ORACLE

# Oracle Converged Database

## 19c/21c/23ai Overview

**Valentin Leonard Tabacaru**

Oracle Database Product Management
June 2025

ORACLE Database LODCoreMadrid uc3m

# Inscríbete al Datathon y consigue tu OCI gratis

## Escucha a los expertos de Oracle para ayudarte con el reto

- Intro a la tecnología y a la nube de Oracle
- Herramientas de Low Code
- Bases de datos convergentes
- Asistentes de IA generativa

</>

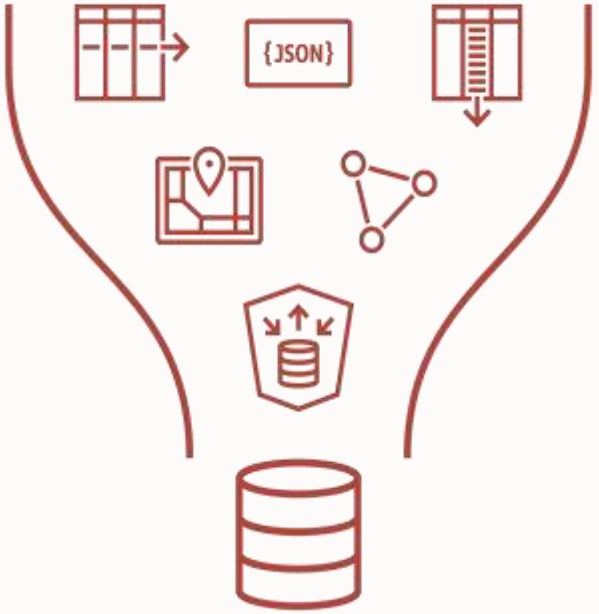
bit.ly/inscribete_datathon

## Únete a Discord

apex.oracle.com/go/db/datathon_oracledb_discord

# Oracle Database 19c/21c/23ai

A Converged, Open SQL Database




**Converged Database**

## Multi-model

Best-of-Breed Relational, JSON, Spatial, Graph, Cube, Text, Blockchain
Cross-model operations enable you to easily create value across all your data

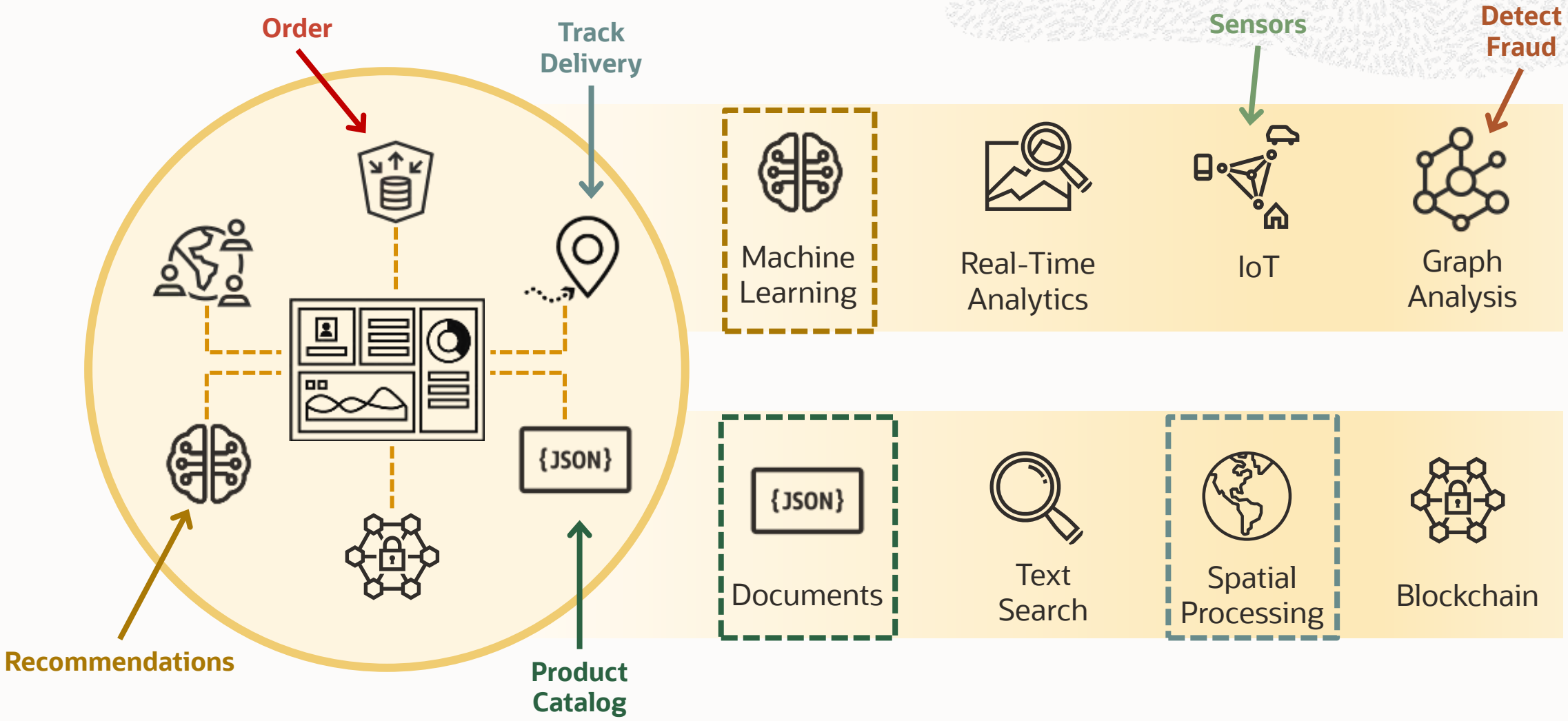
## Multi-workload

High Performance Transactions, DW, Analytics, ML, IoT, Streaming, Multitenant
Deep optimizations deliver exceptional price-performance across all workloads

