
- Oracle REST Data Services (ORDS) Developer defines URI<>SQL mapping
- App Developer calls named URI over HTTP(S) gets and posts

Oracle REST Data Services

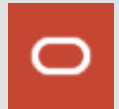
HTTP(s) API App-Dev with Relational Tables in Oracle Database



ORDS maps standard URI requests to corresponding relational SQL (not schemaless): e.g. SQL SELECT from customers and orders table.
ORDS also transforms the SQL results into JavaScript Object Notation (JSON), other formats include HTML, binary and CSV.
Fully committed to supporting any and all standards required by Fusion / SaaS / FMW; we are actively engaged in the ongoing dialog.

SODA: Simple Oracle Document Access

- A simple NoSQL-style API for Oracle
  - Collection Management: Create and drop collections
  - Document Management: CRUD (Create, Retrieve, Update and Delete) operations
  - List and Search: (Query-by-Example) operations on collections
  - Utility and Control: Bulk Insert, index management
- Developers can work with Oracle without learning SQL or requiring DBA support
  - Same development experience as pure-play document stores



Oracle Database as a Document Store

Flexible Schema development

JSON

Via RESTful service

JSON

Via Native APIs

Aplications developed
using SODA APIs

Data persisted in database
In JSON
Using Oracle Database

SQL

Data analyzed via SQL
on JSON Documents

SODA for REST

```
curl --digest -u SCOTT:tiger -X POST  -H "Accept: application/json" -H "Content-type:
application/json" --upload-file po.json http://localhost:8080/DBJSON/SCOTT/MyCollection
```


```
{
      "items": [
                {
                          "id": "A450557094D04957B36346F630CDDF9A",
                          "etag": "C1354F27A5180FF7B828F01CBBC84022DCF5F7209DBF0E6DFFCC626E3B0400C3",
                          "lastModified": "2015-02-09T01:03:48.291462",
                          "created": "2015-02-09T01:03:48.291462"
                }
      ],
      "hasMore": false,
      "count": 1
}
```

SODA for REST

```
curl --digest -X GET    --write-out "%{http code}\n" -u SCOTT:tiger
http://localhost:8080/DBJSON/SCOTT/Collection1/14E6656206114A12950ABF745C309CC5
```


```
{
      "PONumber": 14,
      "Reference": "SVOLLMAN-20140525",
      "Requestor": "Shanta Vollman",
      "User": "SVOLLMAN",
      "CostCenter": "A50",
      "ShippingInstructions": {
                "name": "Shanta Vollman",
                "Address": {
                          "street": "200 Sporting Green",
                          "city": "South San Francisco",
                          "state": "CA",
                          "zipCode": 99236,
                          "country": "United States of America"
                },
                "Phone": [
                          {
                                    "type": "Office",
                                    "number": "823-555-9969"
                          }
                          ]
      },
      "Special Instructions": "Counter to Counter",
      "LineItems": […]
}
```

Workshop

MultiTenant (Advanced) + JSON

https://github.com/OracleDataManagementSpain/ConvergedDatabase/tree/master/MTJSONIMACO

