ORACLE

23ai

Oracle Database 23ai

New Features

Oracle Database Vision

Make modern apps and analytics
easy to develop and run
for all use cases at any scale

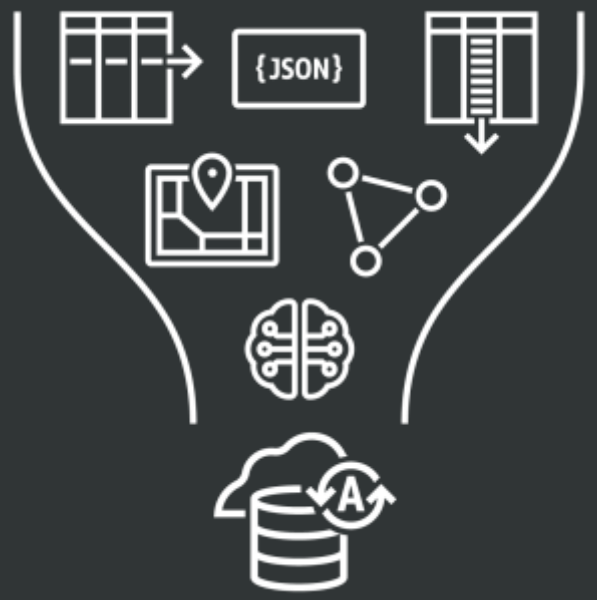
Oracle Database Vision

With Generative AI (LLM)

Make modern apps and analytics
easy to **generate** and run
for all use cases at any scale

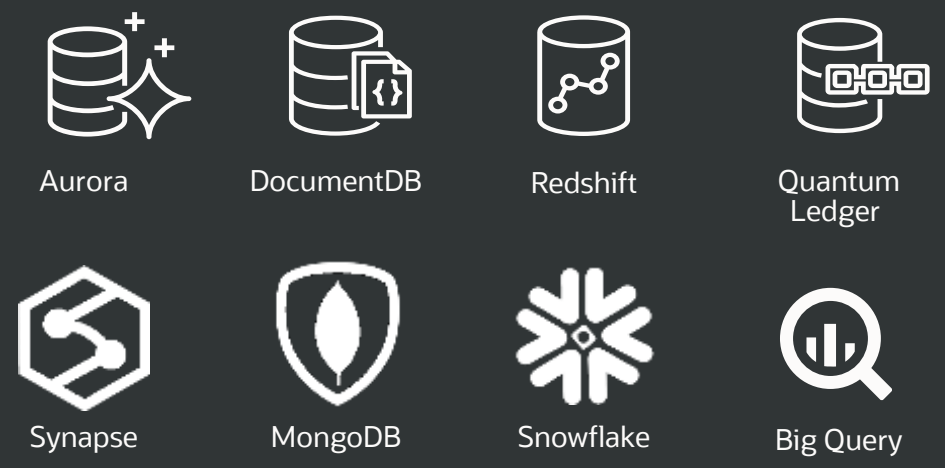
Comparing Database Strategies

Run converged, open, SQL Database



Developers and IT focus on Innovation

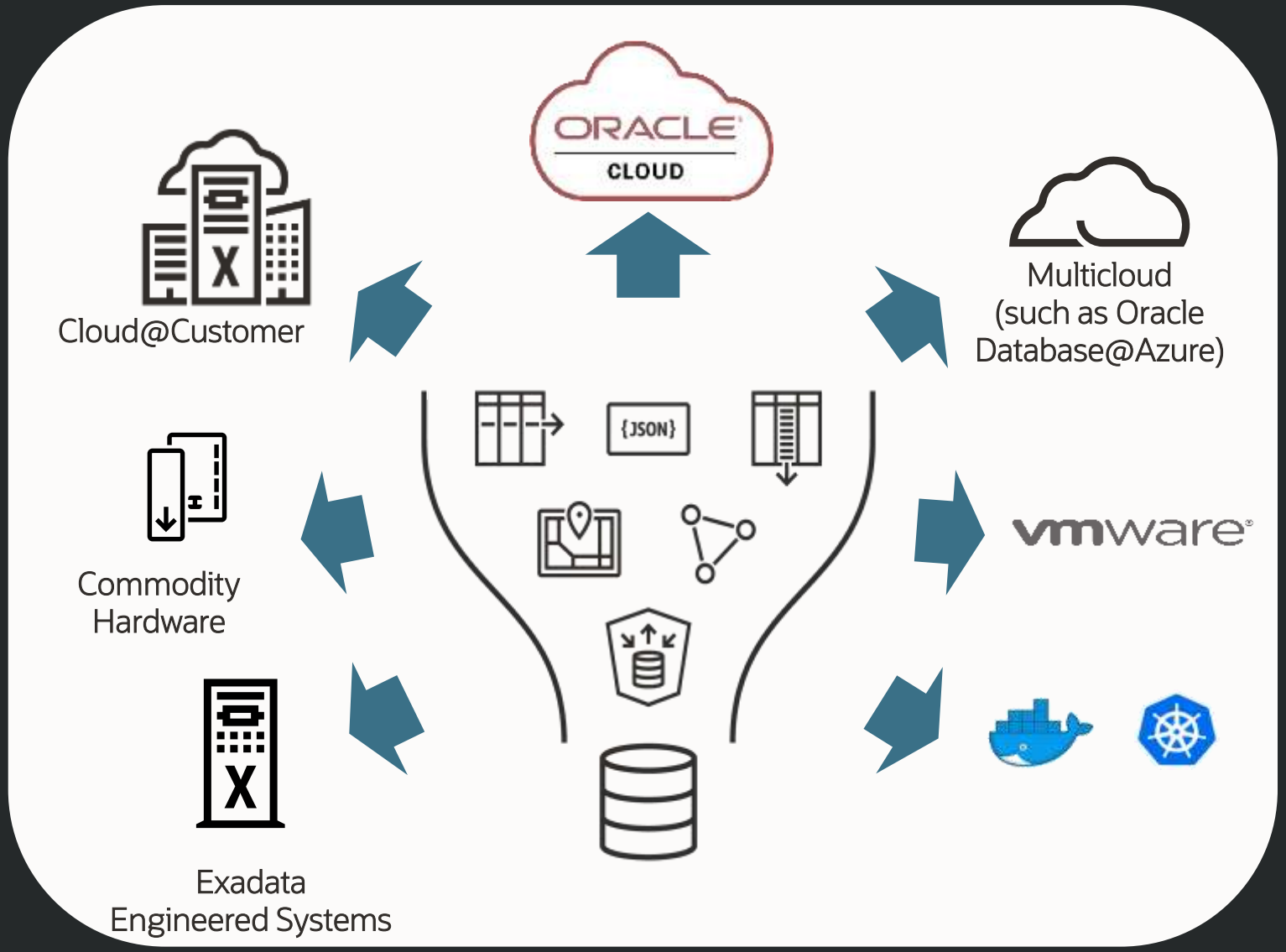
Instead of single-use proprietary databases



Developers and IT focus on Integration

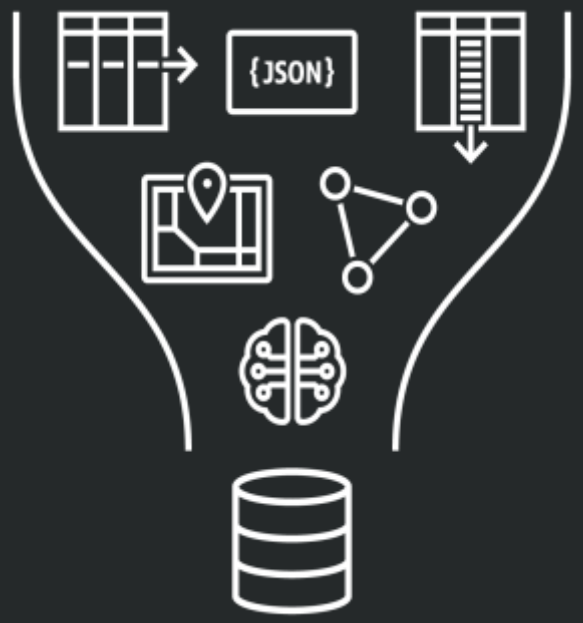
Oracle Database Deployment Choices

Develop and deploy Oracle anywhere – extreme portability



Same database,
same skills

Next Generation Converged Database – Database 23ai



The Converged Database approach has been very successful

- But there are still tradeoffs that make it tempting to deploy solutions which sacrifice the data consistency and efficiency of the relational data model for easier app development

Oracle is introducing revolutionary new Converged Database technologies that unify data models at a more fundamental level

These definitively eliminate the remaining tradeoffs

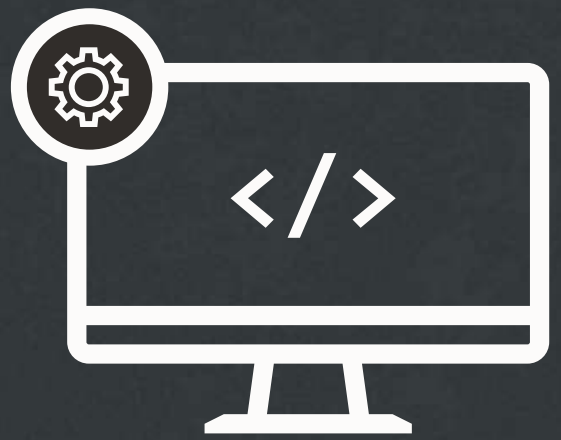
Next-Generation Converged Database

Over 300 major new features plus thousands of enhancements

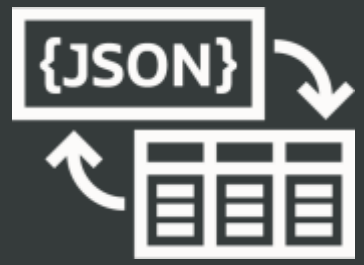
The latest long-term support release of Oracle's flagship Database

Customers with support contracts get free upgrade

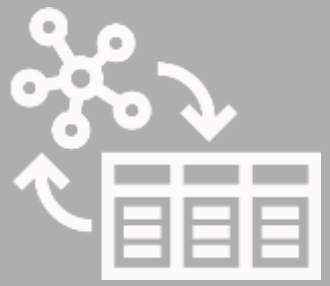
What’s new in Oracle Database 23ai for App Dev?



JSON Relational Duality Views



Property Graph Views



JSON Relational Duality Views allow the generation of JSON format and APIs from relational tables

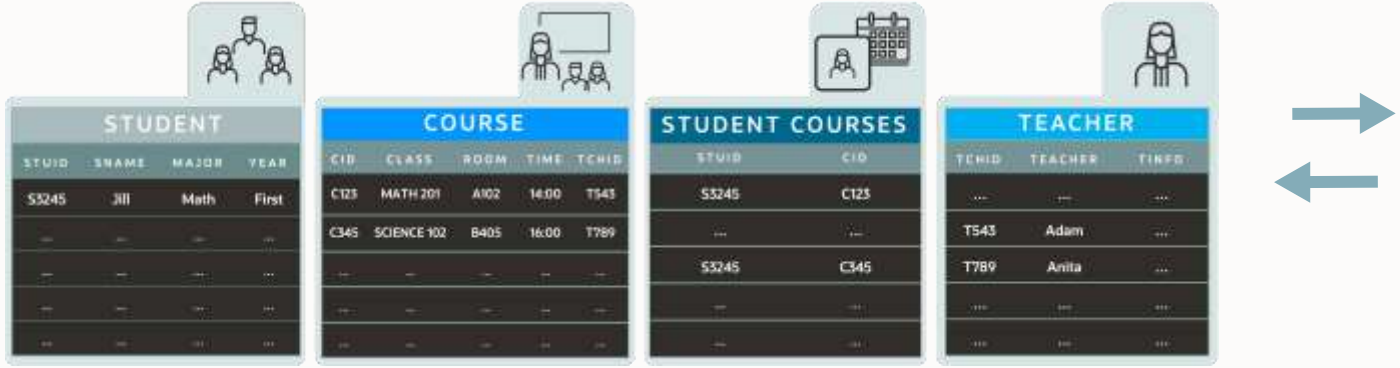
The data for this app can be stored in normalized relational tables

- Great for storage independence, data consistency, declarative SQL

The app can simply GET a JSON document with all the needed data in a hierarchical format

The app can then edit the document and PUT it back

- The database automatically takes care of updating the tables



SCHEDULE FOR: JILL

```
{
 "student"     : "S3245",
 "name"        : "Jill",
 "major"       : "Math",
 "schedule"    :
   [  {
        "time"    : "14:00",
        "course"  : "Math 201",
        "room"    : "A102",
        "teacher" : "Adam"
      },
      {
        "time"    : "16:00",
        "course"  : "Science 102",
        "room"    : "B405",
        "teacher" : "Anita"
      }
   ]
}
```