## Productivity for developers and analysts

Same SQL and transactions operate on any data and workload
Integrated microservices, events, REST, CI/CD, Low-code

# Oracle Database 19c/21c/23ai Supports Data Driven Apps



# Oracle Database 19c/21c/23ai New Features

**Application Development**

**Big Data & Data Warehousing**

**Management**

**Performance & High Availability**

**Security**

# Native JSON Type

- JSON data type provides efficient storage of JSON data and improved performance of Java applications
- Use classes and interfaces of the JDBC API `oracle.sql.json` package to work with JSON data type
- Oracle Data Pump and SQL*Loader support the JSON data type

```
CREATE TABLE j_order
(
   id      INTEGER PRIMARY KEY,
   po_doc JSON
);
```

# JSON Document Relational Duality

Data is stored as rows in tables to provide the benefits of the relational model and SQL access

Data can be accessed as JSON documents to deliver the application simplicity of documents

Duality view is a database object like table or view

- No separate mapping language or tools, no programming, no deploying, no configuring
- Document-level consistency, and table row-level consistency, are guaranteed together

JSON Duality Views declare the recipe for assembling normalized rows into a JSON document

```
-- JSON Duality View from relational
tables

CREATE OR REPLACE JSON DUALITY VIEW FROM
Student AS
student_schedule
{   name:         sname
    student_id:  stuid
    schedule:    student_courses
      { course: course
        {     time
              course:     cname
              course_id: cid
              room
              teacher:    teacher
               {   teacher:     tname
                   teacher_id: tid
      }   }     }
};
```

# SQL Macros

19c

- Simpler way to encapsulate complex processing logic directly within SQL
- Factor out common SQL expressions and statements into reusable, parameterized constructs
- Increase developer productivity, simplify collaborative development, and improve code quality

```
CREATE or REPLACE FUNCTION
orders_waiting_to_ship(order_value number)
    RETURN varchar2 SQL_MACRO(TABLE)
    IS
BEGIN
    RETURN q'[
    SELECT i.*
    FROM orders o,
         order_items i
    where order_status > 6
      and o.order_total >= order_value
      and o.ORDER_ID = i.ORDER_ID
    ]';
end orders_waiting_to_ship;
```

```
SELECT *
FROM orders_waiting_to_ship(200);
```

**New feature in Oracle 21c – backported to Oracle 19.7**

# JavaScript Stored Procedures

Store and execute JavaScript inside the database

- Allows developers to create stored procedures using JavaScript in the database.
- Allows developers to leverage the huge number of JavaScript libraries.

```
CREATE FUNCTION hello_world RETURN VARCHAR2
AS MLE LANGUAGE JAVASCRIPT
{{
  return 'Hello World!';
}};
/

SELECT hello_world();

HELLO_WORLD
------------
Hello World!
```

- Use JavaScript modules to extend database capabilities

```
CREATE MLE MODULE jsmodule LANGUAGE JAVASCRIPT
AS
 export function concat (str1, str2)
   { return str1 + str2; }
 export function substr(str, start)
   { return str.substring(start); }

CREATE FUNCTION js_contact
 (str1 IN VARCHAR2, str2 IN VARCHAR2)
 RETURN VARCHAR2 AS MLE MODULE jsmodule
 SIGNATURE 'concat(string , string)';

SELECT js_concat('Hello ', 'World!');

JS_CONTACT('HELLO ', 'WORLD!')
------------------------------
Hello World!
```

# Schema Annotations

- It is complex to record and share application metadata
- For example:
  - Mark data as sensitive (GDPR, PCI, etc.)
  - Indicate which columns should be hidden in UI
- This is now simple using Annotations inside the database
- Annotations can be defined on tables, views, MVs, columns, indexes, and domains during creation or via an `ALTER` statement

```
-- Annotate the table and columns
CREATE TABLE emp (
  id number
      ANNOTATIONS(Identity, Display 'emp id'),
  ename varchar2(50)
      ANNOTATIONS(Display 'emp name'),
  sal number
      ANNOTATIONS(Display 'salary', UI_hidden)
  )
    ANNOTATIONS
      (Display 'Emp Table', App_Version '2.0');

-- Find annotations usage for Emp table and
its columns
SELECT *
  FROM user_annotations_usage
WHERE Object_Name = 'EMP'
  AND Object_Type = 'TABLE'
  AND Column_Name IS NOT NULL;
```

# Data Use Case Domains

- It is complex to write applications that use user-defined types (object types) in the database
- Converting app-tier language types to database user-defined types makes programs more complex
- Now Extended ISO SQL Domains provide a simple alternative for common cases
- Specify the intended usage of data
- Like the “format” annotation in OpenAPI
- Examples: credit card, password, email, telephone, URL, quantity, etc.