18 Enero
WORKSHOP_Multitenant_Multimodel_InMemory_HOL_v3_part1of3.pdf

19 Enero
WORKSHOP_Multitenant_Multimodel_InMemory_HOL_v3_part2of3.pdf

20 Enero
WORKSHOP_Multitenant_Multimodel_InMemory_HOL_v3_part3of3.pdf

ORACLE

Oracle Database MultiTenant/MultiModel/InMemory/ACO

Workshop

Juan Carlos Diaz
José Vázquez
Francisco Alvarez
Francisco Rivas

Madrid 20 de Enero de 2022

Overview

- Multitenant
- MultiModel - JSON
- Oracle In-Memory Database
- ACO

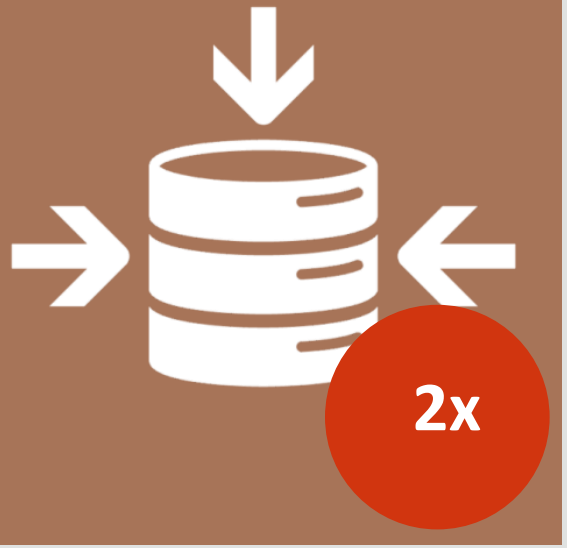
Overview

- MultiTenant
- MultiModel - JSON
- Oracle Database In-Memory

Oracle Database In-Memory Goals

Real Time Analytics

Accelerate Mixed Workload OLTP



No Changes to Applications

Trivial to Implement

Row Format Databases vs. Column Format Databases

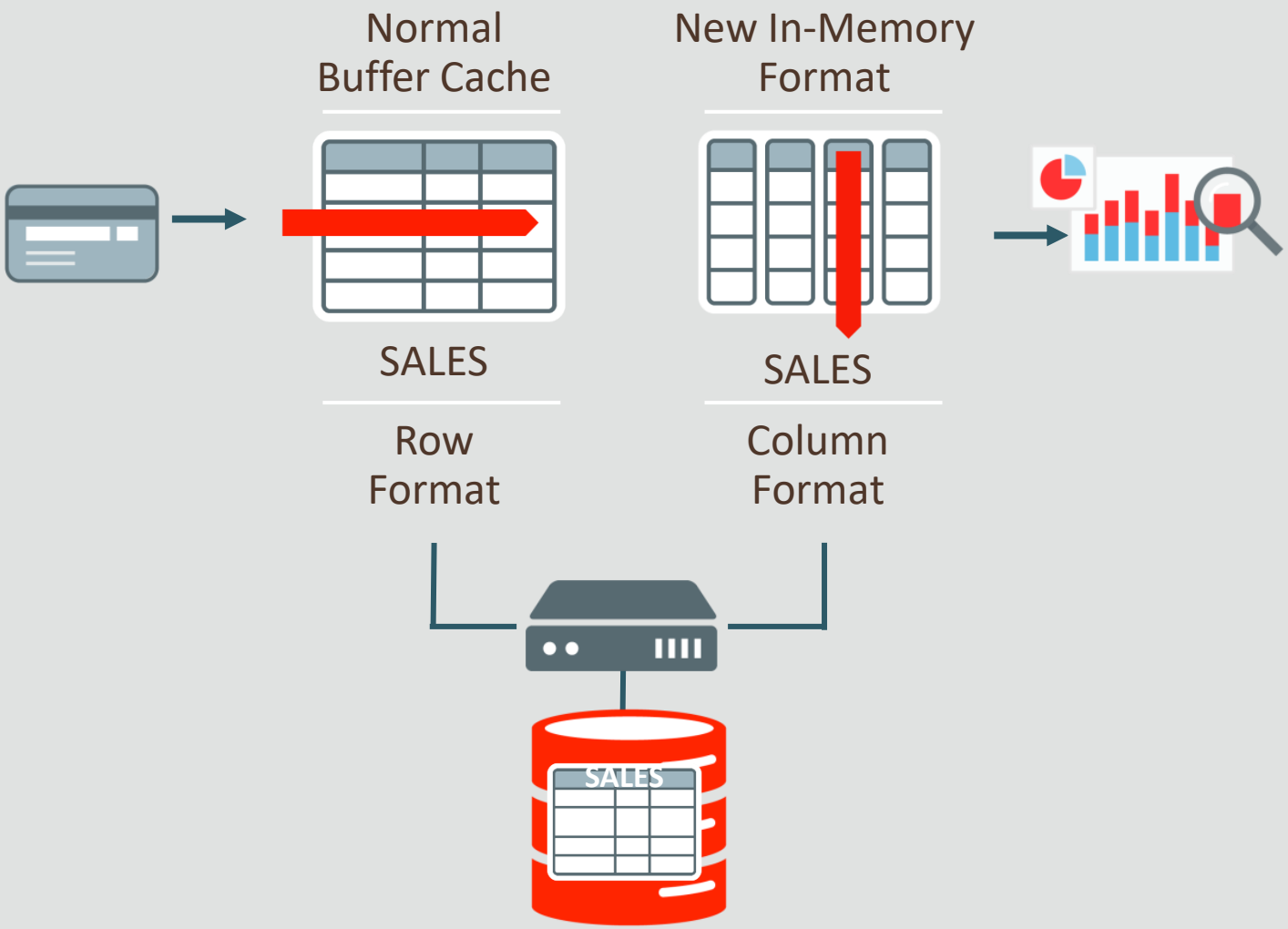
Row

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

Column

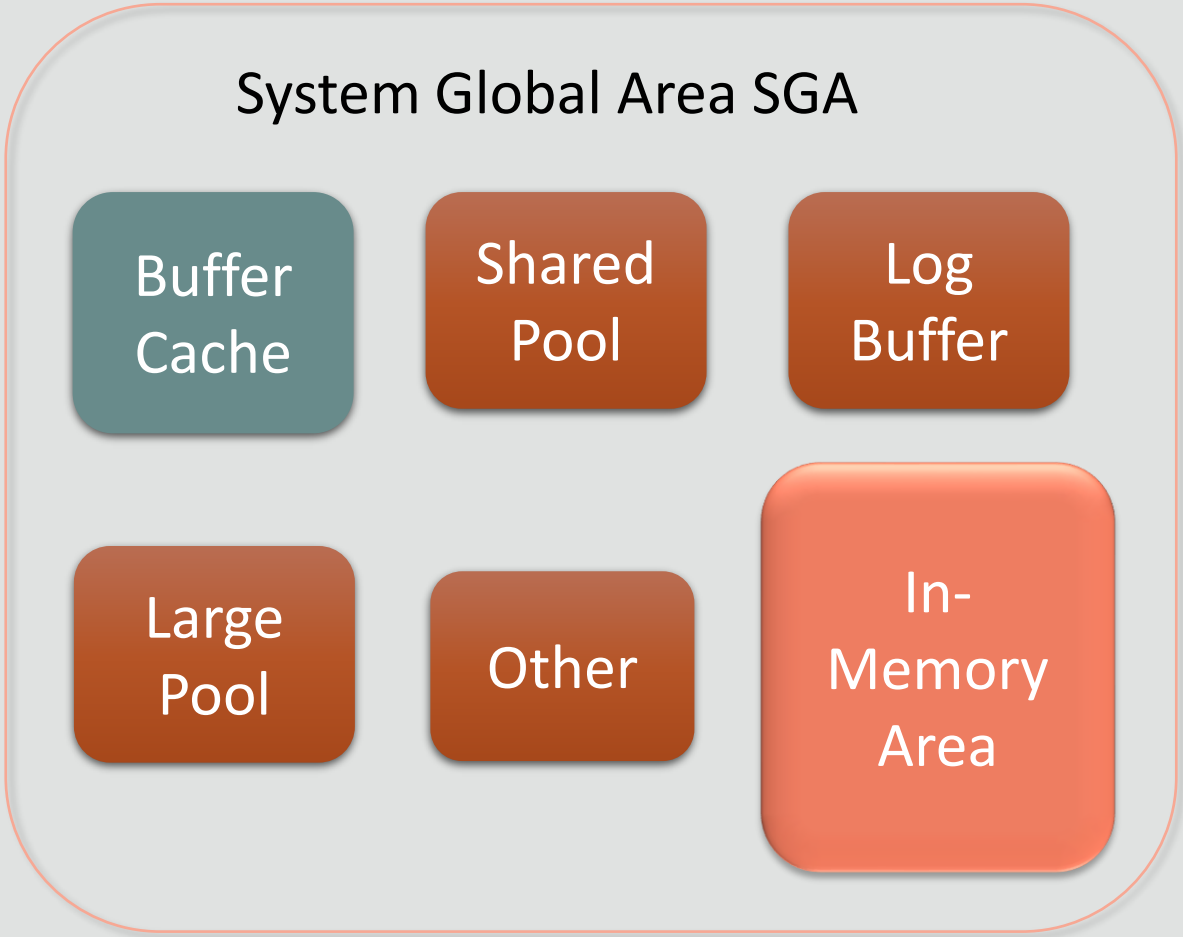
- **Analytics** run faster on column format
  - Example : Report on sales totals by region
  - Fast accessing few columns, many rows

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

In-Memory Area: Static Area within SGA



- Contains data in the new In-Memory Columnar Format
- Controlled by INMEMORY_SIZE parameter
  - Minimum size of 100MB
- Can be re-sized larger while database is running (12.2)
- SGA_TARGET must be large enough to accommodate In-Memory area

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

Populating : In-Memory Column Store

```
ALTER TABLE sales INMEMORY;

ALTER TABLE sales NO INMEMORY;


CREATE TABLE customers ……
PARTITION BY LIST
   (PARTITION p1 …… INMEMORY,
   (PARTITION p2 …… NO INMEMORY);
```

- New INMEMORY ATTRIBUTE
- Following segment types are eligible