Oracle Database 23ai JSON Relational Duality

Storage Format

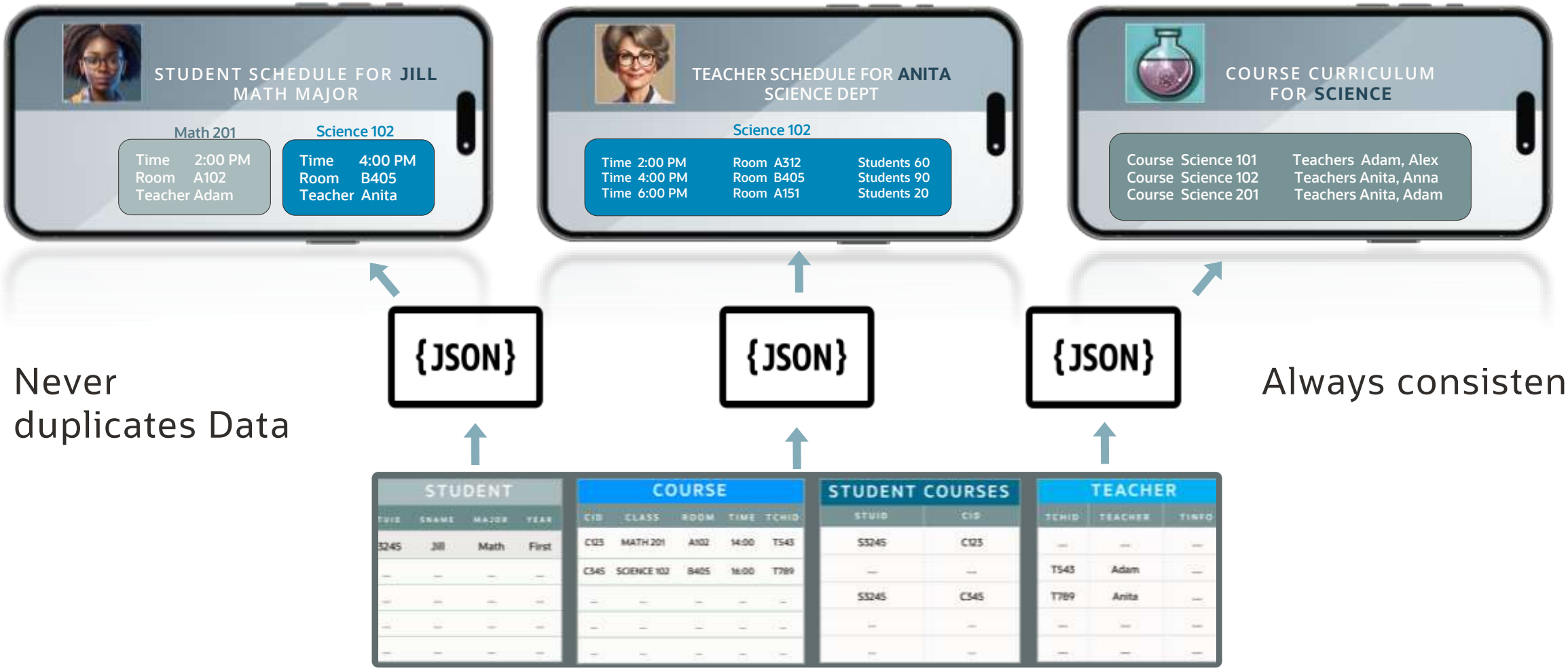
first_name	order	quantity
Jake	{ “product” : ”TV”, “price” : 250 }	1

Access Formats

```
{
 "first_name" : “Jake",
 "order" :
    {
        "product" : "TV",
        "price" :  250
    },
 "quantity" : 1
}
```

JSON Relational Duality delivers the simplicity of JSON for accessing data plus the simplicity of relational for storing and manipulating data

JSON Duality views allow the same underlying data to be customized to match the needs of each app use case



JSON Relational Duality View | Using SQL or GraphQL

Create using regular SQL

```
create JSON relational duality view DEPT_EMP_SQL as
  select JSON { 'deptno' : d.deptno,
                'dname'  : d.dname,
                'staff'  : [ select JSON { 'empno'    : e.empno,
                                           'ename'    : e.ename }
                             from EMP e with insert update delete
                             where e.deptno = d.deptno ]}
  from DEPT d with insert update delete;
```

Or use GraphQL

```
create JSON relational duality view DEPT_EMP_GraphQL AS
  DEPT @insert @update @delete
     {deptno  : deptno,
      dname   : dname,
      staff   : EMP @insert @update @delete
                  [  {empno  : empno,
                      ename  : ename} ]};
```

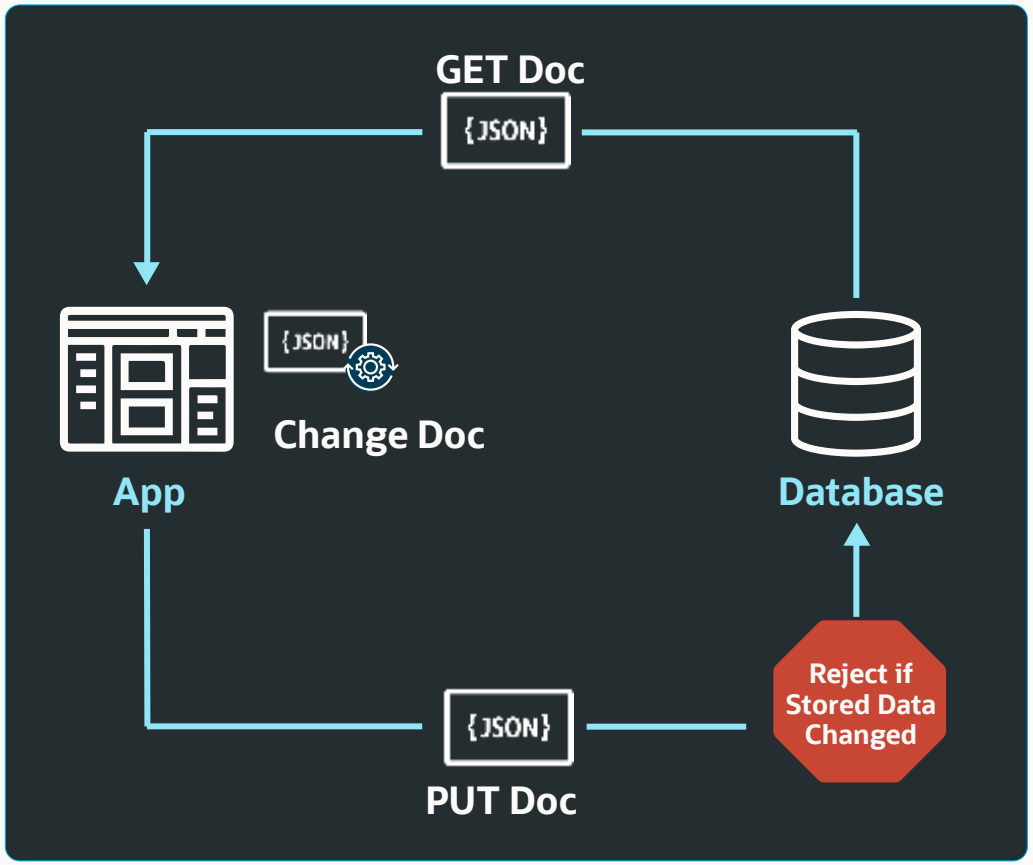
Game Changing Lock-Free Concurrency Control

Conventional locking does not work when REST GET and PUT APIs are used

Value-based concurrency control - The database automatically detects when the database data underlying a document has hung between the initial document read and the subsequent write

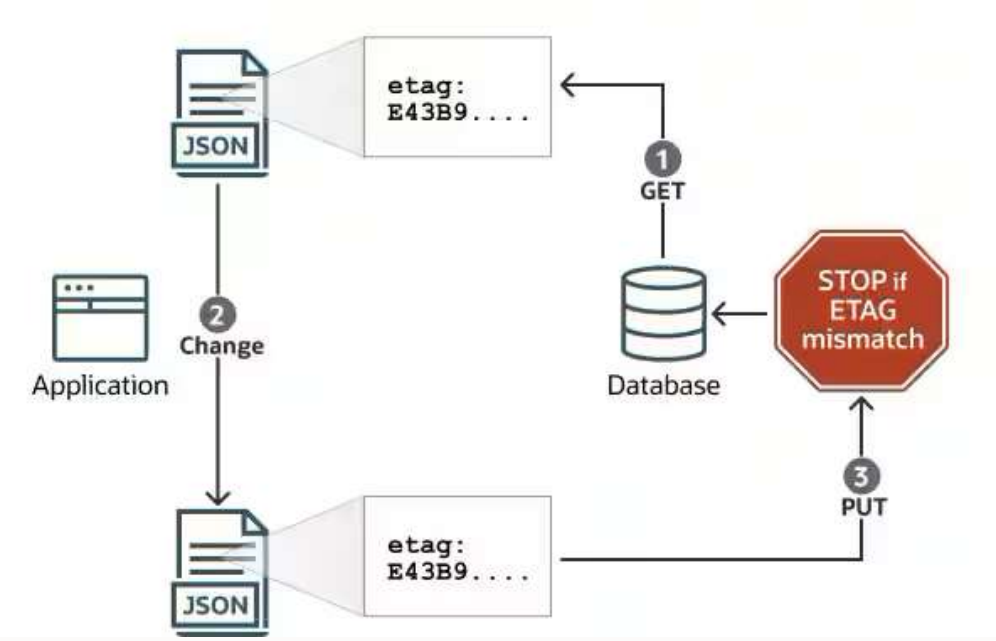
- If a change is committed, the write operation is automatically rejected and returns an error
- The app can then reissue the write based on the changed data
- Called **Optimistic Concurrency Control**

- Great for interactive applications since the data is not locked during human thinking time
- Great for mobile disconnected apps since writes of stale documents are rejected



Value-Based Concurrency Control

- **ETAGs** are not just for documents
- **ETAGs** can be used now on pure relational data for lock-free row updates using SQL
- Great for interactive or mobile applications that directly access tables

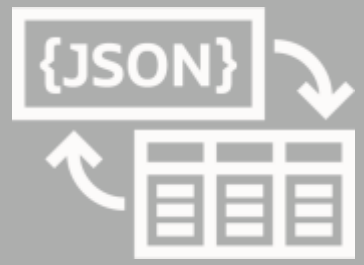



```
-- Select the ETAG along with data columns
SELECT stid, name, sinfo, SYS_ROW_ETAG(stid, sinfo)
  FROM student
 WHERE name = 'JILL';
```

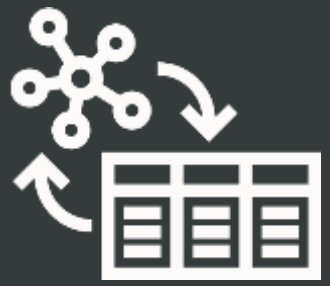
Human think time

```
-- Validate the ETAG before updating
UDPATE student SET sinfo = 'New Data’
 WHERE  name = 'JILL'
   AND  SYS_ROW_ETAG(stid, sinfo) = 'xxxx';
```

JSON Relational Duality Views

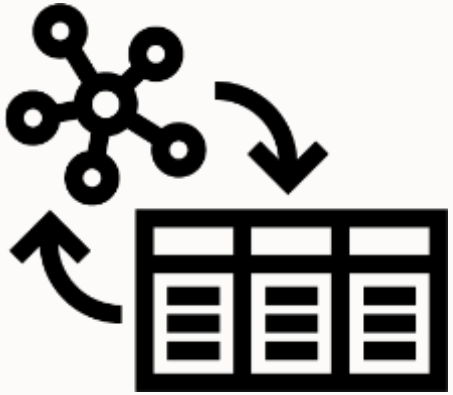


Property Graph Views



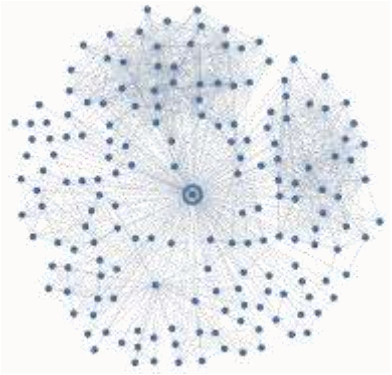
Property Graph Views

treat relational data as vertices or edges

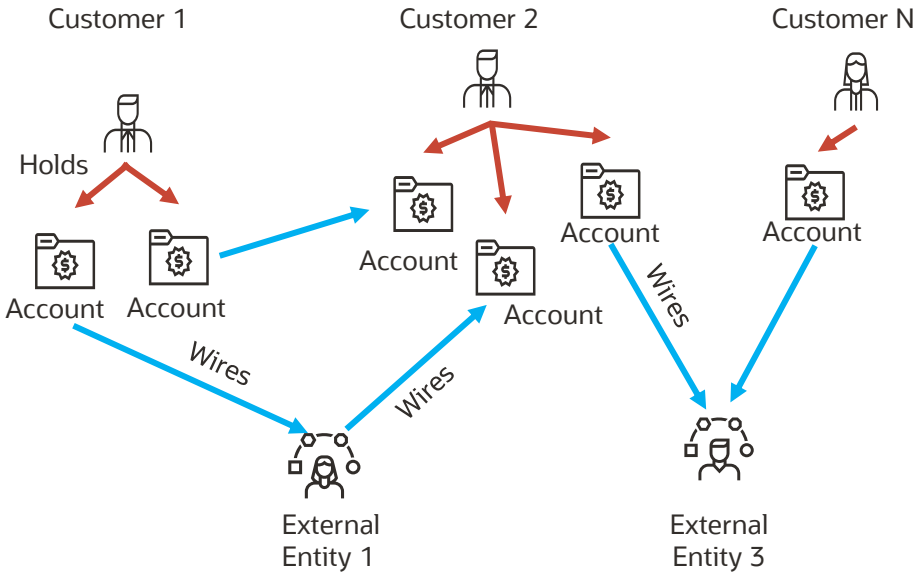


Connections between entities are everywhere

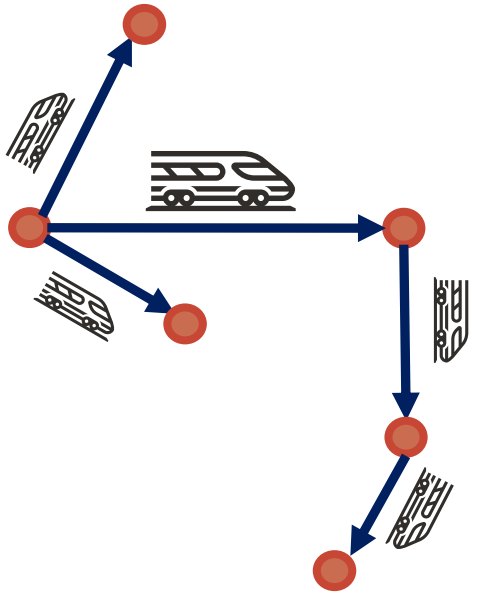
Social networks



Customer Interactions



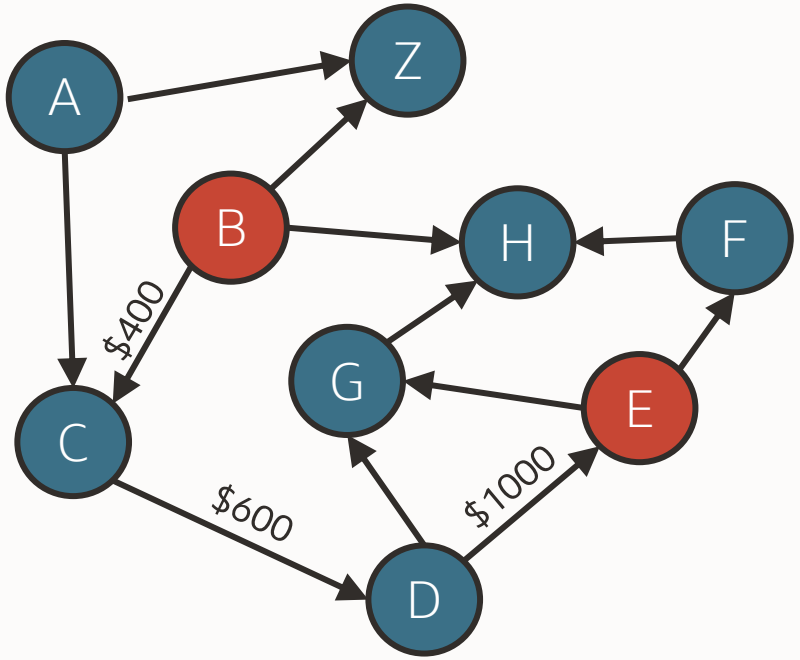
Transportation Networks



Graphs are a powerful way to query connections and relationships between data

Property Graph Views in Oracle Database 23ai
declare intent to treat data as vertices or edges in a graph

For example, to discover indirect money movements from bank account 'B' to bank account 'E'



Oracle Database 23ai Operational Property Graphs

- **First** commercial database to implement SQL/PGQ standard
- Enables easy use of graph analytics in **transactional** systems

Before SQL Property Graph

```
-- Snippet from a 50+ Line SQL Query
SELECT "v1.id" AS "THOM_ID", "v2.id" AS "LARRY_ID"
FROM (SELECT "V1"."ID" AS "v1.id", "V2"."ID" AS "v2.id"
FROM "GRAPHUSER"."USERS" "V1", "GRAPHUSER"."USERS" "V2", (WITH t1(src_table, src_key,
        dst_table, dst_key,
        exp, lvl)
AS (
  -- Anchor member.
  SELECT "src_table" AS src_table, "src_key" AS src_key, "src_table" AS dst_table, "src_key" AS dst_key,
  '' AS exp, 0 AS lvl   FROM(SELECT 'USERS' AS "src_table", v1.id AS "src_key"
FROM "GRAPHUSER"."USERS" "V1"
WHERE ("V1"."ID" = 1))
  UNION ALL
  -- Recursive member.
  SELECT t1.src_table, t1.src_key,
        t2."dst_table" AS dst_table, t2."dst_key" AS dst_key,
        t1.exp || t2."exp" AS exp,
        t1.lvl + 1 AS lvl
   FROM (SELECT 'USERS' AS "src_table", "anonymous_1".follower_id
   AS "src_key", 'USERS' AS "dst_table", "anonymous_1".followed_id AS "dst_key", '' AS "edge_table",
   '' AS "edge_key", (('<EXPRESSIONS>' || '') || '</EXPRESSIONS>') AS "exp"
FROM "GRAPHUSER"."FOLLOWS" "anonymous_1"
WHERE (NOT("anonymous_1"."FOLLOWER_ID" IS NULL) AND NOT("anonymous_1"."FOLLOWED_ID" IS NULL))) t2, t1
... 40+ lines
```

SQL Property Graph

```
SELECT DISTINCT Thom_ID, Harry_ID from graph_table(social_graph
MATCH (v1)-[IS follows]->{1,5}(v2)
WHERE v1.id = 1 AND v2.id = 15
COLUMNS (v1.id AS Thom_ID, v2.id AS Harry_ID));
```

Create a Graph with Data in these Tables

https://www.youtube.com/watch?v=4uknPkJkUIk

https://oracle-base.com/articles/23/sql-property-graphs-and-sql-pgq-23

BANK_ACCOUNTS

ID	Name	Balance
1		
2		
3		
...		
672		
673		
674		
...		
831		
832		
833		

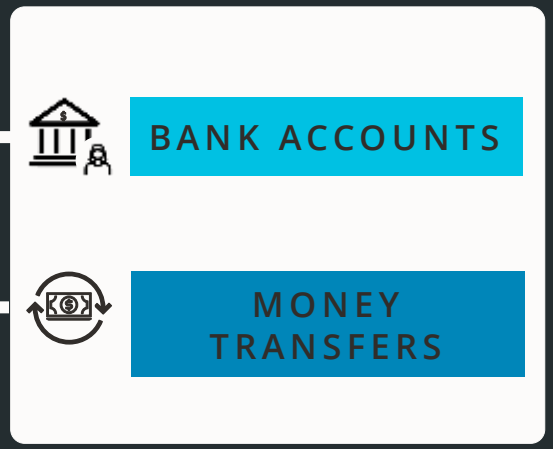
BANK_TRANSFERS

SRC_ACCT_ID	DEST_ACCT_ID	DESCRIPTON	AMOUNT
1	672		1000
1	584		1000
1	259		100000
2	833		5001
2	840		7050
2	493		4363
...	...		...

Defining a property graph view is simple

Just declare the tables whose rows represent vertices or edges in the graph

```
CREATE PROPERTY GRAPH bank_graph
       VERTEX TABLES (
         bank_accounts as accounts
         PROPERTIES (id, balance))
       EDGE TABLES  (
         bank_transfers
         SOURCE KEY      (from_acc) REFERENCES ACCOUNTS(ID)
         DESTINATION KEY (to_acc)   REFERENCES ACCOUNTS(ID)
         PROPERTIES (amount, to_acc))
;
```



Create a Graph with Data in these Tables

BANK_ACCOUNTS

ID	Name	Balance
1		
2		
3		
...		
672		
673		
674		
...		
831		
832		

1000

BANK_TRANSFERS

SRC_ACCT_ID	DST_ACCT_ID	DESCRIPTON	AMOUNT
1	672		1000

```
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)
    );
```

Graph name

Table holding vertex data

Column values become properties

Table holding edge data (connections you want to traverse)

SOURCE and DESTINATION KEYS link vertices with the edge

Column values become properties

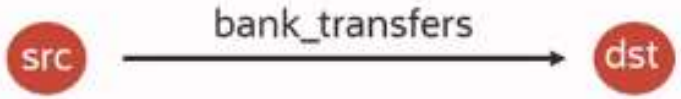
Querying the view is simple

/* List accounts which have transferred money to another account */

```
select account_id from GRAPH_TABLE(bank_graph
   MATCH (src)-[is bank_transfers]->(dst)
   COLUMNS (src.id as account_id) );
```

Vertices are in (), edges are in []

Columns to return



Querying the view is simple

/* List accounts which have transferred money to another account */

```
select account_id from GRAPH_TABLE(bank_graph
   MATCH (src)-[is bank_transfers]->(dst)
   COLUMNS (src.id as account_id) );
```

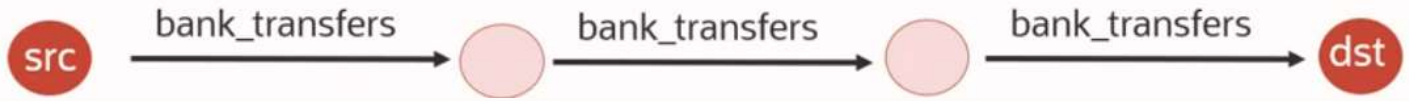
Vertices are in (), edges are in []

Columns to return

/* List accounts which have transferred money through intermediate accounts */

```
select account_id from GRAPH_TABLE(bank_graph
   MATCH (src)-[is bank_transfers]->{1,3}(dst)
   COLUMNS (src.id as account_id) );
```

Path length in {} – from 1 to 3 hops

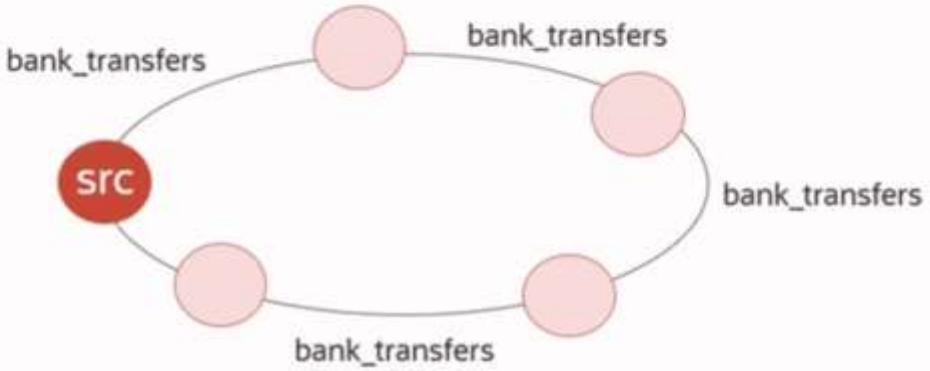


Querying the view is simple

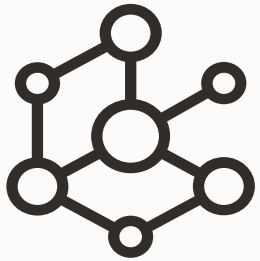
/* List accounts which have 5 hop transfers that start and end with the same account, and order by number of such cycles*/

```
SELECT acct_id, COUNT(1) AS Num_5hop_Chains
    FROM graph_table (BANK_GRAPH
        MATCH (src) - []->{5} (src)
        COLUMNS (src.id AS acct_id)
    ) GROUP BY acct_id ORDER BY Num_5hop_Chains DESC;
```

