ORACLE

Building applications using JSON Collections and the Oracle API for MongoDB

Developer Tech Days

Your Name
Title

3 Key Points to pay attention to

1 Store, use, and manage collections, JSON documents, and relational data in a single converged database. Unified management, security, consistency model

2 Comprehensive document-store APIs and language support for Java, Python, node.js, and others, supporting MongoDB and Oracle SODA. No knowledge of Oracle or SQL required

3 Leverage existing MongoDB skills and easily move your applications and data to a single converged database and work with your data in whole new ways

How are you working with JSON data today?

1. Can you describe any challenges or limitations you've encountered with JSON, and how have you addressed them?
2. How do you version and manage changes to JSON data structures over time, especially when multiple systems are involved?
3. How do you ensure data integrity and security when working with JSON data, especially in sensitive business contexts?

Why store JSON?

```
{
  "movie_id" : 1652,
  "title" : "Iron Man 2",
  "date" : "2010-05-07",
  "cast" : [
    "Robert Downey Jr.",
    "Larry Ellison",
    ..
  ]
}
```

```
class Movie {

  int movie_id;
  String title;
  LocalTime date;
  List<String> cast;

  Movie() {
    ...
}
```

```
c = customers.find(
            {lastName:Smith});
```

versus

```
stmt = "
 SELECT id, data
 FROM customers
 WHRE data.lastName='Smith'";
rs = execute(stmt);
```

Schema-flexible

- No upfront schema design
- Application-controlled schema
- Simple data model

Less Impedance Mismatch

- Maps to application objects
- Supports nested structures
- Read/write without joins

Easy Data Access

- REST
- NoSQL (document store) API
- Easy CRUD operations
- Faster to code, easier to read

JSON in the Oracle database
About JSON

- JSON's built-in data type has maximum size of 32 megabytes JSON's code is 119 ☺ (there are 26 different built-in data types in Oracle 23ai)
- Oracle's native binary JSON format called OSON (Oracle's optimized binary JSON format) is the Oracle extension of the JSON format
  - Adding scalar types (date and double) which are not part of the JSON standard
  - The SQL data type JSON uses format OSON has better query performance
    - Textual JSON data no longer needs to be parsed
- Set init.ora parameter compatible to at least 20 when using the JSON data type
- JSON Data Type Support
  - varchar2, CLOB, BLOB

JSON in the Oracle database
About JSON

- Oracle also provides a family of Simple Oracle Document Access(SODA) APIs for access to JSON data stored in the database
  - SODA for Java —Java classes that represent database, collection, and document
  - SODA for REST —SODA operations as representational state transfer (REST) requests, using any language capable of making HTTP calls

ADB: alter table <table_name> modify lob(<lob_column>) (retention min <time_in_secs>);

Oracle Database for JSON storage

Converged Database

- NoSQL-style document storage
- High-concurrency, low-latency, interactive applications
- Analytics and reporting

Access options

- SQL
- REST
- MongoDB API

Autonomous JSON Database

Converged Database PLUS

- Managed cloud service
- