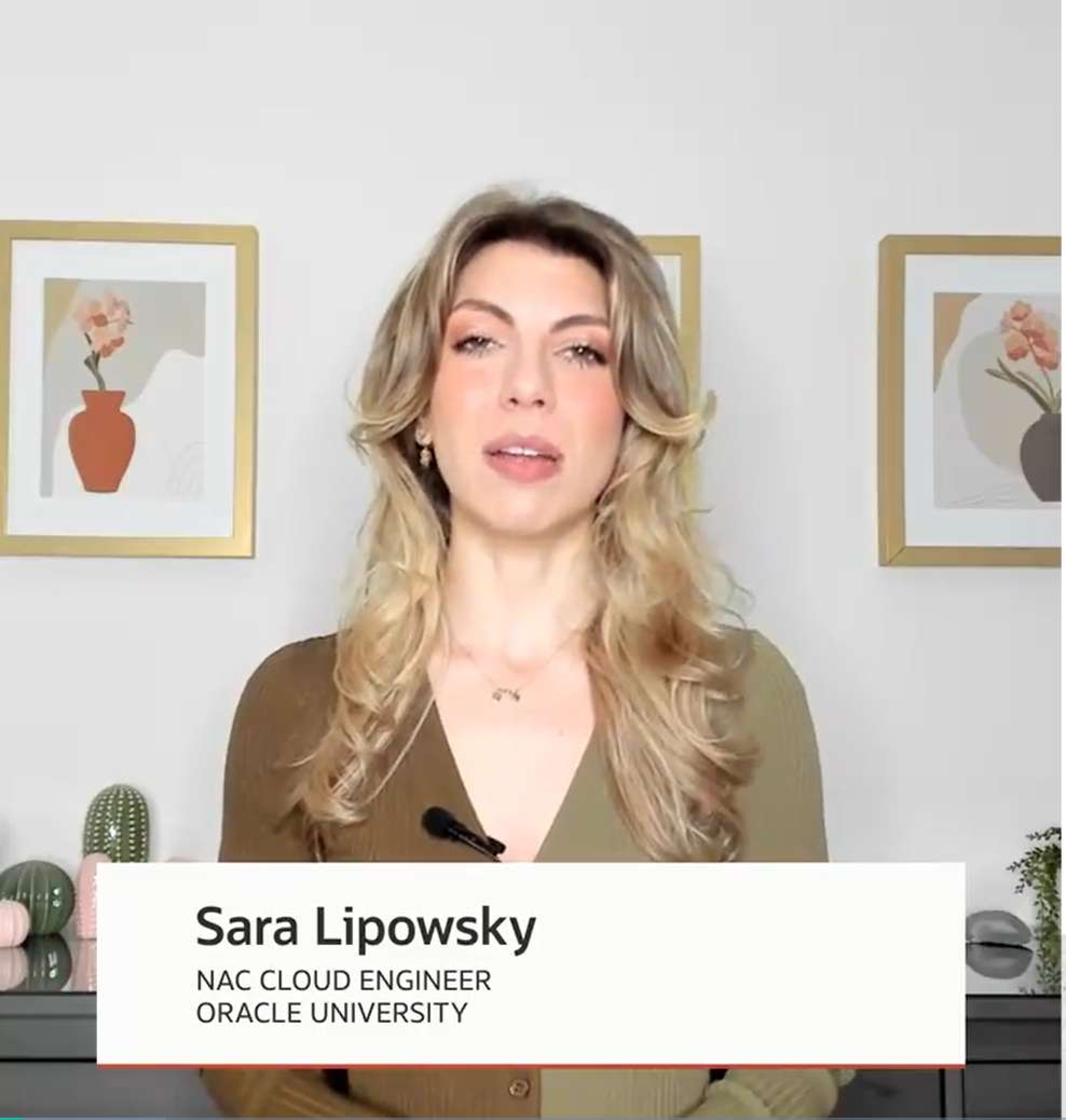

Sara Lipowsky
NAC CLOUD ENGINEER
ORACLE UNIVERSITY

ORACLE
University

Oracle Cloud Infrastructure

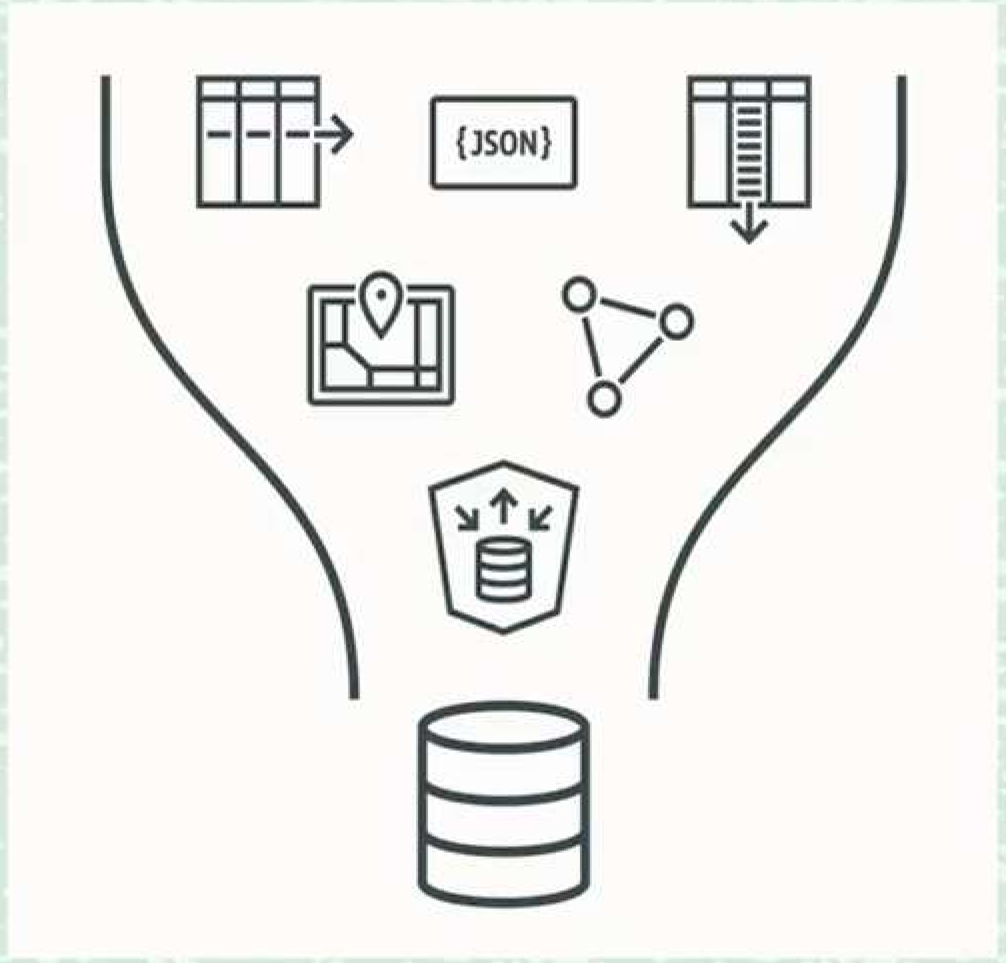