Starting vertex and ending vertex are the same



With Oracle Database 23ai, one part of an app can treat the data as relational, while other parts treat the same data as a document, and others treat it as a graph



You get the best of all these worlds, at the same time
Another huge benefit for app dev

Oracle Database 23ai – Additional Features For App Dev

Boolean Datatype

A more intuitive way of storing and manipulating logical values within the database

```
CREATE TABLE customers(
        cust_id number,
        Active boolean);

SELECT cust_id
FROM customers
WHERE active;
```

JavaScript Stored Procedures

JavaScript joins PL/SQL & Java as first-class server-side dev languages

Executed by our fast Multilingual Engine (MLE), powered by GraalVM

Reduces the number of roundtrips to the database

Wider Tables

Support for up to 4096 columns per table

Simplifies development of applications that need large numbers of attributes such as for ML and IoT workloads

```
ALTER SYSTEM SET
max_columns = EXTENDED;
```

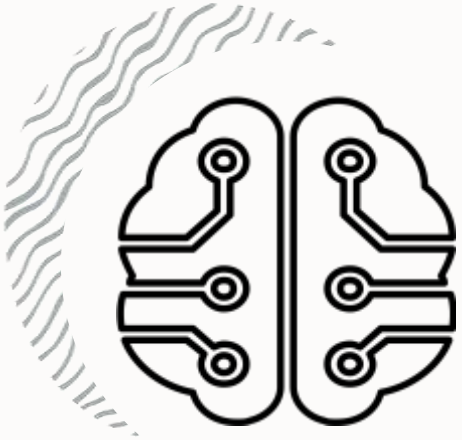
Lock-free Column Value Reservations

Allows applications to reserve part of a value in a column without locking the row

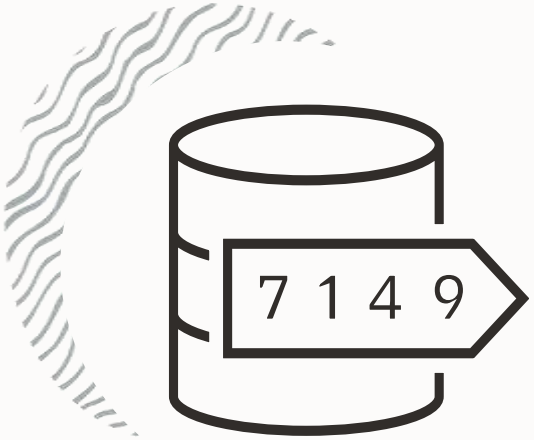
For example, reserve part of a bank account balance or reserve an item in inventory without locking out all other operations on the bank account or item

What’s new in Oracle Database 23ai for AI?

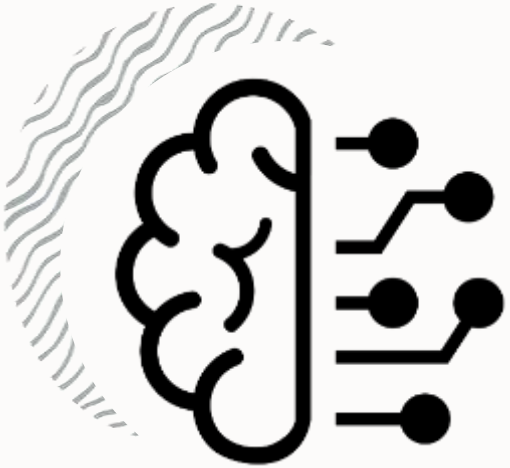
Native AI in Oracle Database



In-database AI/ML Predictive models



AI Vector Search



Generative AI

Oracle Database is a Machine Learning Engine

Continuously Enhanced for over 20 years

Oracle makes it simple to make data-driven predictions

- Use declarative SQL to build models and run Machine Learning directly on business data

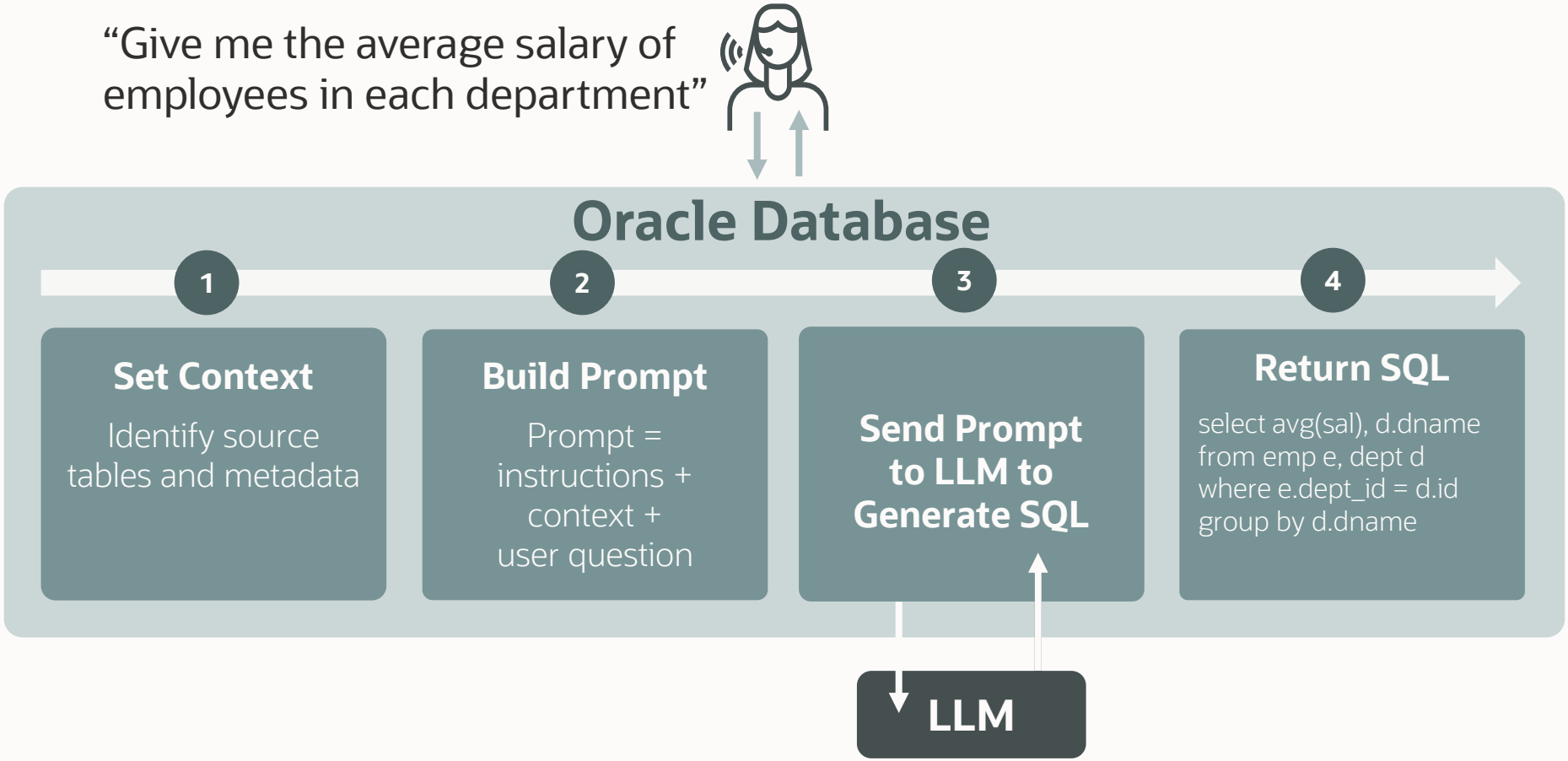
Over 30 in-database parallel and scalable ML algorithms

- Eliminates costly, risky, and slow data movement to separate ML engine

Business Use Case	ML Technique	Oracle ML Algorithms
Customer Loyalty and Retention	Classification	SVM, GLM, Random Forest, XGBoost, et al.
Customer Segmentation	Clustering Classification	K-Means, Expectation Maximization Decision Tree
Demand Forecasting	Time Series Regression	Exponential Smooth SVM, GLM, Neural Networks, XGBoost
Cross-sell / Up-sell	Association Rules Classification	A priori SVM, GLM, Random Forest, XGBoost, et al.
Credit Risk	Regression Classification	SVM, GLM, Neural Networks, XGBoost SVM, GLM, Random Forest, XGBoost, et al.

Generative AI Use Case

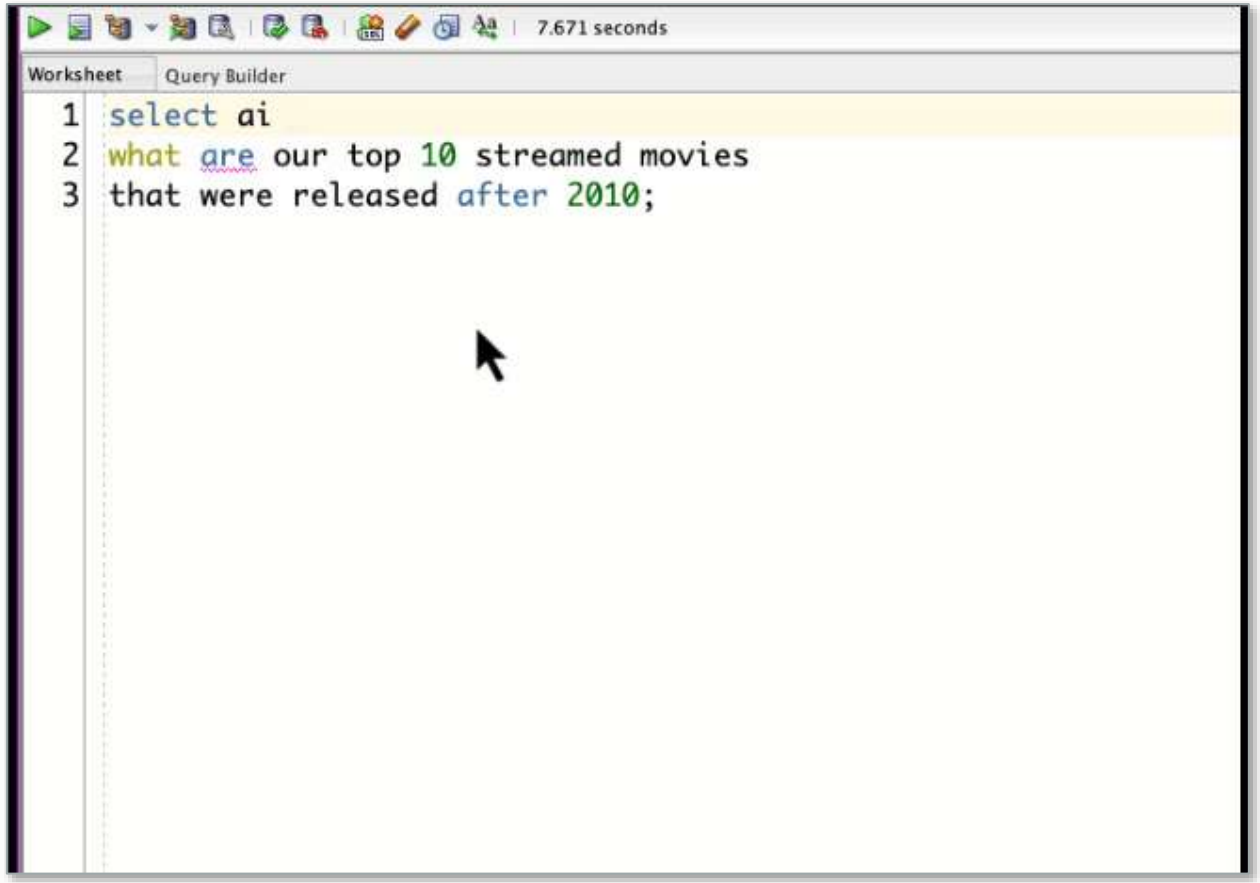
SQL code generation from natural language using LLM