  - Tables
  - Partitions
  - Subpartition
  - Materialized views
- Following segment types not eligible
  - IOTs
  - Hash clusters
  - Out of line LOBs

Pure OLTP Features

Populating : In-Memory Column Store

```
ALTER TABLE sales INMEMORY
NO INMEMORY (PROD_ID);
```

```
CREATE TABLE orders
   (c1 number,
    c2 varchar(20),
    c3 number)
INMEMORY PRIORITY CRITICAL
NO INMEMORY (c1);
```

- Possible to populate only certain columns from a table or partition
- Order in which objects are populated controlled by PRIORITY subclause
  - Critical, high, medium, low
  - Default – none (populate on first access)
  - Does not control the speed of population

Populating : In-Memory Column Store

```
ALTER MATERIALIZED VIEW mv1
INMEMORY MEMCOMPRESS FOR QUERY;
```

```
CREATE TABLE trades
   (Name varchar(20),
    Desc varchar(200))
INMEMORY
MEMCOMPRESS FOR DML(desc);
```

- Objects compressed during population
- New compression techniques
  - Focused on scan performance
- Controlled by MEMCOMPRESS subclause
- Multiple levels of compression
- Possible to use a different level for different partitions in a table

Populating : In-Memory Column Store

```
CREATE TABLE ORDERS ……
PARTITION BY RANGE ……
   (PARTITION p1 ……
    INMEMORY NO MEMCOMPRESS
    PARTITION p2 ……
    INMEMORY MEMCOMPRESS FOR DML,
    PARTITION p3 ……
    INMEMORY MEMCOMPRESS FOR QUERY,
    :
   PARTITION p200 ……
    INMEMORY MEMCOMPRESS FOR CAPACITY
   );
```

- Different levels
  - FOR DML
    Use on tables or partitions with very active DML activity
  - FOR QUERY
    Default mode for most tables
  - FOR CAPACITY
    For less frequently accessed segments
- Easy to switch levels as part of ILM strategy