Oracle Autonomous JSON Database

An Overview

0:08 / 12:21 1x

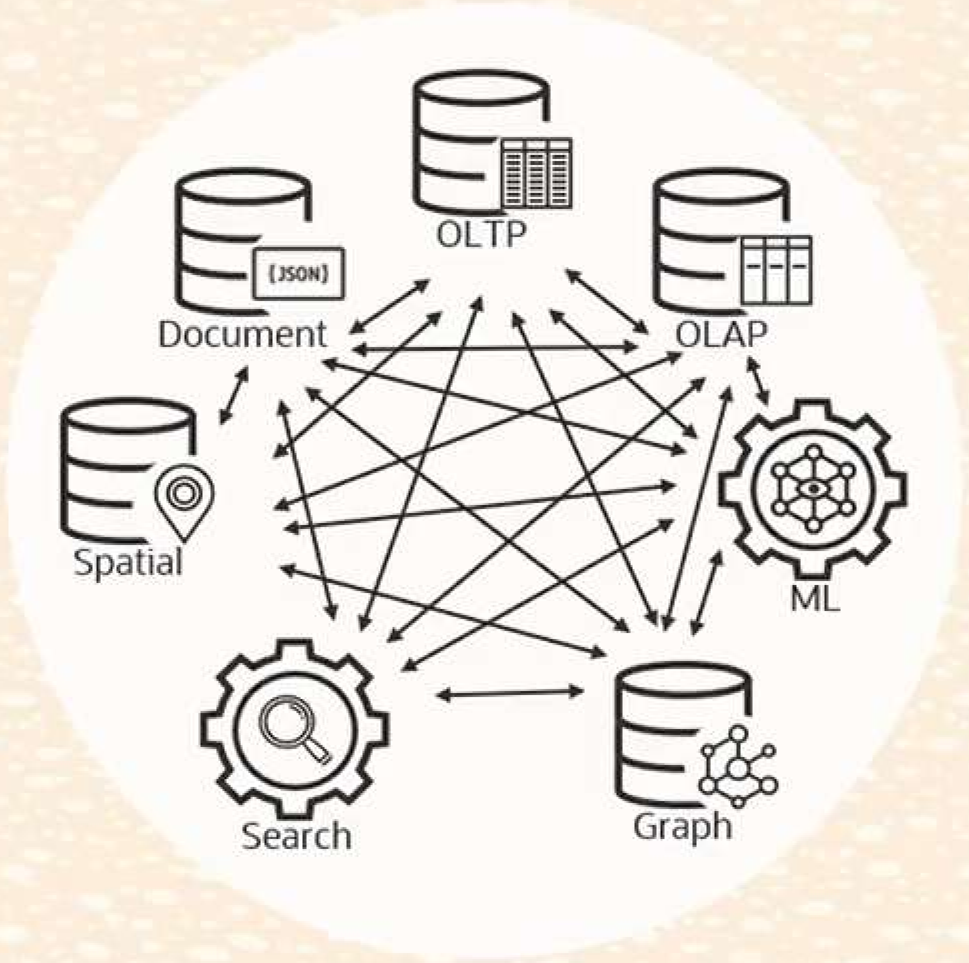
Converged Database

Converged Database Architecture



for **any** data type or workload

Single-purpose databases



for each data type and workload

Multiple security models, query languages, skills, licenses, and so on

Why JSON?