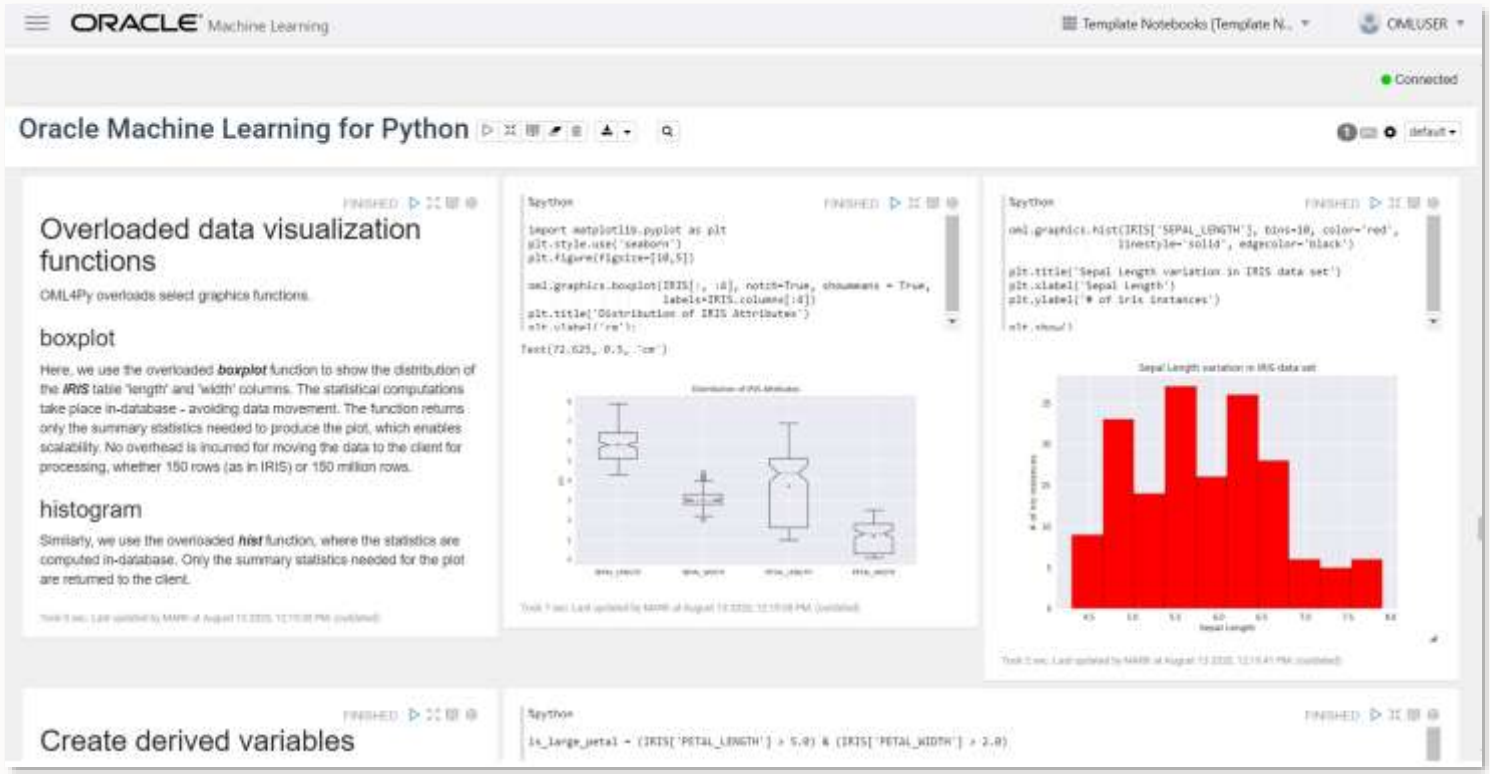
```
--Create a new domain to identify email
addresses

CREATE DOMAIN Email AS VARCHAR2(30)
DEFAULT ON NULL t_seq.NEXTVAL || '@oracle.com'
CONSTRAINT EMAIL_C CHECK
 (REGEXP_LIKE (email, '^(\S+)\@(\S+)\.(\S+)$'))
DISPLAY '---' ||
   SUBSTR(Email, INSTR(Email, '@') + 1);

--Create a new domain to identify credit cards

CREATE DOMAIN credit_card AS varchar2(16);

-- Use the new domain when creating a contacts
table
CREATE TABLE customers(
              c_name varchar2(50),
              cr_card_no DOMAIN credit_card,
              … );
```

# Oracle Database 19c/21c/23ai New Features

| Application Development | **Big Data & Data Warehousing** | Management | Performance & High Availability | Security |
|---|---|---|---|---|

# Oracle Machine Learning for Python (OML4Py)

19c



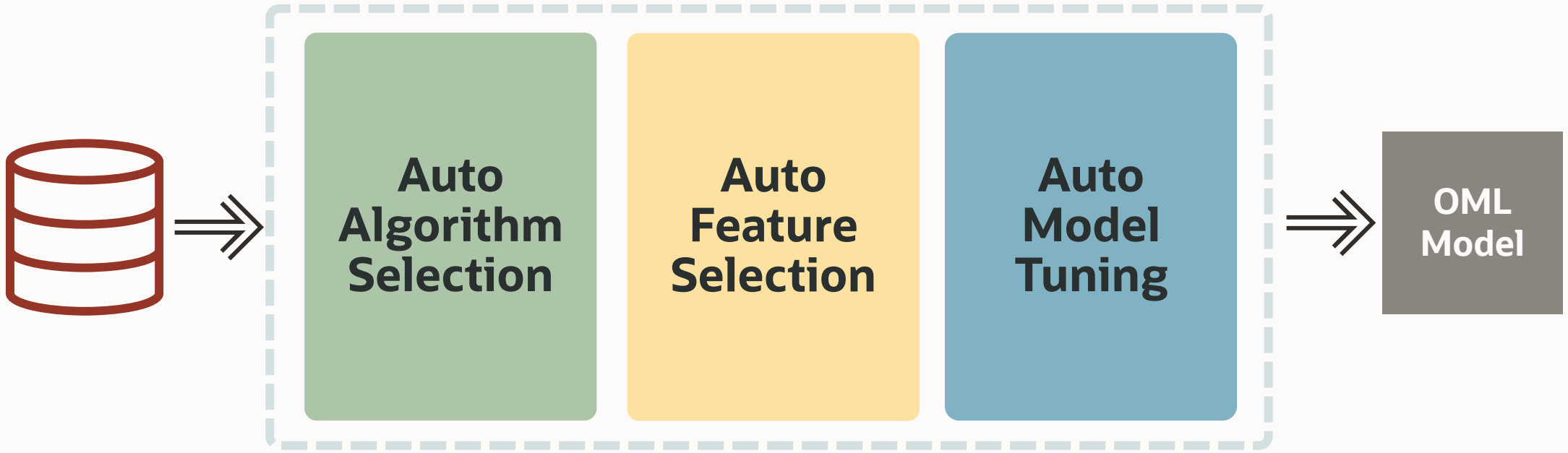

- Enables Python users to manipulate data in database tables and views using Python syntax
- OML4Py functions and methods transparently translate a select set of Python functions into SQL
- Powerful in-database algorithms are made more accessible to Python developers