Oracle Compression Advisor And In-Memory

```
DECLARE
  l_blkcnt_cmp       BINARY_INTEGER;
  l_blkcnt_uncmp     BINARY_INTEGER;
  l_row_cmp          BINARY_INTEGER;
  l_row_uncmp        BINARY_INTEGER;
  l_cmp_ratio        NUMBER;
  l_comptype_str     VARCHAR2(100);
BEGIN
     dbms_compression.get_compression_ratio(
   -- input parameters
       scratchtbsname   => 'USERS',                    -- scratch tablespace
       ownname          => 'SSB',                       -- owner of the table
       objname          => 'LINEORDER',                 -- table name
       subobjname       => NULL,                        -- partition name
       comptype         => DBMS_COMPRESSION.COMP_INMEMORY_QUERY, -- compression algorithm
   -- output parameters
       blkcnt_cmp       => l_blkcnt_cmp,                -- number of compressed blocks
       blkcnt_uncmp     => l_blkcnt_uncmp,              -- number of uncompressed blocks
       row_cmp          => l_row_cmp,                   -- number of rows in a compressed block
       row_uncmp        => l_row_uncmp,                 -- number of rows in an uncompressed block
       cmp_ratio        => l_cmp_ratio,                 -- compression ratio
       comptype_str     => l_comptype_str               -- compression type
     );
     dbms_output.put_line('LINEORDER '||l_comptype_str||' ratio: '||to_char(l_cmp_ratio,'99.999'));
END;
```

- Easy way to determine memory requirements
- Use DBMS_COMPRESSION
- Applies MEMCOMPRESS to sample set of data from a table
- Returns estimated compression ratio

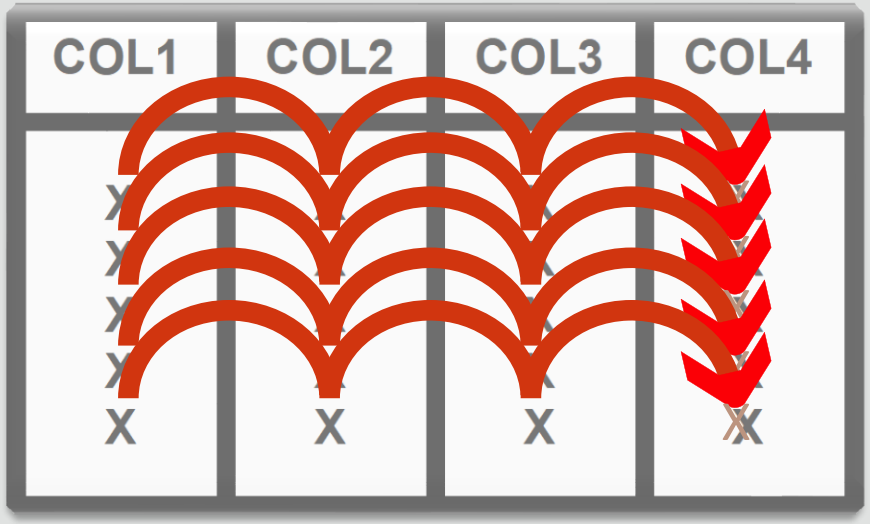
Oracle In-Memory Advisor

Object Type	Object	Estimated In-Memory Size	Analytics Processing Seconds	Estimated Reduced Analytics Processing Seconds	Estimated Analytics Processing Performance Improvement Factor	Benefit / Cost Ratio (Improvement Factor / In-Memory Size)
Table	SOE.LOGON	451.76MB	2114	1,887	9.3X	20.586
Table	SOE.CARD_DETAILS	607.32MB	8346	7,248	7.6X	12.514
Table	SOE.ADDRESSES	1.09GB	5237	4,621	8.5X	7.798
Partition	SOE.PRODUCT_MOCKUP.Y2014Q1	812.6MB	2003	1,489	3.9X	4.799
Table	SOE.CUSTOMERS	1.10GB	108	95	8.2X	7.455
Table	SOE.ORDER_ITEMS	2.19GB	7128	6,393	9.7X	4.429
Table	SOE.ORDERS	1.34GB	3512	2,917	5.9X	4.403
Table	SOE.PRODUCT_INFORMATION	1.78MB	2873	2,205	4.3X	2.416
Partition	SOE.PRODUCT_MOCKUP.Y2013Q4	1.62GB	97	1,489	3.7X	2.284
Partition	SOE.PRODUCT_MOCKUP.Y2014Q2	3.37GB	642	493	4.3X	1.276

- New In-Memory Advisor
- Analyzes existing DB workload via AWR & ASH repositories
- Provides list of objects that would benefit most from being populated into IM column store