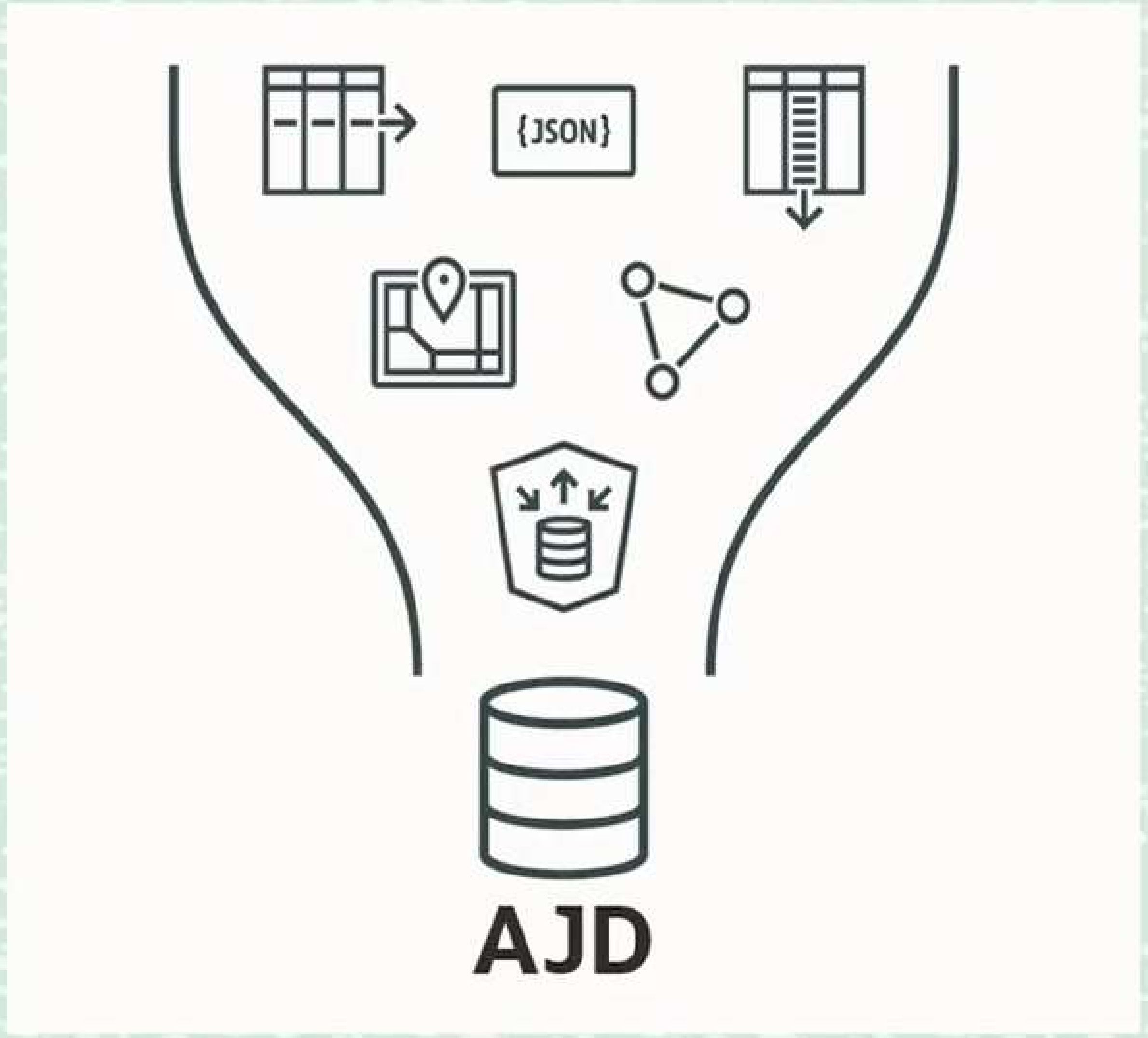
- Schema-flexible
  - Applications can evolve to store new attributes without modifying data definitions.
- Easily consumed by applications
  - Nested structures
  - Maps to application objects
  - Read/write without joins
- Good common format
  - Supported by most programming languages
  - Human readable
  - Simplifies data exchange across app, servers, and database tiers

```
{
   "name"   : "Thomas Anderson",
   "job"    : "Programmer",
   "addresses" : [
      {
        "street" : "123 Main",
        "city" : "Santa Cruz",
        "zip" : 95041
      }
   ]
}
```

Autonomous JSON Database

- Low-latency, scalable, JSON storage
- MongoDB APIs or SQL
- No database management
- Always-free service
- All the features of the Autonomous Database
  - Limited to **20 GB for non-JSON data**

The converged database
as a **managed cloud service**



Autonomous JSON Database

Powered by Autonomous Database, a family of Cloud Services introduced in 2018

Self-Driving

Automates all database and infrastructure management, monitoring, tuning

Self-Securing

Protects from both external attacks and malicious internal users

Self-Repairing

Protects from all down time including planned maintenance

Spend Less, Reduce Risk, Innovate More

Autonomous JSON Database

All the benefits of a "one-trick" document database

Elastic compute and storage

Single-digit latency reads and writes

Highly available

Low price, always-free tier

Autonomous Database Workloads

OLTP+ JSON workloads

Transaction Processing (ATP)

Preconfigured for row format, indexes, and data caching to accelerate **transaction processing** and **mixed workloads**

- **Converged database** with no data storage limits (relational or JSON), JSON data fully supported
- Includes **Oracle Database API for MongoDB**

JSON-centric workloads

JSON (AJD)

Price optimized for transactions and analytics on **JSON data**

- Same features as ATP but **75% lower price**
- **Unlimited JSON Collections** + up to 20GB non-JSON data
- Includes **Oracle Database API for MongoDB**