**New feature in Oracle 21c – backported to Oracle 19.5**

# AutoML in OML4Py

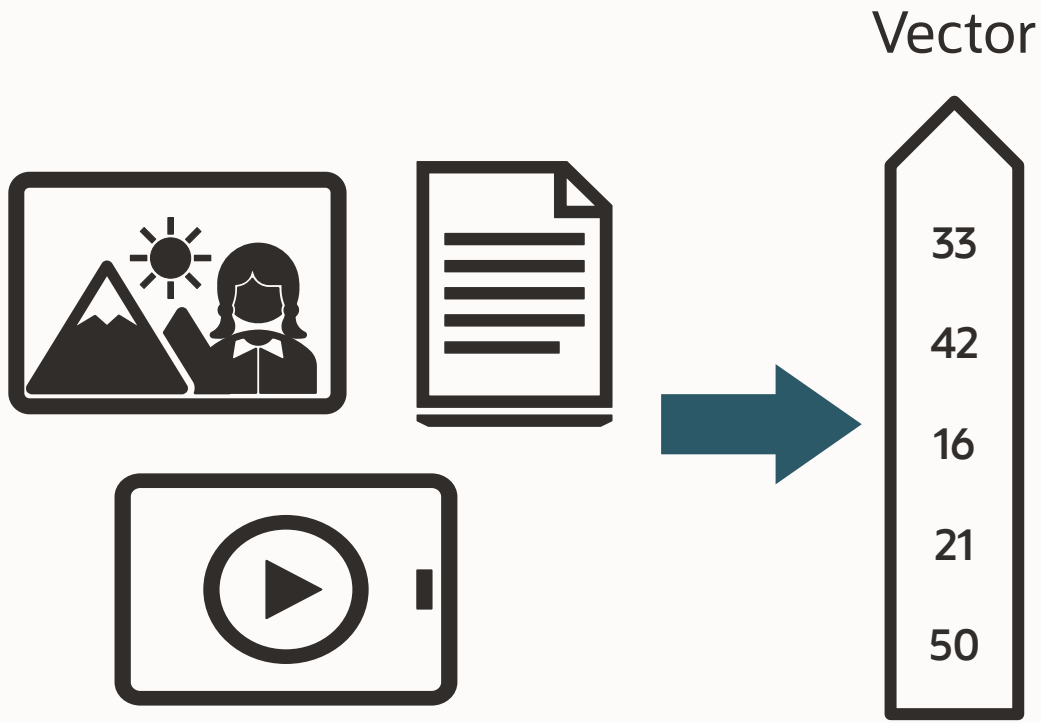
19c

- Eliminate repetitive tasks of model building/evaluation and increase user productivity
- Apply ML to the ML process to reduce algorithm and hyperparameters search space, and reduce compute time and cost
- Enable non-expert users to leverage machine learning



# Vectors are used in AI to encode unstructured data

Such as images, documents, videos, etc.

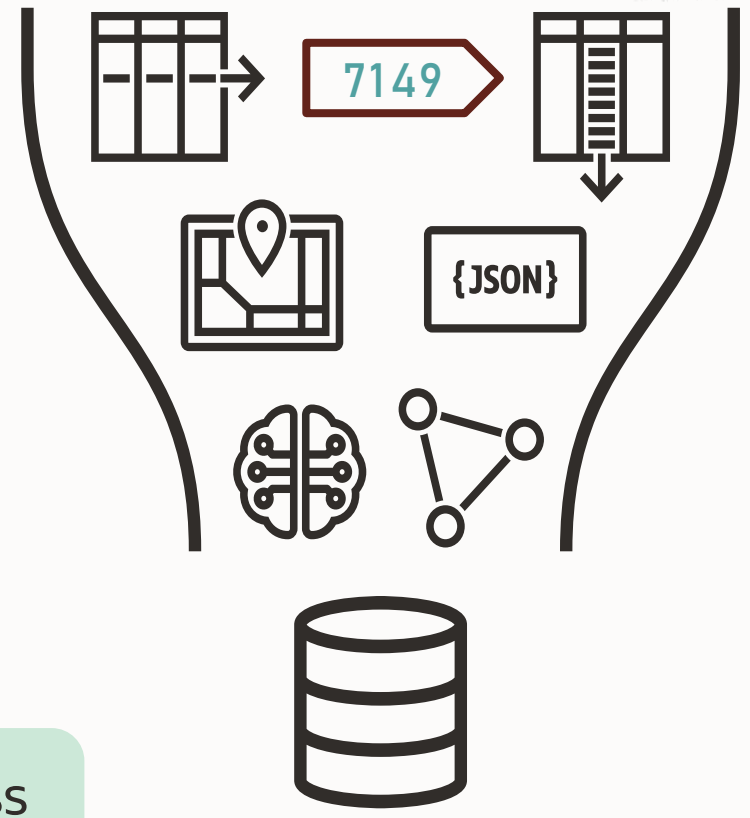



- A vector is a sequence of numbers, called dimensions, representing the important “features” of the data
- Vectors represent the semantic content of data, not the actual words in a document or pixels in an image
- Vectors are produced from unstructured data by AI models such as Neural Networks

# Introducing Oracle AI Vector Search

New set of capabilities in Oracle Database 23ai

- Process vector search and other workloads in the same Oracle converged database
- Designed to be simple to use and easy to understand
- New VECTOR data type for storing vectors
- New SQL syntax and functions express similarity search with ease
- New Approximate search indexes packaged and tuned for high-performance and quality




Converged Database

Perform vector search in queries alongside business data about customers and products