SQL Generation with Select AI

Available today on Oracle Autonomous Database and under Oracle Database 23.7

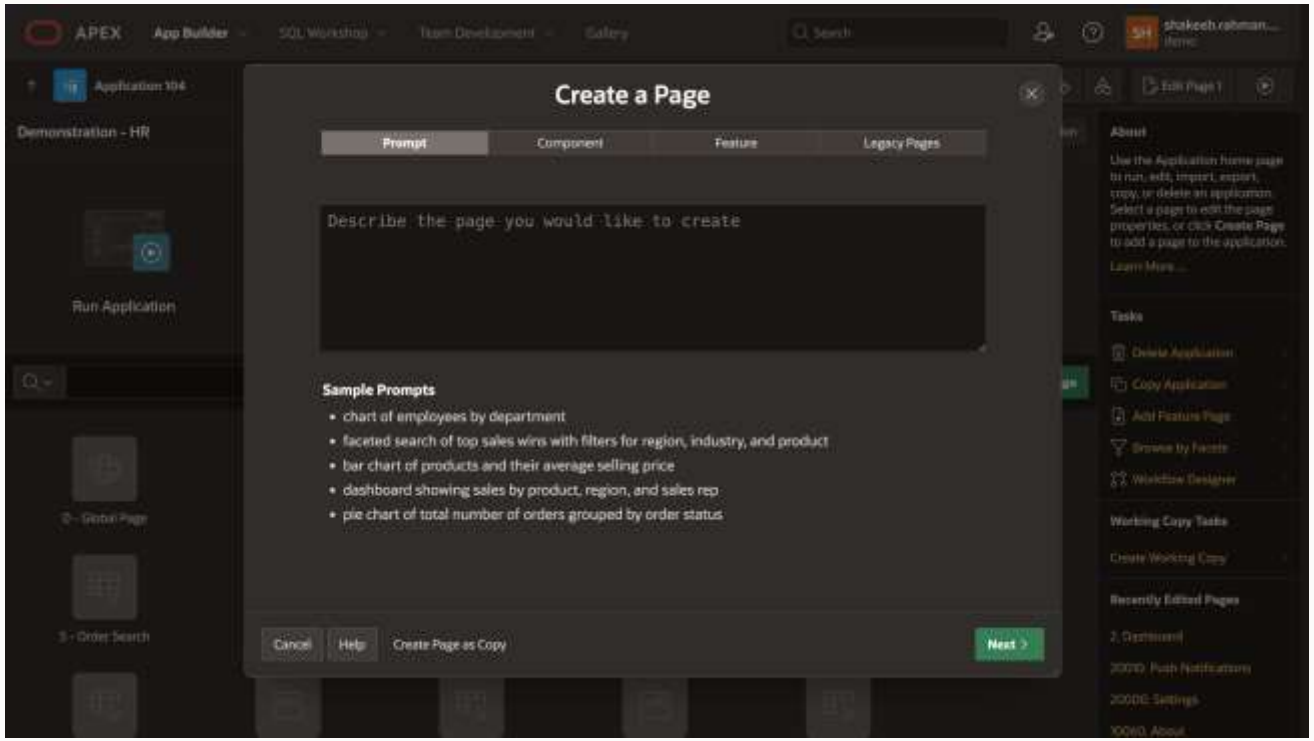
- Use natural language to query data with the help of LLMs
- Increase application developer productivity
- Enable non-technical users to query information from their database
- Invoke from SQL command line and PL/SQL function
- Inherit security and authentication of the database



https://blogs.oracle.com/machinelearning/post/introducing-natural-language-to-sql-generation-on-autonomous-database

Augmenting Oracle APEX with Generative AI

- As a low-code app development platform, Oracle APEX already enables 700k+ developers and analysts to rapidly create 19M+ operational and analytic apps
- 75% of Fortune 500 use Oracle APEX
- Adding generative AI capabilities in Oracle APEX further improves user productivity
- **APEX Assistant** will support the use of natural language in both generating SQL and generating app components
- Integrates easily with OCI AI services to develop AI-powered applications



Make it easy to generate and run modern apps and analytics for all use cases at any scale

Oracle Database Vision

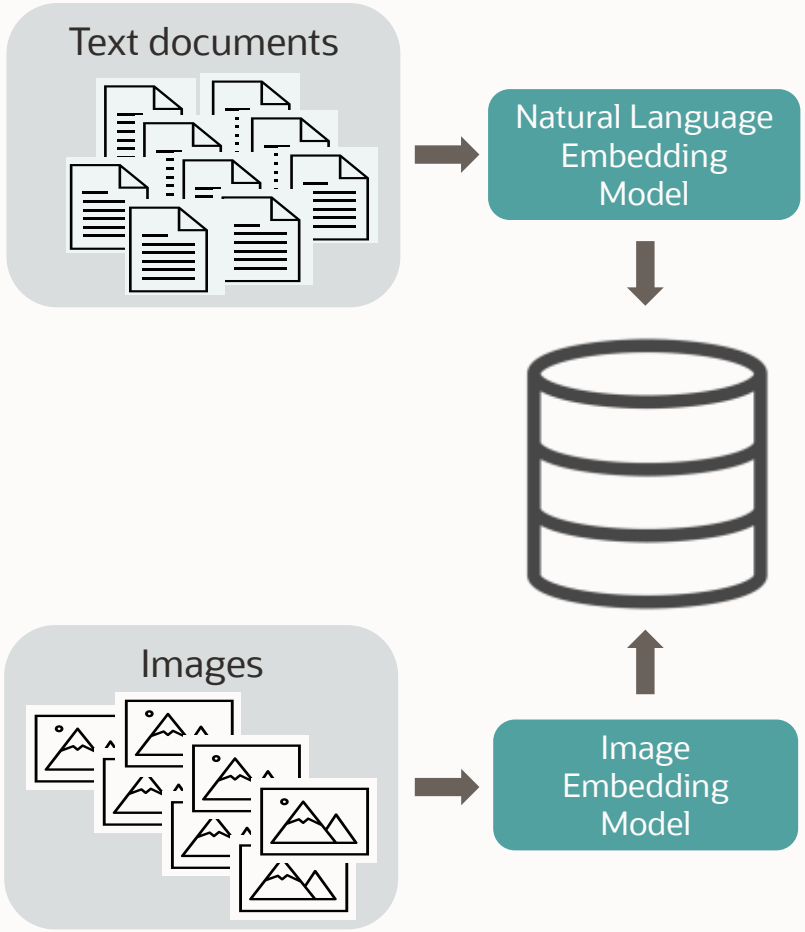
with Generative AI

A new technology called AI Vector Search enables semantic searches on unstructured data

50 21 16 42 33

Generating vector embeddings

Use an embedding model to convert unstructured data (text, images, audio & video) into vectors



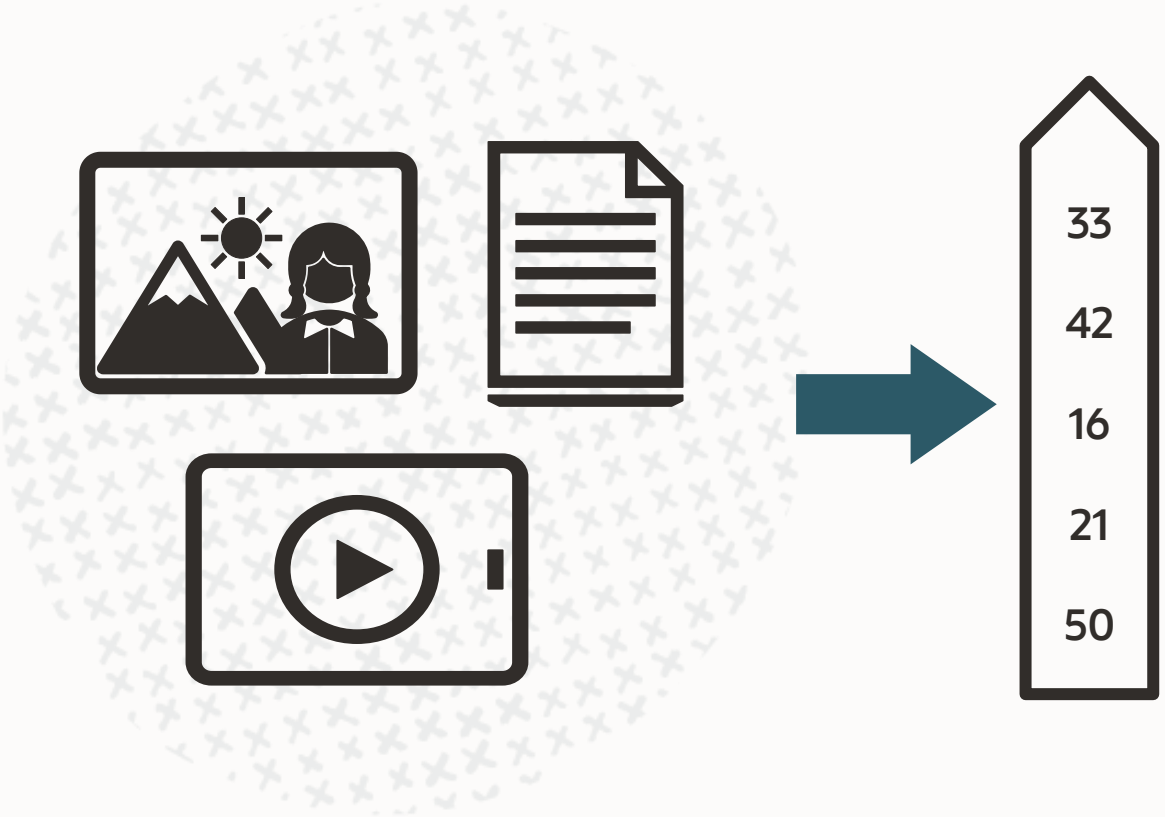
Text Vector Table

id	vector	text
1	[0.8, 0.5, 1.6, -2.5, ...]	"It was the best of times, it was the worst of times, it was.."
2	[1.1, 0.3, 0.6, -1.3, ...]	"It is a truth universally acknowledged, that a single man.."
3	[1.3, 0.1, 0.2, -1.1, ...]	"It was a bright cold day in April, and the clocks were striking.."
...	...	...

Image Vector Table

id	vector	Image
1	[0.5, 1.5, 2.6, -1.1, ...]	
2	[1.0, 0.9, 1.6, -1.3, ...]	
3	[0.6, 1.1, 1.3, -0.9, ...]	
...	...	...

Vectors are used in AI to encode unstructured data such as images, documents, videos, etc.

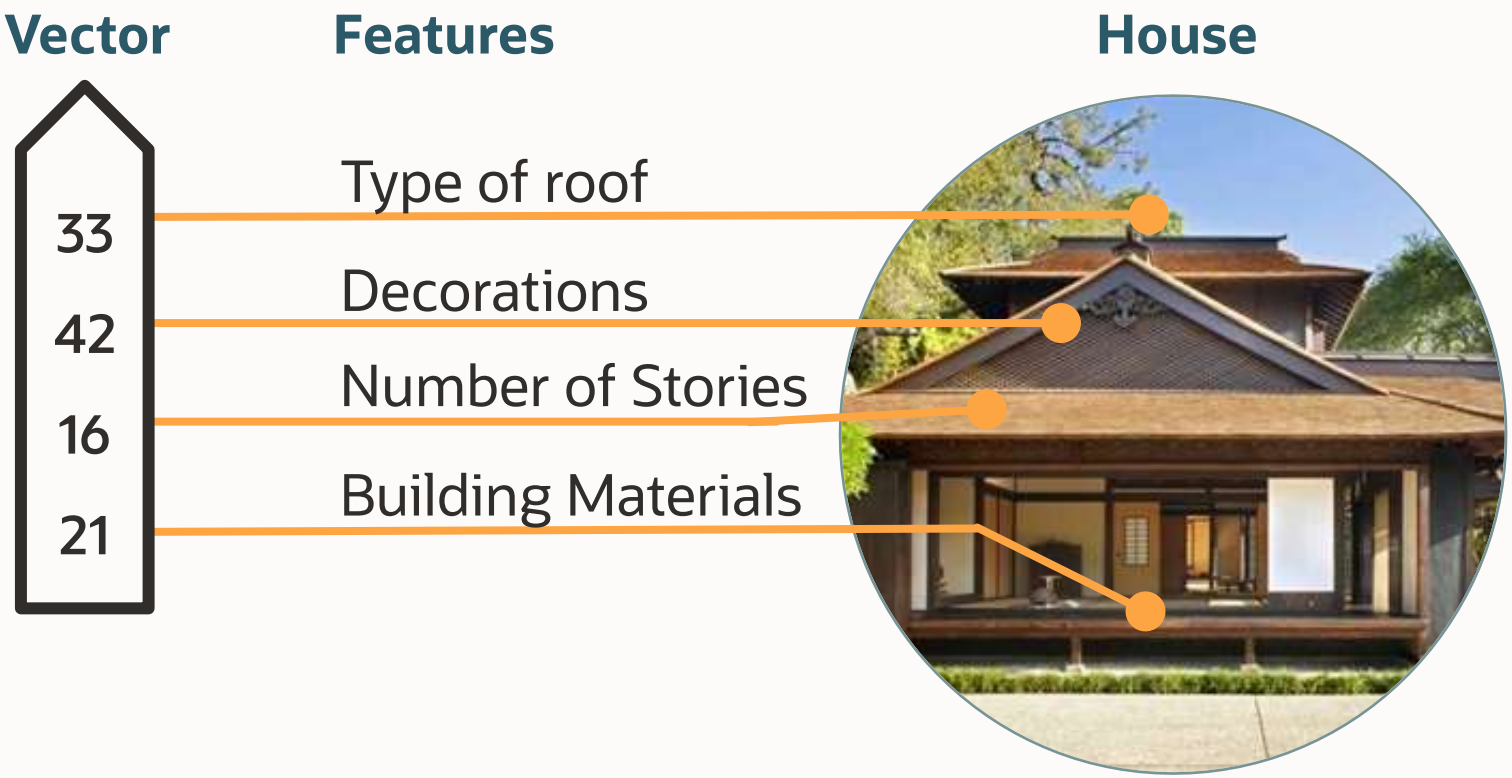


A vector is a sequence of numbers, called dimensions, representing the important “features” of the data