Single-click upgrade to ATP

Analytics workloads including JSON data

Data Warehousing (ADW)

Preconfigured for columnar format, partitioning, and large joins to accelerate **analytics, data warehouse**, and **data lakehouse**

- Same features as ATP, optimized for analytics
- JSON data fully supported

Autonomous JSON Database: Pricing and Performance

Autonomous JSON Database pricing:

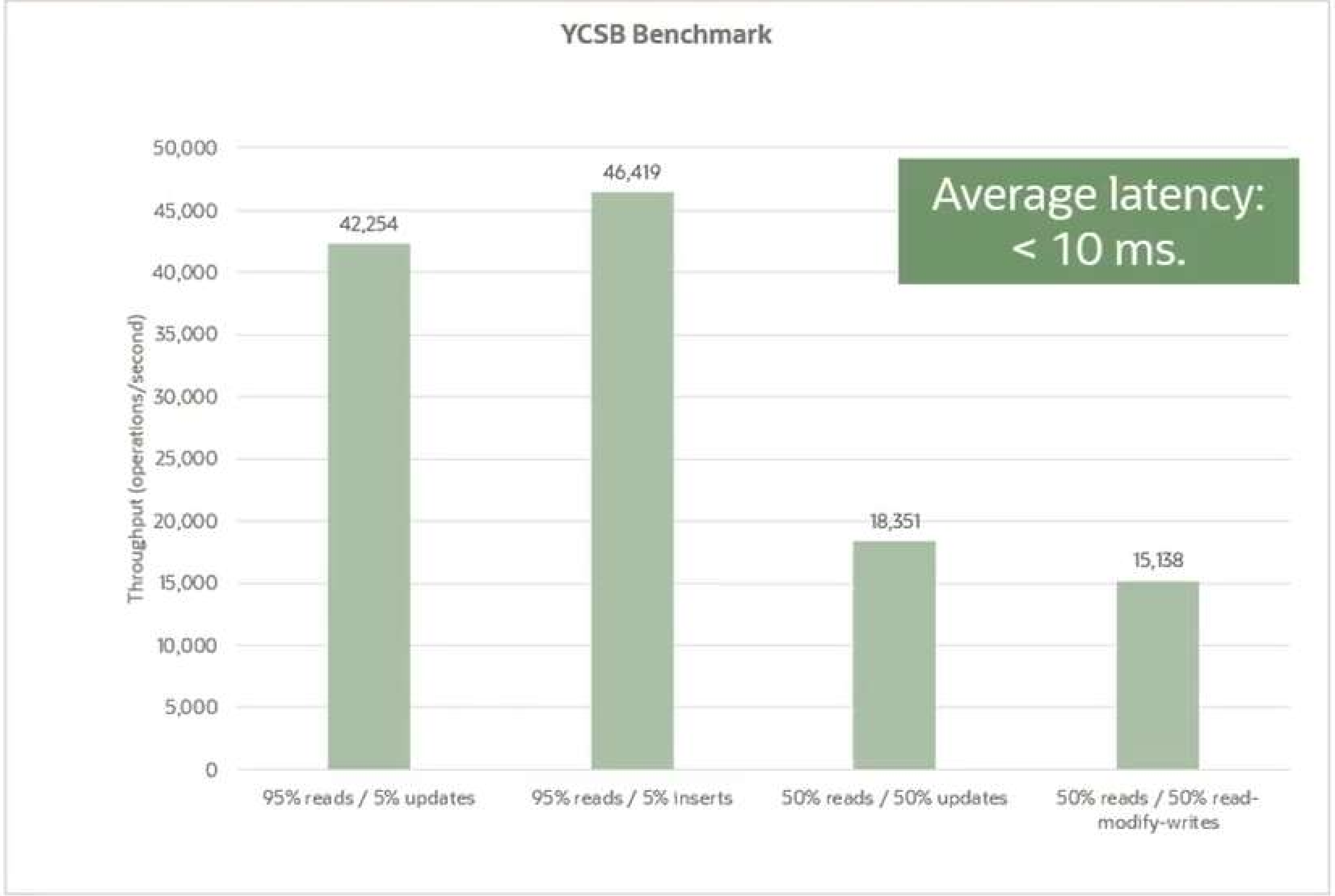
- $0.1344 per vCPU-hour, available in 2 vCPU increments ($240/month for 2 vCPUs)
-