# Allows queries that combine AI Vector Search with business data

About customers and products

- Find houses that are similar to this picture and match the customer's preferred city and budget
- Combines customer data, product data, and AI search in a few lines of SQL!
- A single integrated solution, all data is fully consistent



```
CREATE TABLE house_for_sale (house_id number,
                             price number,
                             city varchar2(400),
                             house_vector vector);


SELECT … FROM house_for_sale
WHERE price <= (SELECT budget FROM customer …)
  AND city in (SELECT search_city FROM customer …)
ORDER BY
  vector_distance(house_vector, :input_vector)
FETCH APPROX FIRST 10 ROWS ONLY
WITH TARGET ACCURACY 95 PERCENT;
```

# Property Graph: Graph Views on Tables

- Improved mapping of relational data to graph model through views
- Better support for dynamic data and less storage consumption

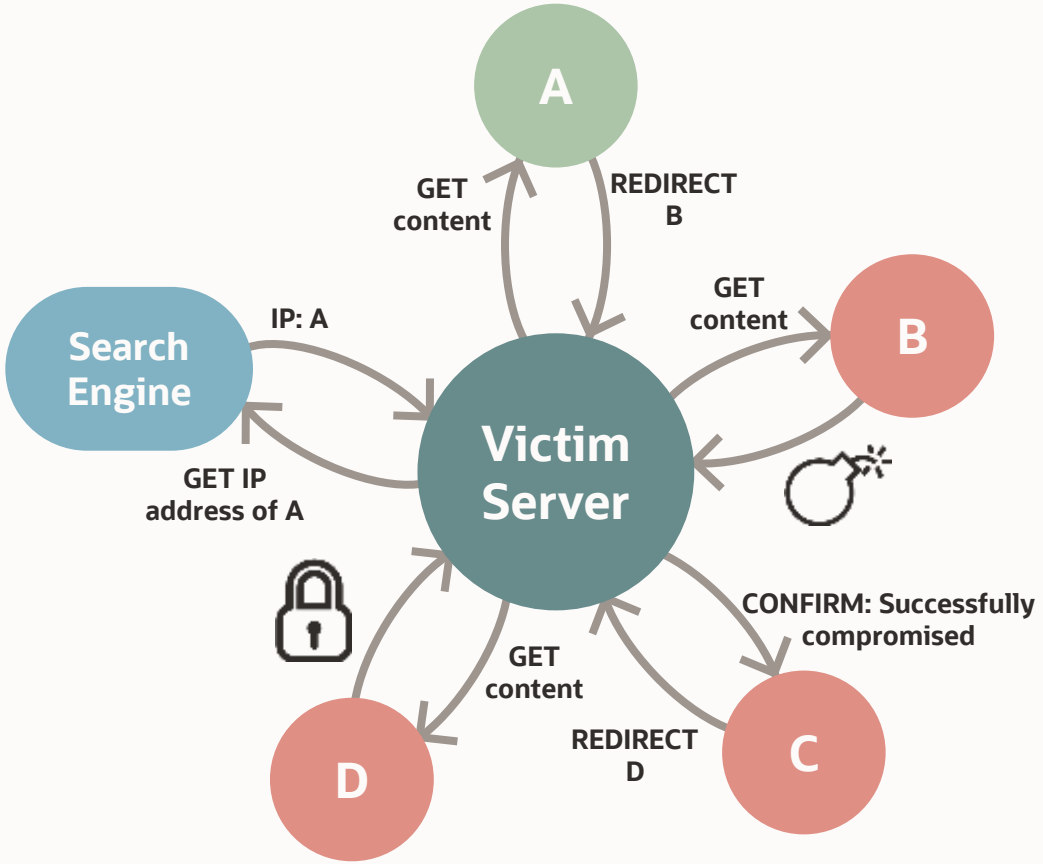
```
CREATE PROPERTY GRAPH sh_purchase
  VERTEX TABLES (
    sh.customers
      PROPERTIES (CUST_ID, CUST_FIRST_NAME),
    sh.products
      PROPERTIES (PROD_ID, PROD_NAME)
  )
  EDGE TABLES (
    sh.sales
      SOURCE customers
      DESTINATION products
      LABEL purchased
      PROPERTIES (QUANTITY_SOLD)
  )
  OPTIONS (PG_VIEW)
```

```
GRANT PGX_SERVER_MANAGE TO GRAPH_ADMINISTRATOR;
GRANT PGX_SESSION_READ_MODEL TO GRAPH_DEVELOPER;
GRANT PGX_SESSION_CREATE TO GRAPH_USER;
```

# Property Graph: Graph Machine Learning Algorithms

- Graph Machine Learning algorithms to create vertex or graph embeddings
- More accurate predictions
- Support for DeepWalk, GraphWise (Graph Neural Network), Pg2vec, and Deep Graph Infomax

## Use GraphML algorithms to detect malware



# Graph traversal queries with SQL/PGQ

23ai


- SQL/PGQ added in SQL:2023 ISO standard
- Create a property graph to define the relationship between rows
- Use the GRAPH_TABLE operator and MATCH clause to traverse the graph

```
CREATE PROPERTY GRAPH BANK_GRAPH
    VERTEX TABLES (
        BANK_ACCOUNTS
        KEY (ID)
        PROPERTIES (ID, Name, Balance)
    )
    EDGE TABLES (
        BANK_TRANSFERS
        KEY (TXN_ID)
        SOURCE KEY (src_acct_id) REFERENCES BANK_ACCOUNTS(ID)
        DESTINATION KEY (dst_acct_id) REFERENCES BANK_ACCOUNTS(ID)
        PROPERTIES (src_acct_id, dst_acct_id, amount)
    );
```

```
-- Find the top 10 accounts by incoming transfers
SELECT acct_id, COUNT(1) AS Num_Transfers
    FROM graph_table ( BANK_GRAPH
        MATCH (src) - [IS BANK_TRANSFERS] -> (dst)
        COLUMNS ( dst.id AS acct_id )
    ) GROUP BY acct_id ORDER BY Num_Transfers DESC
FETCH FIRST 10 ROWS ONLY;
```