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

Buffer Cache



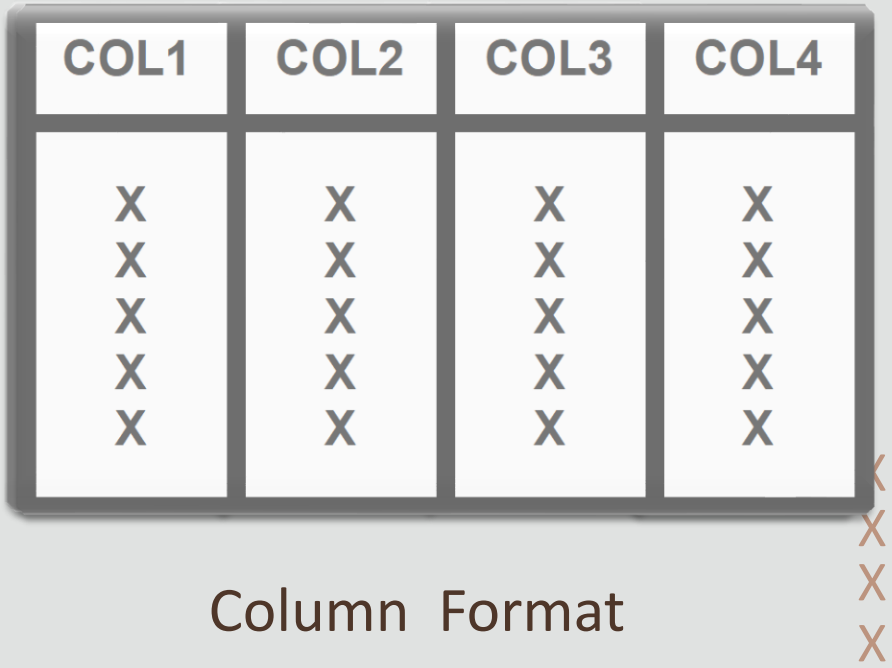
Row Format

```
SELECT COL4 FROM MYTABLE;
```