	Autonomous JSON Database	MongoDB Atlas
Configuration	16 vCPUs 1 TB storage	M60 on AWS 16 vCPU 320 GB storage
Price	**$2.74 / hour**	**$3.95 / hour***

PLUS: Autonomous JSON Database is auto-scaling; not limited to fixed shapes.

* https://www.mongodb.com/pricing
2 vCPUs = 1 OCPU, billing is done on a per-second basis based on OCPUs

Autonomous JSON Database provides consistent performance across different workloads.**



** Based on Autonomous JSON Database with 8 OCPU running in San Jose region

Classic Relational Model

- There are multiple "flat" tables that are related.
- A **table** contains **rows.**
- A **schema** contains tables.
- Rows are structured (the 'S' in SQL).
- Data is accessed using SQL.
- Related entities are joined.

Normalized Tables

id	name	job
123	Anderson	Programmer
345	Smith	Agent

Accessed with SQL

```
select e.id, e.name
from employee e
where e.job = Programmer'
```

Oracle Database API for MongoDB

- **Data model:** JSON collections, not tables
- **Developers keep their skills** and continue to use MongoDB's tool, drivers, and so on.
- **Easy migrations** of MongoDB workloads to Oracle
- **Enables SQL:**
  - More and faster analytical capabilities, machine learning
  - Query JSON alongside other data models: relational, XML, spatial, and so on
  - Expose relational data, reports, query results as MongoDB collections

Document Collections

- A **document** is a JSON value:
  - Structure is flexible
  - Have a unique **key** (_id)
- A **collection** contains documents:
  - Supports insert, get, update, filter
- A **database** contains collections.
- No SQL is required.

MongoDB Collection API

```
use admin;

db.createCollection("employee");

db.employee.insertOne(
{
   "_id" : 123,
   "name" : "Thomas Anderson",
   "job"  : "Programmer"
});

db.employee.find(
  {"job"  : "Programmer"}
);
```

Oracle API for MongoDB

Database => **Schema**

Collections created in database “admin” will be in the “ADMIN” schema.

```
use admin;

db.createCollection("employees");

db.employee.insertOne(
{
   "_id" : 123,
   "name" : "Thomas Anderson",
   "job"  : "Programmer"
});

db.employee.find(
  {"job"  : "Programmer"}
);
```

Oracle API for MongoDB

Collection => **Table**

Collections are an abstraction or a view of a table with a single JSON column.

```
CREATE TABLE employee (
    ID VARCHAR2,
    DATA JSON
)
```

```
use admin;

db.createCollection("employees");

db.employee.insertOne(
{
   "_id" : 123,
   "name" : "Thomas Anderson",
   "job"  : "Programmer"
});

db.employee.find(
  {"job"  : "Programmer"}
);
```

Oracle API for MongoDB

Document => Row

Inserting a document into a collection inserts into the backing table.

```
INSERT INTO employees (data)
VALUES (:1);
```

```
use admin;

db.createCollection("employee");

db.employee.insertOne(
{
   "_id" : 123,
   "name" : "Thomas Anderson",
   "job"  : "Programmer"
});

db.employee.find(
  {"job"  : "Programmer"}
);
```

Oracle API for MongoDB

Filter => Query

Filter expressions are executed as SQL over the backing table; fully utilizes core Oracle Database features such as indexing, cost-based optimization, and so on.

```
SELECT data
FROM employee e
WHERE e.data.job = 'Programmer'
```