# Oracle Database 19c/21c/23ai New Features





| Application Development | Big Data & Data Warehousing | Management | Performance & High Availability | Security |
|---|---|---|---|---|

# Priority Transactions

Priority Transactions **automate the process** of aborting a low-priority transaction that holds a row lock that blocks a high-priority transaction

How it works:

- A new parameter sets the **priority** (HIGH, MEDIUM, LOW) of a user transaction
- Users can configure the **maximum time** a transaction with high priority will wait before aborting a lower-priority transaction holding a row lock


**Benefits**:

- Reduces the administrative burden for DBAs
- Maintains response time and transaction throughput for high priority transactions

# Inter-Instance Resource Management

Server Level Inter-Instance Resource Management enables specifying the minimum CPU resources needed by each CDB and sharing of CPU resources above the minimum

- Integrates with Linux cGroups to implement CPU shares and CPU limits
- Allows a server to be over-subscribed, yet guarantee CPU resources for a particular database instance
- If a database instance does not use its guaranteed CPUs at any moment in time, the other database instances can use it


**Benefits**:

- DBAs can specify the priority of different databases running on the same server
- Better utilization of hardware and reduces the risk associated with server consolidation

# Oracle Database 19c/21c/23ai New Features



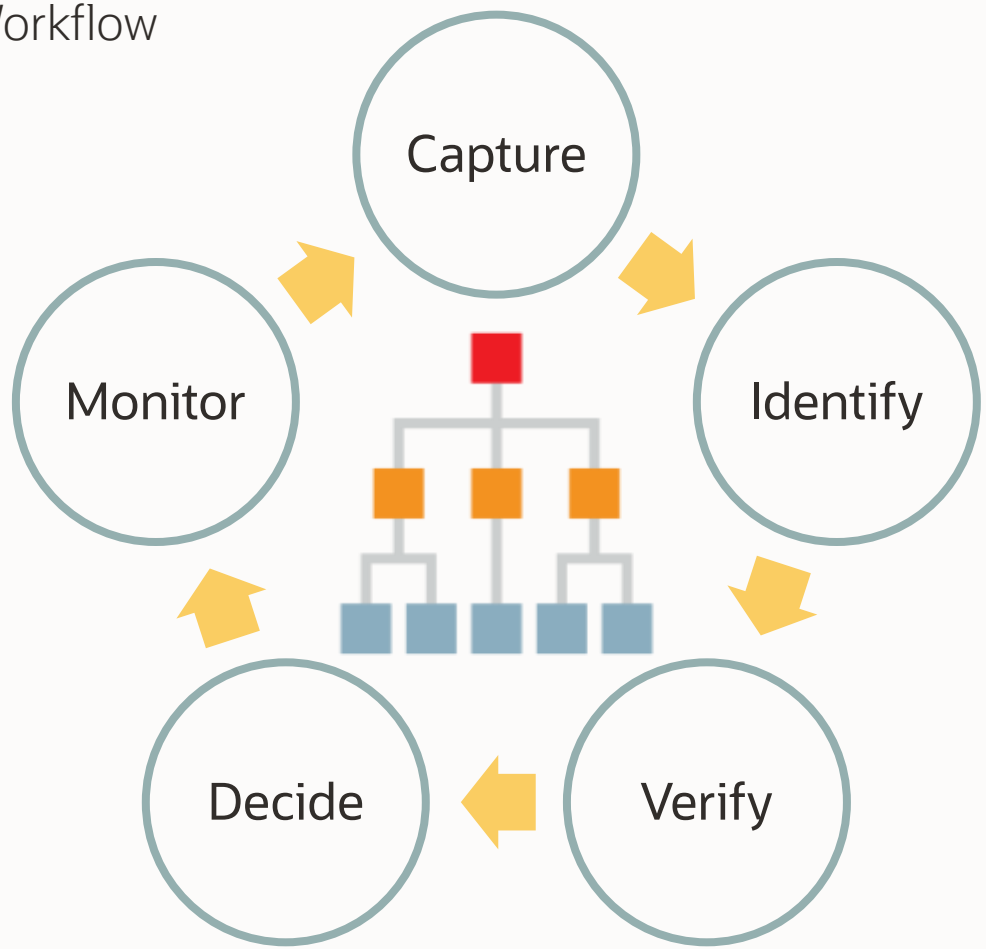


| Application Development | Big Data & Data Warehousing | Management | Performance & High Availability | Security |
|---|---|---|---|---|

# Automatic Indexing

19c

Workflow




The Automatic Indexing methodology is based on a common approach to manual SQL tuning

It identifies candidate indexes and validates them before implementing

The entire process is fully automatic

Transparency is equally important as sophisticated automation

All tuning activities are auditable via reporting

# Automatic Materialized Views

23ai

In Oracle Database 23ai, Automatic Materialized Views' internal algorithms have been enhanced to take advantage of a broader range of features in the database

- Automatic Materialized Views can take advantage of Automatic Partitioning
- Enhanced internal costing model for Automatic Material Views selection, balancing:
  - Access benefits
  - Maintenance (refresh) cost
  - Frequency of executions
  - Broader rewrite capabilities available, including outer-join queries with filter predicates

**Benefits**:

- Automatically balances the trade-off between the operational overhead of maintaining materialized views, and the query performance benefits of materialized views

# Real-time SQL Plan Management

**SQL Plan Management** has been enhanced to detect and prevent SQL performance regressions in real-time instead of as a background process

Prevents performance regressions

- Plan changes are detected at parse-time
- Performance of previous SQL execution plans is compared with the performance of the new plan
- If a previous plan out-performs the new plan, the previous plan will be enforced in future executions of the SQL statement