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

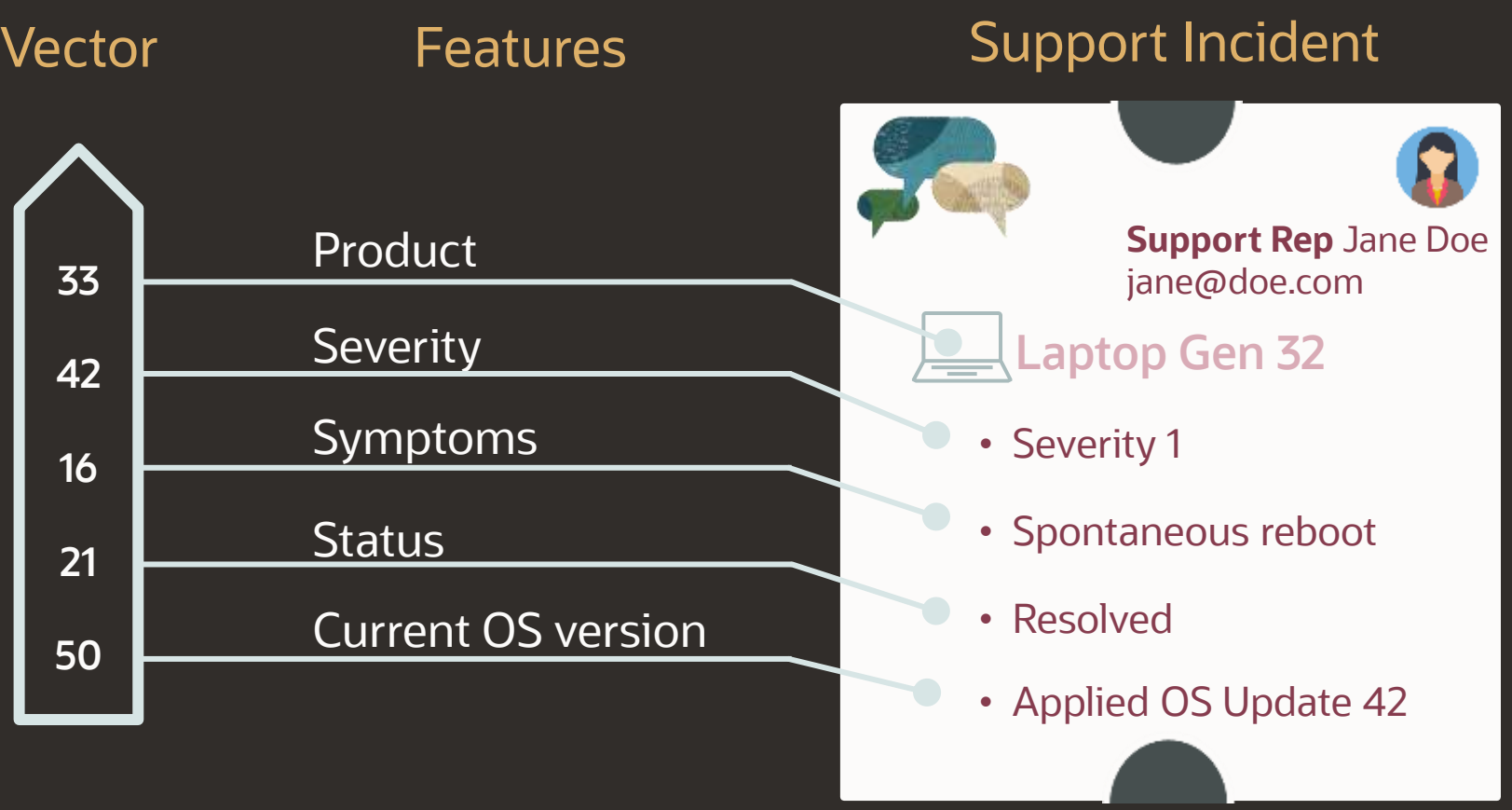
Example: the features for a house image could be



Each dimension (number), represents a different feature of the house

Note: In reality, ML algorithms determine features, so they are not as simple as shown here

Example: The Vector for a Support Incident could be ...



Each dimension (number), represents a different feature of the support incident

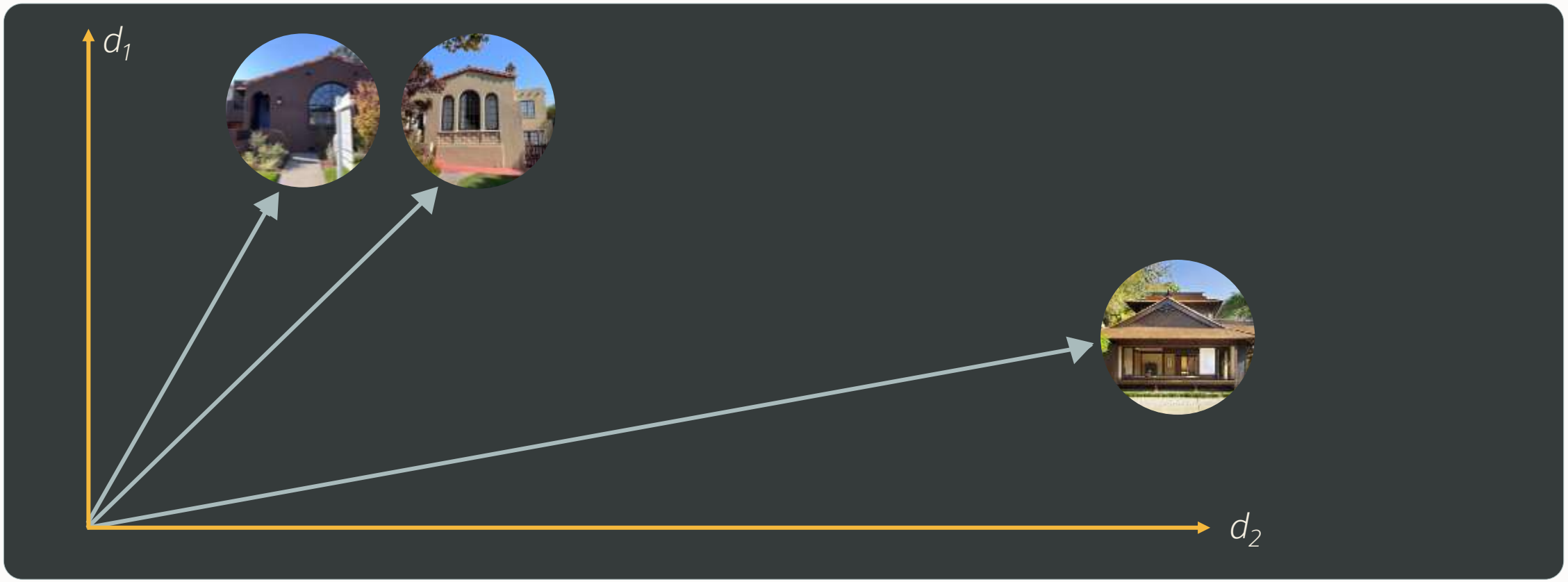
Note: Features determined by actual AI models are much more complex

The main operation on vectors is the Mathematical Distance between them

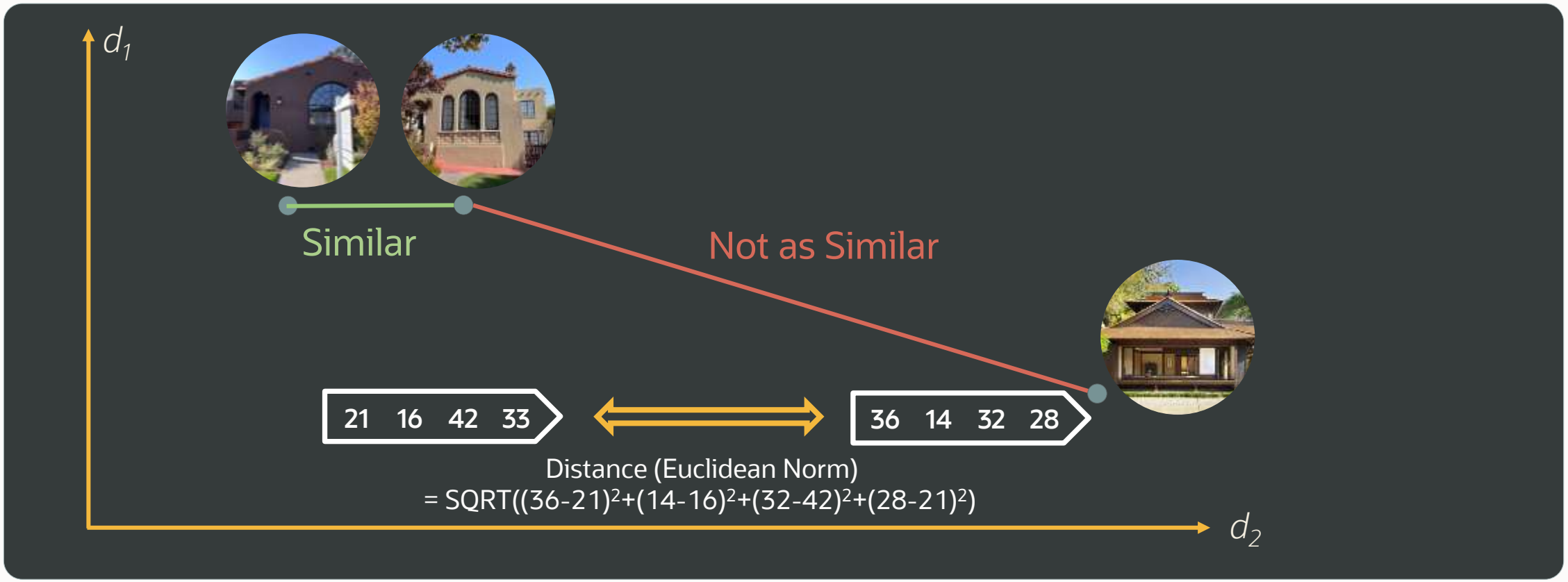
Distance (Euclidean Squared)
$= ((3-2)^2+(1-6)^2+(2-2)^2+(8-3)^2)$

There are many mathematical distance formulas

House vectors when collapsed into 2 dimensions instead of hundreds could look like this



The distance between the vectors is proportional to their semantic similarity



Introducing Oracle AI Vector Search

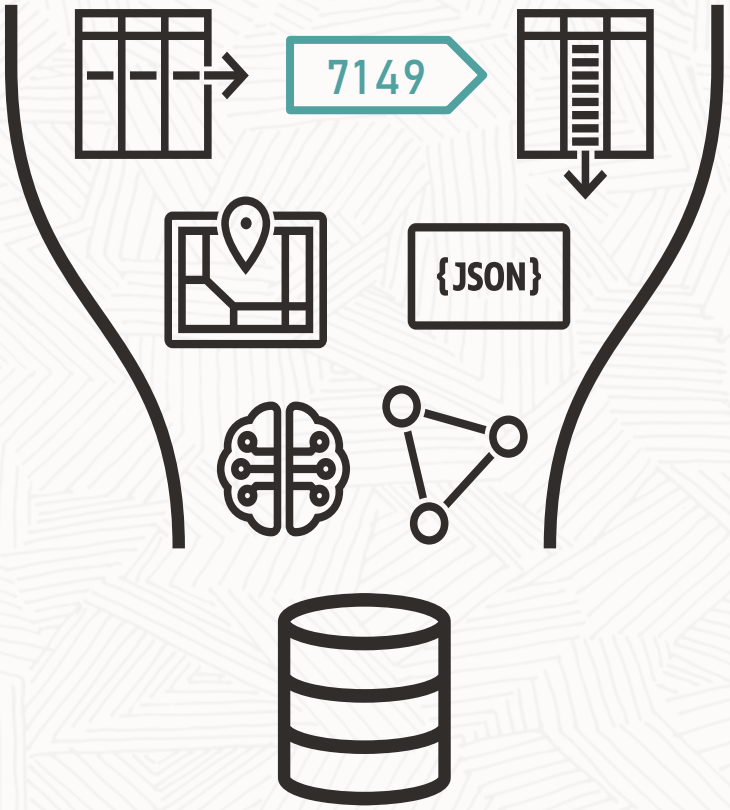
New set of capabilities coming in Oracle Database 23ai

Process vector search and other workloads in 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

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



Converged Database

VECTOR Datatype

New VECTOR datatype (full PL/SQL support)

```
CREATE TABLE my_images (
     id          NUMBER,
     data_image BLOB,
     image_vec  VECTOR(768, FLOAT32));
```

Optional dimension count

Optional dimension format

Dimension format can be:
`INT8`, `FLOAT32`, and `FLOAT64`

Future support for `BIT`, `FLOAT16`, `BFLOAT16`

Simpler VECTOR specification

```
CREATE TABLE my_images (
     id          NUMBER,
     data_image BLOB,
     image_vec  VECTOR);
```

Why is this useful?