RESULT

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

IM Column Store

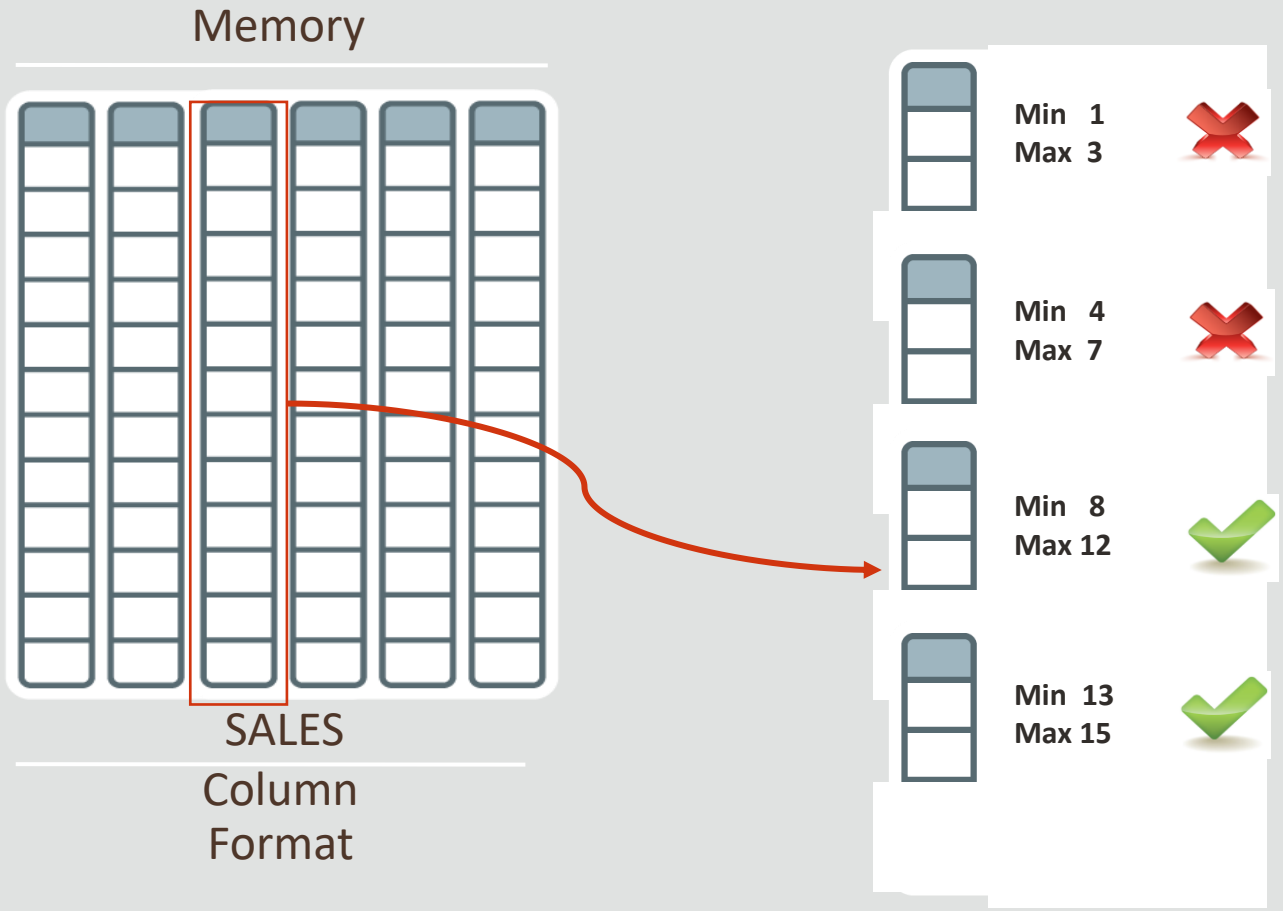



```
SELECT COL4 FROM MYTABLE;
```

RESULT

Oracle In-Memory Column Store Storage Index

Example: Find sales from stores with a store_id of 8 or higher



- Each column is the 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

SQL statement is scanning data in the IM ?

```
SELECT Max(lo_ordtotalprice) most_expensive_order
FROM lineorder
WHERE  lo_shipmode = 5;
```

```
-------------------------------------------------------------
| Id  | Operation                          | Name           |
-------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                |
|   1 |  SORT AGGREGATE                    |                |
|   2 |   TABLE ACCESS IN MEMORY FULL      | LINEORDER      |
-------------------------------------------------------------
```