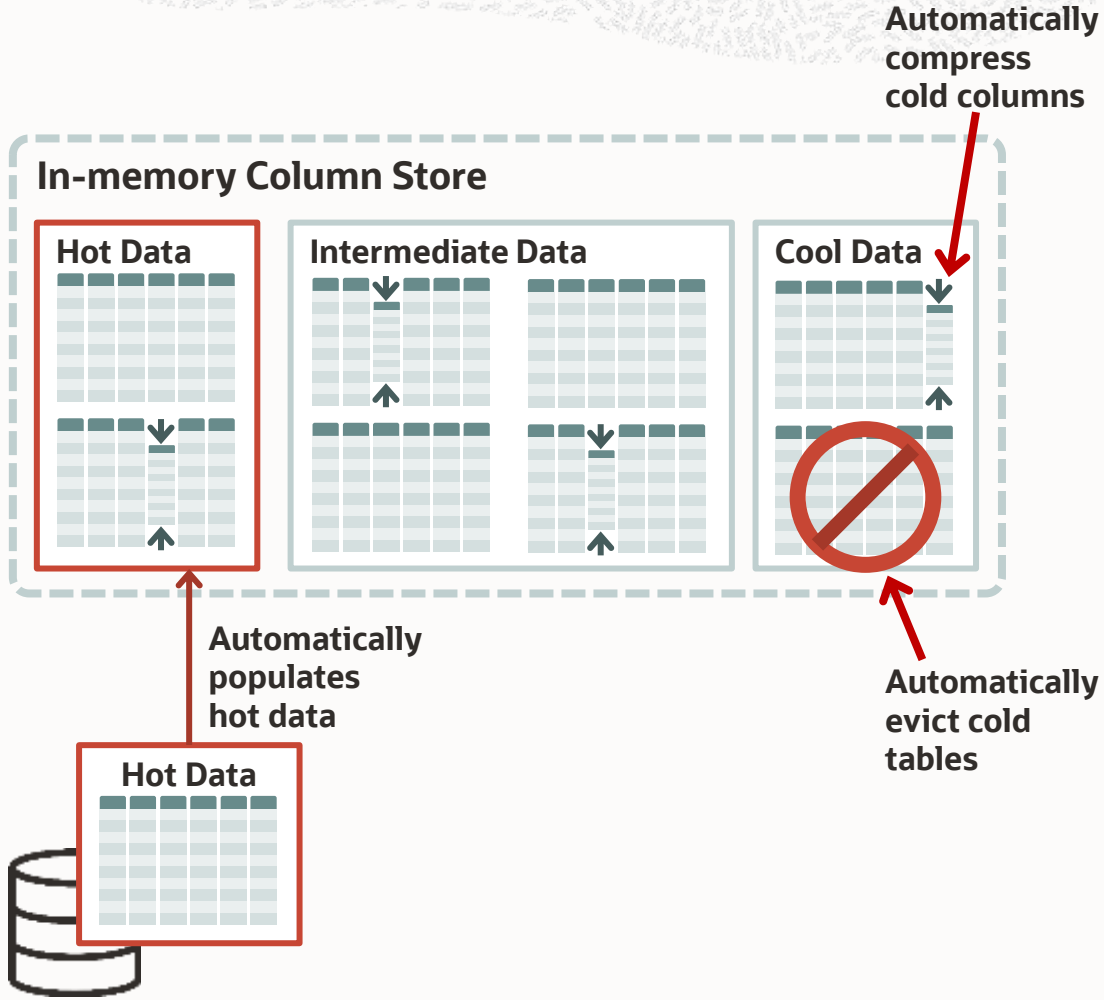
Reduces risk by immediately finding, testing, and repairing SQL plans

Improves **application availability**

- Repairs are automatic, eliminating the need for slow manual intervention

# Automatic In-Memory

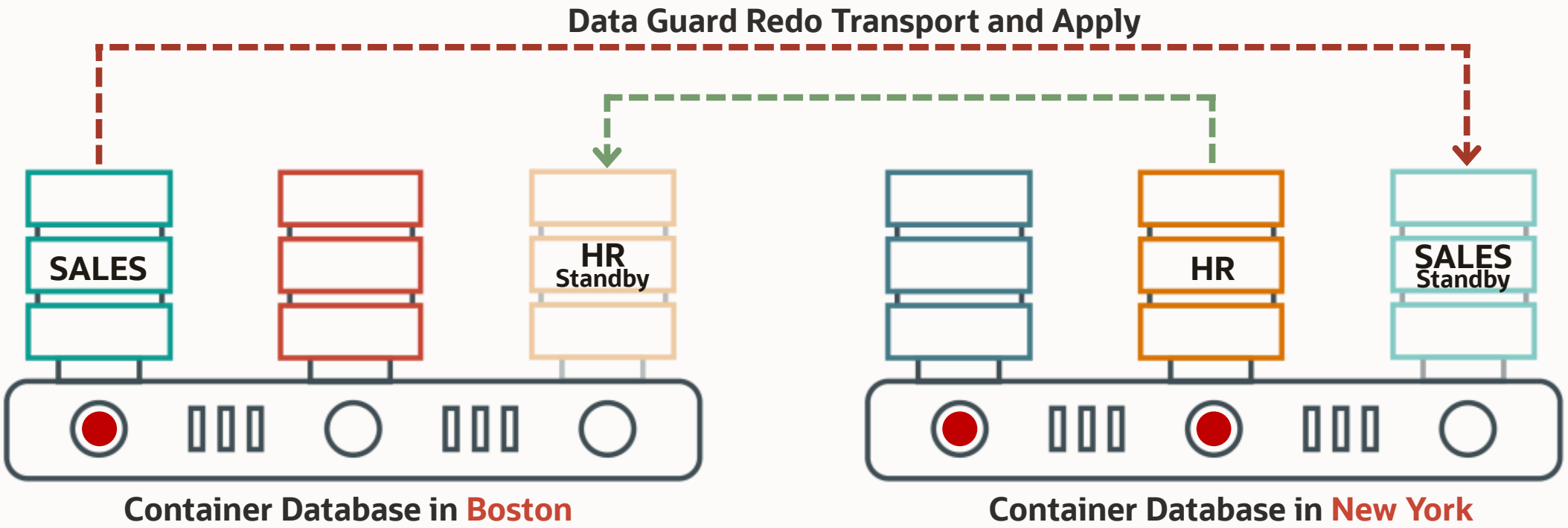
- Optimizes a SQL workload as it changes without manual intervention
- Maintains the working data set in the IM column store
- Enable by setting `INMEMORY_AUTOMATIC_LEVEL` to `HIGH`