You can embed your data with newer ML embedding models as AI technology evolves, but your schema can stay the same, and applications don't need to be rewritten

VECTOR Operations

Insert

`TO_VECTOR()` converts a string representing an array of vector dimensions into a native VECTOR

```
CREATE TABLE my_images (
     id        NUMBER,
     image_vec VECTOR(3, FLOAT32));

INSERT INTO my_images VALUES
(1, TO_VECTOR('[1.1, 2.2, 3.3]'));
```

Fetch

`FROM_VECTOR()` converts a vector into a CLOB or VARCHAR2 – default behavior for pre-23ai clients

```
SELECT image_vec FROM my_images;

SELECT FROM_VECTOR(image_vec)
FROM my_images;

'[1.1, 2.2, 3.3]'
```

Several 23ai client drivers (Python, Node.js, OCI, JDBC, ODP.NET) have **native VECTOR support** and hence, can insert and fetch Vectors directly without string conversions

VECTOR Distance Function

The key operation is vector distance computation to gauge similarity

```
VECTOR_DISTANCE(VECTOR1, VECTOR2, <optional distance metric>)
```

Different embedding models can use different distance metrics, but the basic concept remains the same:

- The Distance between two vectors is smaller for entities that are more similar

Distance functions supported in 23ai are:

COSINE (Default), EUCLIDEAN, EUCLIDEAN_SQUARED, HAMMING, MANHATTAN, DOT

Allows queries that combine AI vector search with business data about customers and products

Combines customer data, product data, and AI search in a few lines of SQL!

A single integrated solution, all data is fully consistent

Find houses that are similar to this picture and match the customer's preferred city and budget

```
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;
```

New VECTOR_EMBEDDING() function to generate vectors

Completeness: Many customers want to be able to generate vectors within the database

Oracle Database supports the Open Neural Net Exchange (ONNX) framework to import models

The VECTOR_EMBEDDING() function can then generate vectors for unstructured data using the imported model

```
// import text model for documents
DBMS_VECTOR.load_onnx_model(
    model_name => "All-MiniLM-L6-v2",
    model_data => "All-MiniLM-L6-v2.onnx
    ...
);
```

```
// generate vectors from support incidents
SELECT
VECTOR_EMBEDDING(All-MiniLM-L6-v2 USING incident_text)
FROM Support_incidents;
```

AI Vector Search can augment Generative AI by retrieving additional, often private, content needed to answer questions more accurately

Called: Retrieval Augmented Generation (RAG)

Secure RAG

Oracle Maximum Security Architecture

Security in RAG is crucial, particularly in enterprise environments where access to data must be controlled and regulated

Oracle has numerous **Advanced Security** features to ensure that only authorized users can access specific data during RAG retrievals

Oracle's **Virtual Private Database** (VPD) features allows fine-grained access control at the row level within a database

- VPD Policies can be created to specify access control rules based on user roles or privileges
- VPD Predicates are enforced at query time to ensure that users can only access authorized data
- VPD works transparently with SQL queries making it ideal for securing RAG applications without requiring significant changes to the underlying database

Oracle AI Vector Search is Fully Integrated

Seamless Integration with core database features for enterprise-grade performance and reliability

RAC

Sharding

Partitioning

Exadata

Transactions

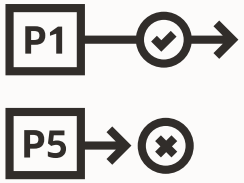
Parallel Execution

Analytics

Security

What’s new in Oracle Database 23ai for DBAs / IT Operators?

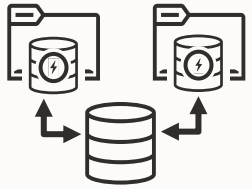
Oracle Database 23ai Mission Critical Apps Enhancements



Priority Transactions

Automatically prioritizes high-priority transactions over low-priority transactions

Low-priority transactions that block high-priority transactions will be automatically aborted

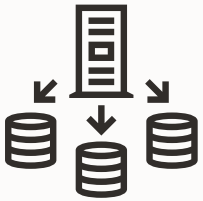


True Cache

A (nearly) disk-less Oracle database instance that is deployed as a cache

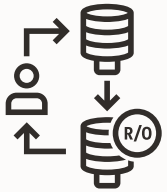
Unlike conventional mid-tier caches such as Redis, data in True Cache is automatically updated

ANY SQL Query can be transparently directed to the cache instead of the database



Active-Active Globally Distributed Database

Database sharding with Raft replication supports applications that require low latency and high availability plus helps meet data sovereignty requirements



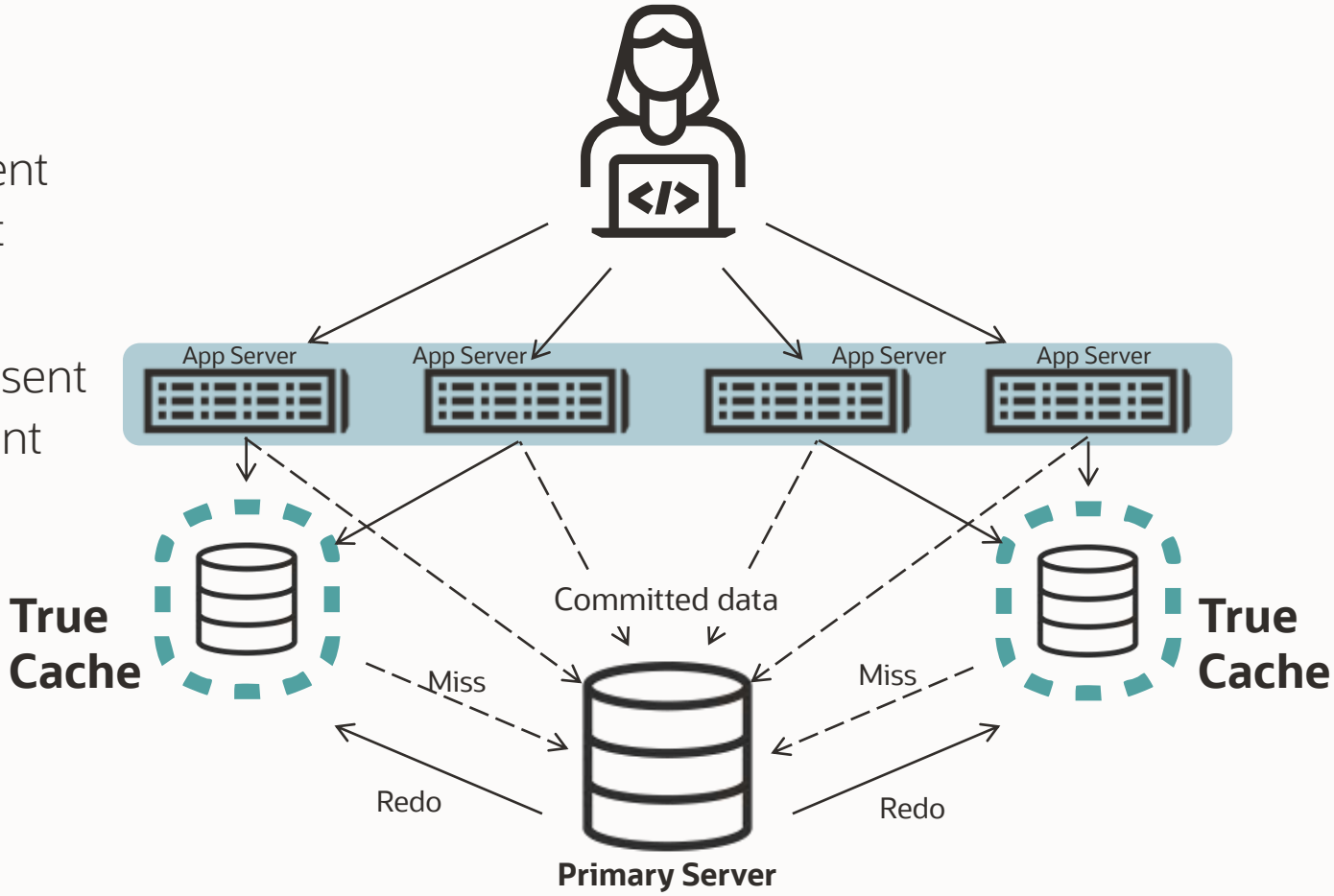
Readable Per-PDB Standby

Per-PDB standby databases can now be opened for read-only workloads

Improving production database performance by offloading resource-intensive backup and reporting operations to standby systems

Oracle Database 23ai True Cache

- Solid lines represent relatively frequent requests
- Dotted lines represent relatively infrequent requests



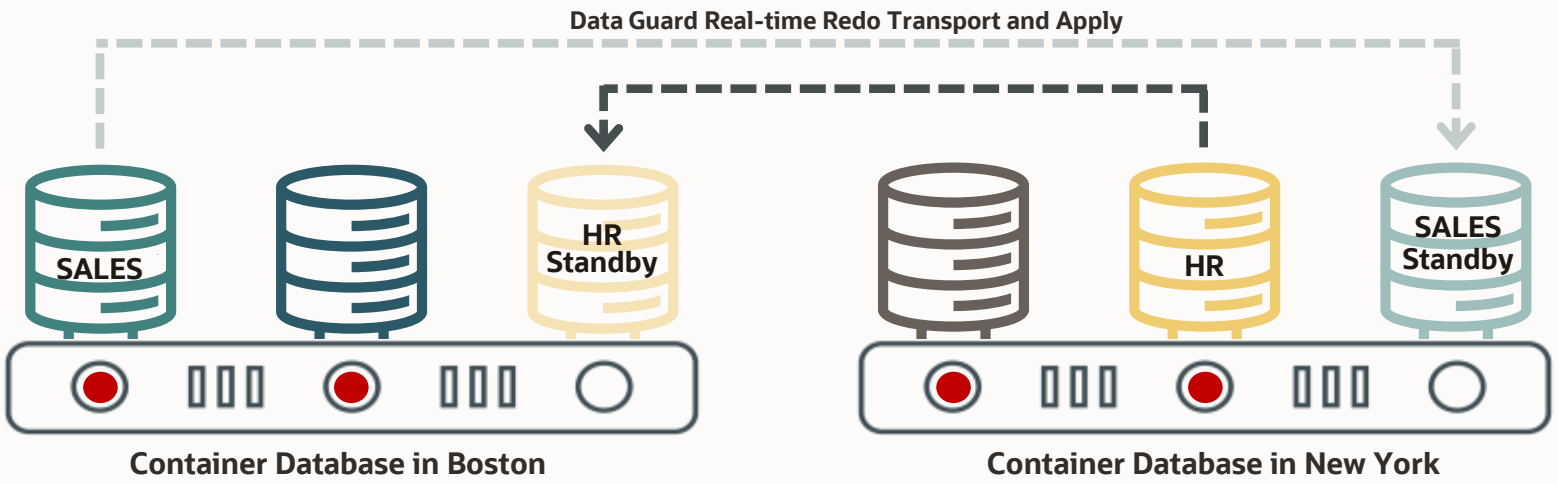
App connect to True Cache and perform SQL queries