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

ORACLE E-BUSINESS SUITE

ORACLE FUSION APPLICATIONS

ORACLE JD EDWARDS

ORACLE PEOPLESOFT

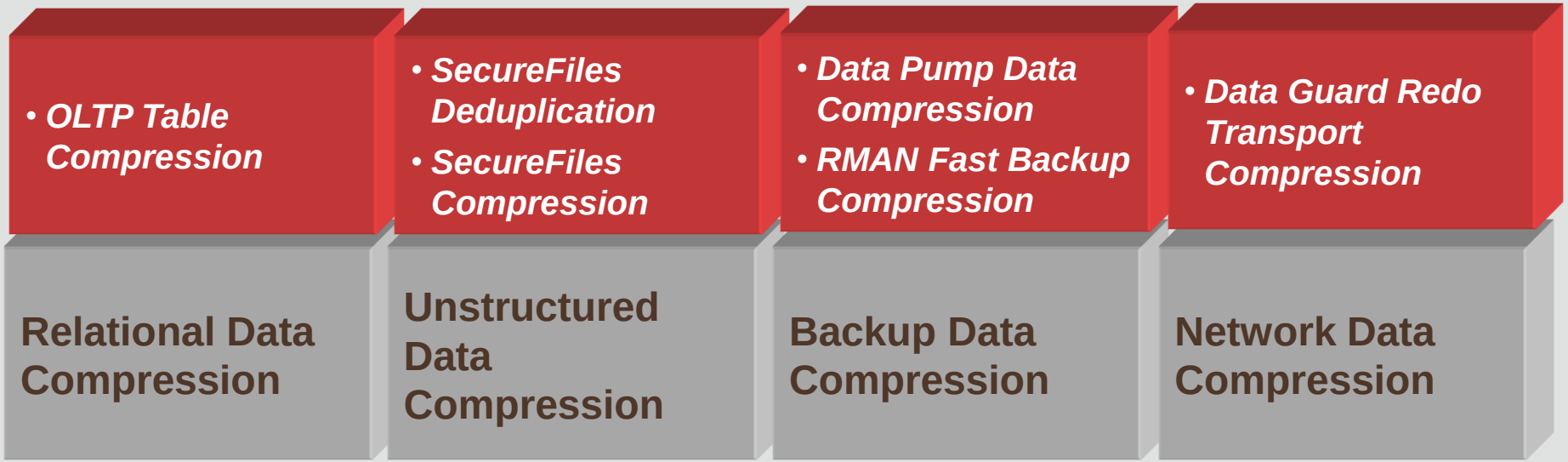
ORACLE SIEBEL

Uniquely Achieves All In-Memory Benefits With No Application Changes

Overview

- MultiTenant
- JSON
- InMemory
- ACO

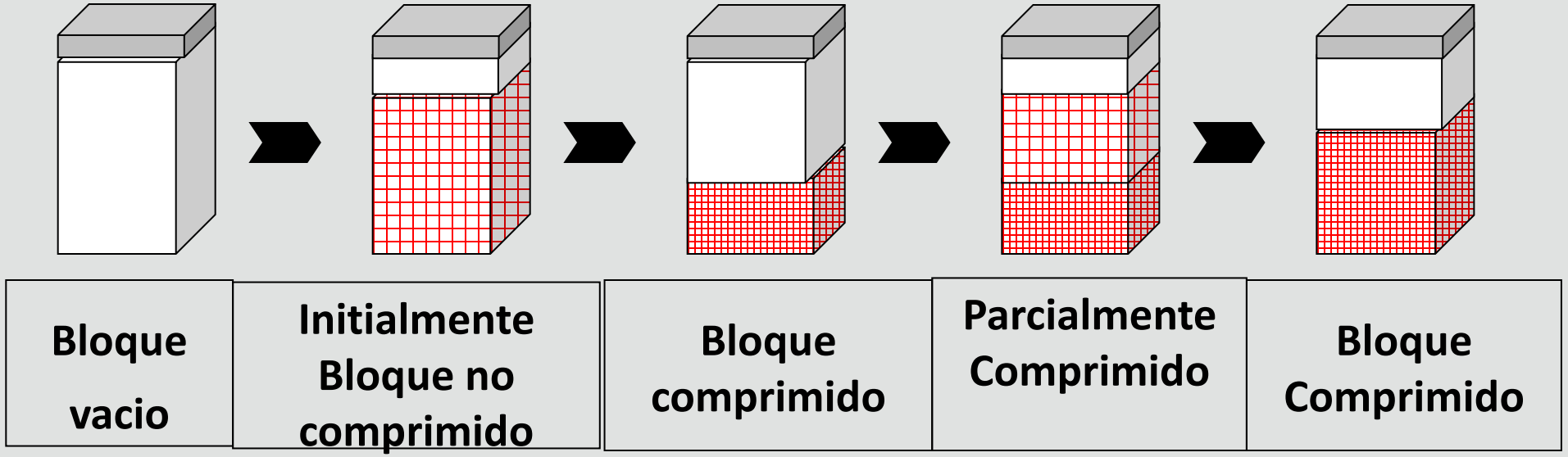
Oracle Advanced Compression Option



Reduce requerimientos de recursos y costes

- Almacenamiento
- Ancho de banda
- Uso de memoria

OLTP Proceso compresión de tabla

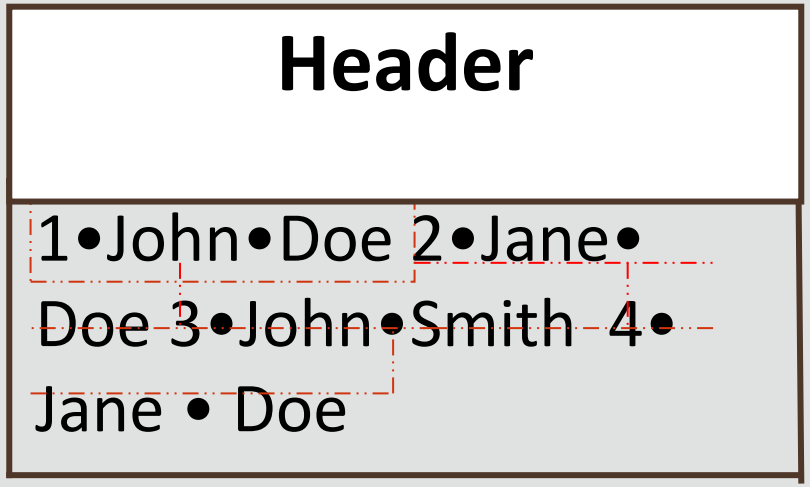


OLTP Compresión Tabla

Employee Table

ID	FIRST_NAME	LAST_NAME
1	John	Doe
2	Jane	Doe
3	John	Smith
4	Jane	Doe

Initially Uncompressed Block




```
INSERT INTO EMPLOYEE
    VALUES (5, 'Jack', 'Smith');
COMMIT;
```