# Data Guard for Pluggable Databases

21c

DB 21.4 – October 2021

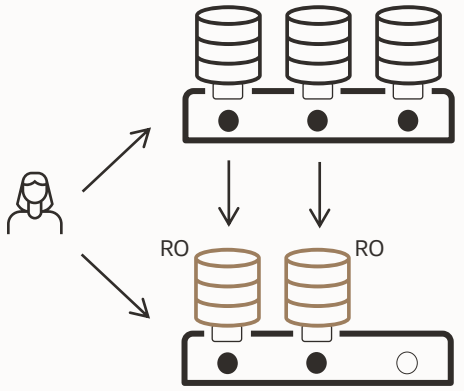


# Read-only Per PDB Standbys

23ai

In Oracle Database 21c, we introduced **Per-PDB Data Guard (DGPDB)**

- Supports two primary Container DBs running on different sites
- Each PDB is  open Read-Write on one site at a time
- Protection on PDB or CDB level using real-time Apply
- Each PDB can independently failover to remote site so no need to failover a full CDB
- Per PDB automatic redo log fetching from the source




**Benefits**:

- Customers can balance the workload between two different sites while maintaining the Multitenant consolidation benefits
- The role transition for a single PDB is significantly faster than doing the same at the CDB level
- In Oracle Database 23ai, Per-PDB Data Guard now supports the standby PDBs being opened in Read-Only mode to **allow offloading of reporting to the DGPDB standby**

# Oracle Database 19c/21c/23ai New Features





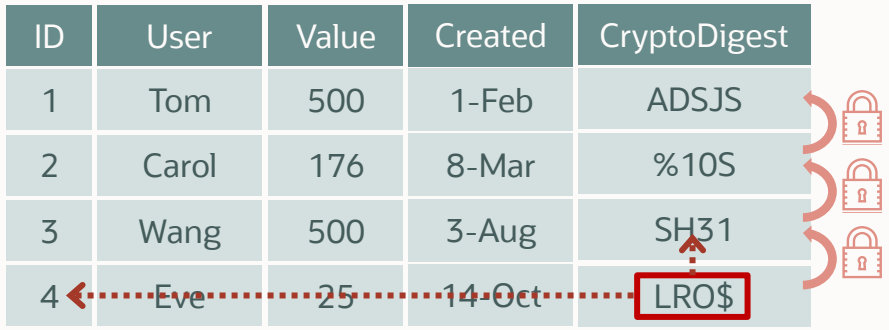
| Application Development | Big Data & Data Warehousing | Management | Performance & High Availability | Security |
|---|---|---|---|---|

# Oracle Blockchain Tables

19c

- An immutable table that automatically chains new rows to existing rows cryptographically
- Append-only tables in which only insert operations are allowed
- Use to implement blockchain applications where the participants trust the Oracle Database provider, but want means to verify that their data hasn't been tampered with.
- Blockchain Tables are simple to integrate into existing applications – they look like standard tables with declarative

```
CREATE BLOCKCHAIN TABLE
  trade_ledger …;
```




**TRADE_LEDGER**

**New feature in Oracle 21c – backported to Oracle 19.10**

# In-Database SQL Firewall

Oracle SQL Firewall is an easy-to-use firewall solution, with minimal performance and operation overhead, for ALL Oracle Database deployments

SQL Firewall is **embedded** and built into the database, ensuring that it cannot be bypassed

- Embedding also gives the firewall full visibility into the top-level SQL, stored procedures and the related database objects

Oracle SQL Firewall offers **real-time protection** against common database attacks by monitoring and blocking “unauthorized SQL” and SQL injection attacks from inside the database

- It first collects all SQL that should be allowed (“allow-list”), and then detects, blocks, and logs any unexpected SQL
- SQL Firewall can also use session context data such as IP address for its enforcement

# Schema Privileges

- It is complex to manage privileges on all the database tables, views, and procedures used by an application

- Now this is simple using `GRANT` on a Schema
- More secure than granting * `ANY` privileges to users

```
-- Grant Scott SELECT on all tables in sales
schema

GRANT SELECT ANY TABLE
   ON SCHEMA sales
   TO scott;
```

# DEVELOPER Role

- It is complex to grant all the privileges developers need to create, debug, and tune apps
- This is now simple using the 'DEVELOPER' role
- The `DB_DEVELOPER_ROLE` includes :
  - `ADMINISTER SQL TUNING SET`
  - `CREATE ANALYTIC VIEW`
  - `CREATE ATTRIBUTE DIMENSION`
  - `CREATE CUBE`
  - `CREATE CUBE BUILD PROCESS`
  - Etc.

```
-- Provide user Scott all privileges
needed to create new app

GRANT DB_DEVELOPER_ROLE TO scott;
```

livelabs.oracle.com

RAG example with Oracle AI Vector Search

- https://livelabs.oracle.com/pls/apex/dbpm/r/livelabs/view-workshop?wid=4127

Easily Build Powerful Analytic Applications Using Oracle APEX

- https://livelabs.oracle.com/pls/apex/dbpm/r/livelabs/view-workshop?wid=3726

# Thank you


valentin.tabacaru@oracle.com
linkedin.com/in/valentinoracle
@valentindb2


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


ORACLE