True Cache is an in-memory, consistent, and automatically managed full SQL cache

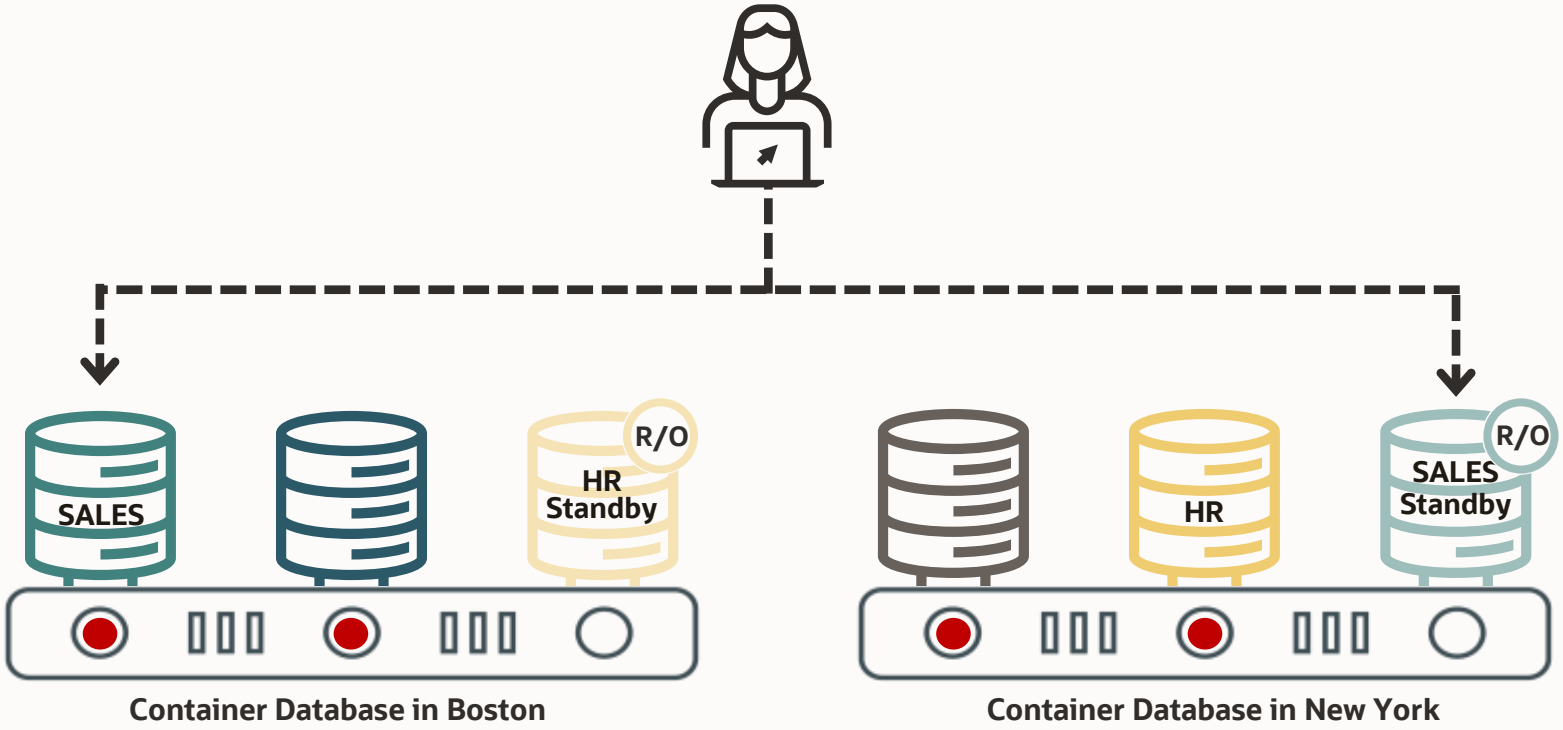
Oracle Active Data Guard for Pluggable Databases

Provides Data Guard disaster recovery, and availability features at the pluggable database (PDB) level

- Each PDB can switchover or failover independently to the remote CDB
- No need to fail over all the PDBs when an issue occurs that only impacts one PDB
- Redo is shipped and applied in real-time



Oracle Readable PDB Standbys



Per-PDB standby databases can now be opened for read-only workloads

Improves production database performance by offloading resource-intensive backup and reporting operations to standby systems

Oracle Database 23ai Security Enhancements

In-Database Firewall

An easy-to-use firewall solution, with minimal perf and operational overhead

Built-in to ensure it cannot be bypassed

Protection against attacks by monitoring and blocking “unauthorized SQL” and SQL injection attacks

Read-Only Users

Users may be created as, or altered to, `READ ONLY` status (default `READ WRITE`)

```
ALTER USER joe
READ ONLY;
```

Read-only users can not insert or update data, nor can they create database objects

Developer Role

It’s complex to grant all the privileges developers need to create, debug, etc.

Now it’s simple using the new DB_DEVELOPER_ROLE :

```
GRANT DB_DEVELOPER_ROLE
TO    scott;
```

Schema Privileges

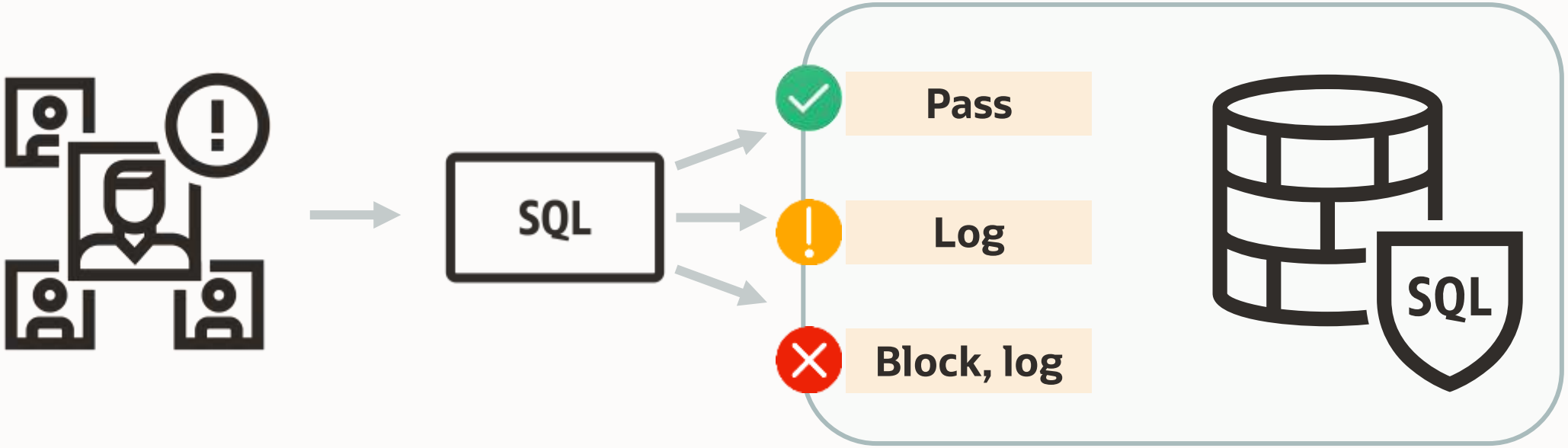
Managing the privileges on all the tables, views, and procedures used by an app can be tricky

Now this is simple using `GRANT` on a schema

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

Oracle In-Database SQL Firewall

Oracle SQL Firewall offers protection against common database attacks by monitoring and blocking **“unauthorized SQL”** and SQL injection attacks



SQL Firewall is **built into** the database, ensuring that it **cannot be bypassed**
No extra hops, management, installation, patching, etc. of a mid-tier database firewall

Oracle Database 23ai Manageability and Availability Enhancements

Shrink Tablespace

A simple way to reclaim unused or free space in a tablespace

Optimizes the storage of big file tablespaces by moving objects to the datafile head, and then resizing the datafile by removing the tail

Real-time SQL Plan Management

Automatically repairs SQL performance regressions

The optimizer detects a plan regression and tries to find a previous plan with better performance

If an alternative plan is found to perform better, a SQL plan baseline is automatically created and that plan will be used

Rolling Patching for Complex Changes

Enables non-rolling patches to be applied online in stages

Phase 1:
The patch is applied to all instances but not enabled

Phase 2:
The patch is enabled via a SQL command

Enhanced Error Messages & Logging

Improved error messages that provide useful problem diagnosis in context and suggest actionable solutions. Easily searchable error message portal.

New attention log that highlights issues requiring prompt remediation

Oracle Database 23ai – The Next Long Term Support Release

Data Use Case Domains

Boolean Datatype

Oracle Database

23ai

Bring AI to your data

Readable PDB Standby

Property Graphs

Real-time SQL Plan Management

JSON Schema

Microservice Sagas

JSON / Relational Duality

AI Vector Search

True Cache

SQL Firewall

Priority Transactions

Globally Distributed Database

Rolling Patching

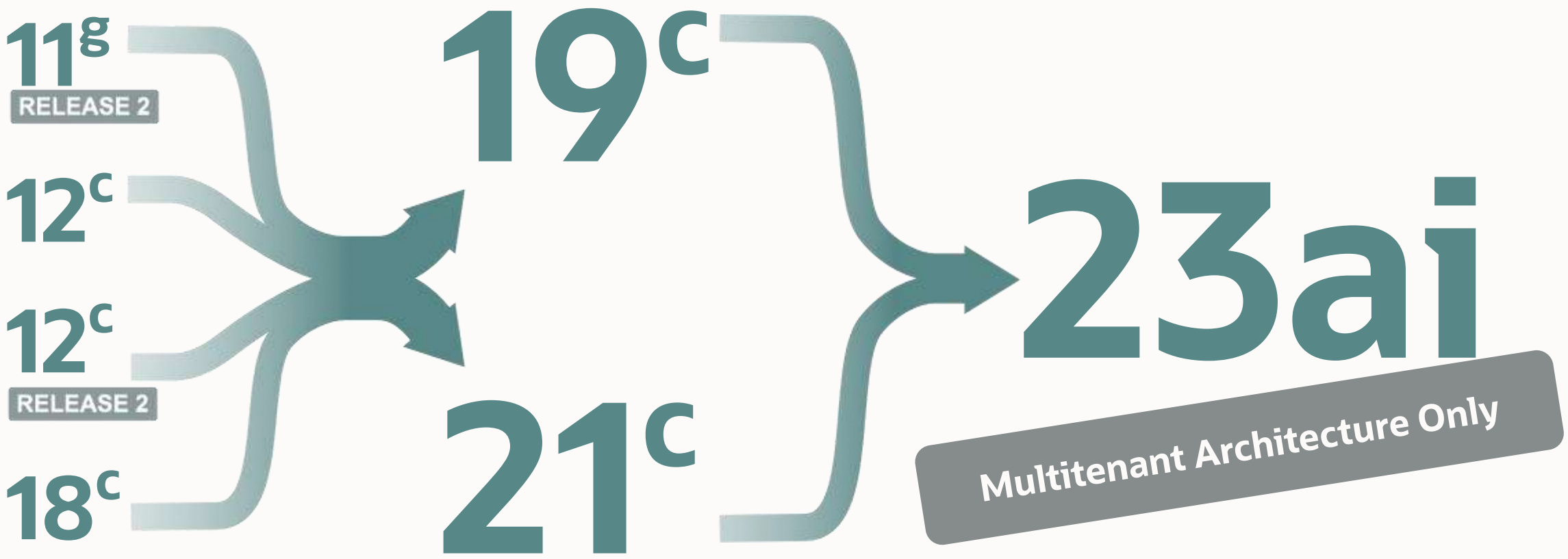
JS Stored Procedures

Developer Role

Shrink Tablespace

Schema Privileges

Upgrade Path to Oracle Database 23ai



Summary: Oracle Database Strategy & Roadmap

Focused on increasing productivity for Developers, DBAs, and Data Analysts

Converged Oracle Database

Simplifies development of operational and analytical applications today, and for tomorrow

Oracle Exadata

Delivers extreme performance and availability for all data workloads

Oracle Autonomous Database

Delivers all the benefits of converged Oracle Database in the cloud, on-premises and in multicloud configurations

Generative AI

Makes it easy to generate and run modern apps and analytics for all use cases at any scale

Oracle Database 23ai

The next Long Term Release that continues to deliver converged database innovations for both cloud and on-premises

Hands-on-Lab
Oracle Database 23ai New Features

Oracle LiveLabs

https://livelabs.oracle.com/pls/apex/r/dbpm/livelabs/view-workshop?wid=3950

Thanks for joining this session

Checkout Oracle Database 23ai
http://www.oracle.com/database/23ai