OLTP Compresión Tabla

Employee Table

ID	FIRST_NAME	LAST_NAME
1	John	Doe
2	Jane	Doe
3	John	Smith
4	Jane	Doe
5	Jack	Sr

Local Symbol Table

Compressed Block Header

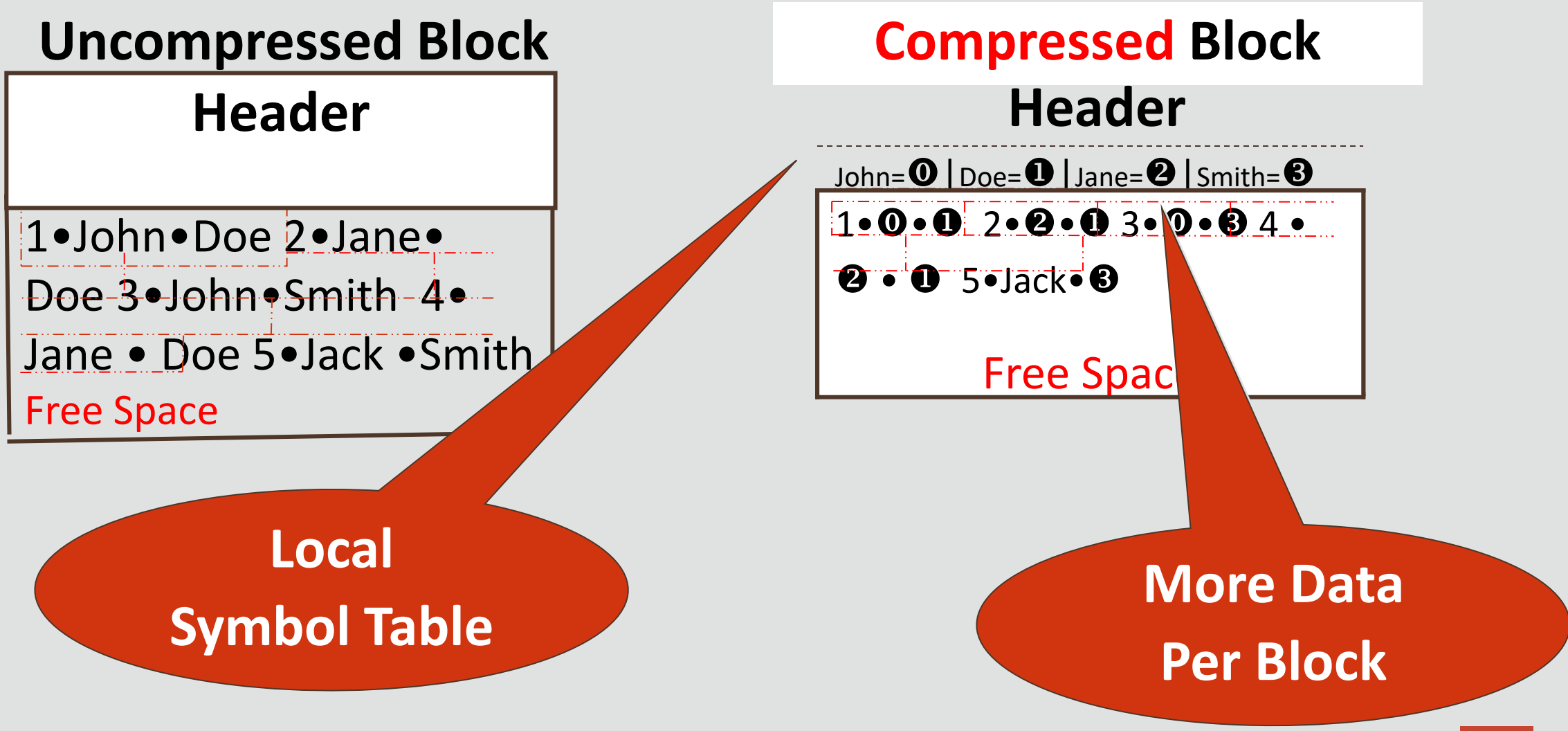
John=❶ | Doe=❶ | Jane=❷ | Smith=❸

1•❶•❶ 2•❷•❶ 3•❶•❸ 4•

❷•❶ 5•Jack•❸

Free Space

OLTP Table Compression

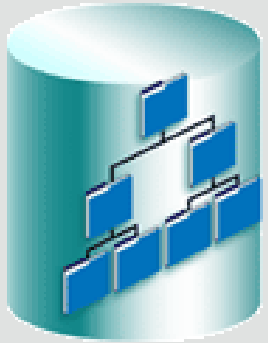


19c Oracle SecureFiles

Gestión de datos consolidada y segura

- **SecureFiles** información no estructurada o semi estructurada almacenada en la BBDD.

- **Reingenieria de los LOBs**. Acceso más rápido y con mas funcionalidad.
  - Eliminación de elementos duplicados de manera transparente (deduplication), compresión y encriptación.
  - Utiliza las funcionalidades de seguridad, disponibilidad,
  - y escalabilidad de la base de datos.
  - Migración sencilla de LOBs.



Compresión Data Pump

- Oracle Database 12*c* extiende la compresión de la tabla durante los exports
  - No necesidad de descomprimir antes de importar
  - `COMPRESSION={ALL | DATA_ONLY | [METADATA_ONLY] | NONE}`
- Simple compresión de ambos data y metadata
  - Datos comprimidos directamente a disco resultando una disminución de espacio

Compresión Backup

- RMAN Backup Compression
  - Backup es comprimido antes de escribirse a disco
  - No necesario descomprimir antes de restauración
  - Compresion nivel LOW Algoritmo de compresión más rápido
    - Más adecuado cuando hay limites en CPU
  - Nivel de compresión MEDIUM
    - Balancea entre uso de CPU y ratio de compresión
  - Compresión LEVEL HIGH Mejor ratio de compresión y más alto utilización de CPU
    - Más adecuado cuando hay límites de red o I/O

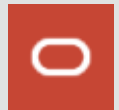
Compresión de Red

Data Guard Redo Transport Services

- Data Guard Redo Transport Services transfiere datos de log a base standby
- Con Advanced Compression, redo data pueden ser transmitidos en formato comprimido para reducir consumo de ancho banda y tiempo de transmisión
- 12c synchronous redo transport (SYNC) o asynchronous redo transport (ASYNC).
- Ejemplo:
  - `LOG_ARCHIVE_DEST_3='SERVICE=denver SYNC COMPRESSION=ENABLE|[DISABLE]'`

Workshop

IN-Memory Database + Advanced Compression Option



https://github.com/OracleDataManagementSpain/ConvergedDatabase/tree/master/MTJSONIMACO

18 Enero
WORKSHOP_Multitenant_Multimodel_InMemory_HOL_v3_part1of3.pdf

19 Enero
WORKSHOP_Multitenant_Multimodel_InMemory_HOL_v3_part2of3.pdf

20 Enero
WORKSHOP_Multitenant_Multimodel_InMemory_HOL_v3_part3of3.pdf

ENCUESTA

ORACLE

Workshop Virtual BD Convergente:
Multitenant, Multimodel, In-memory

¡Tu opinión es muy importante!

Escanea el QR para responder o entra aquí:

https://bit.ly/3AbzRot

Thank you

Oracle Spain