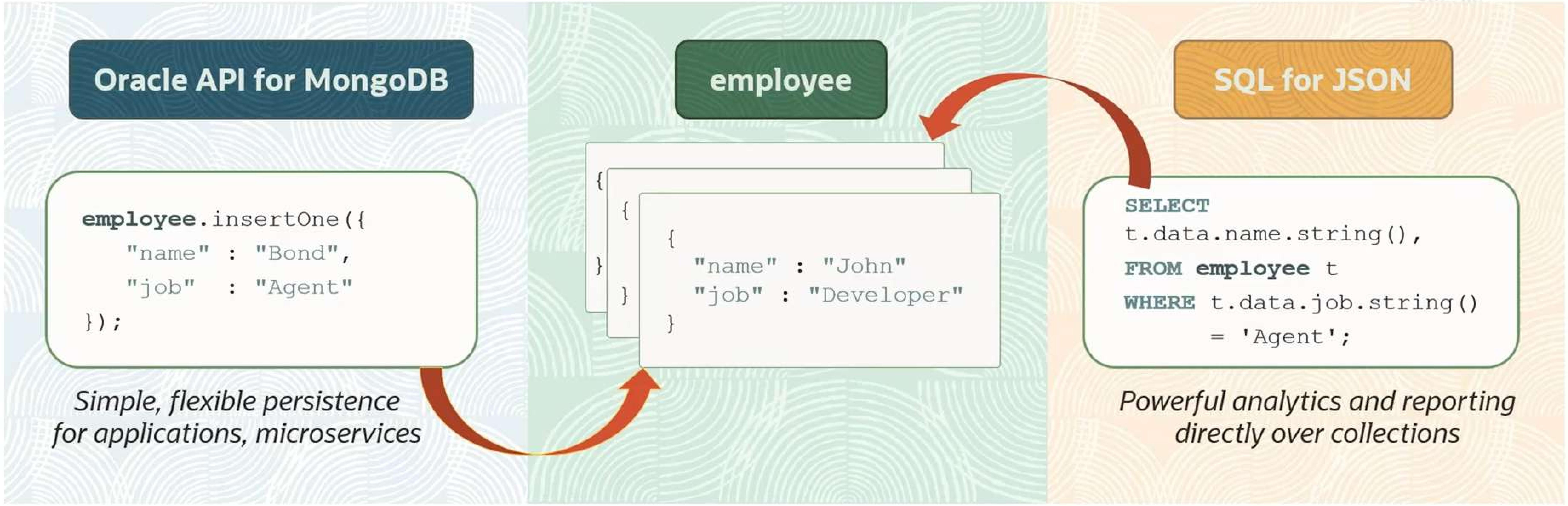
```
use admin;

db.createCollection("employee");

db.employee.insertOne(
{
   "_id" : 123,
   "name" : "Thomas Anderson",
   "job"  : "Programmer"
});

db.employee.find(
  {"job"  : "Programmer"}
);
```

SQL Only When It Is Needed



```
employee.insertOne({
   "name" : "Bond",
   "job"  : "Agent"
});
```

```
{
  "name" : "John"
  "job" : "Developer"
}
```

```
SELECT
t.data.name.string(),
FROM employee t
WHERE t.data.job.string()
      = 'Agent';
```

Simple, flexible persistence for applications, microservices

Powerful analytics and reporting directly over collections

Autonomous Database Is MongoDB Compatible* and More

Run MongoDB workload on Autonomous Database with Oracle Database API for MongoDB

App Developers

Analysts | Data Scientists

APPLICATION Development ⟷ **ANALYTICS** Development

Dev Needs

- Keep using the **same MongoDB app dev toolkit**:
  - — Skills and knowledge
  - — Favorite dev tools
  - — Drivers, libraries, and frameworks
- Develop new MongoDB compatible services that support more workload types, including SQL for analytics

- Keep using **SQL and familiar BI / data science tools**:
  - — Use SQL for analytical queries and reports over JSON data
  - — Join JSON with other non-JSON data (relational, spatial, XML, Graph)
  - — Keep using existing SQL-based tools and dashboard, reporting pipelines

Solution

Autonomous Database

- Oracle Database API for MongoDB
- Compass, Mongo Shell, mongoimport, and so on

- APEX, SQL Developer, and other third-party BI tools
- OML Notebook, AutoML UI, and other third-party IDEs

Use Oracle Autonomous JSON Database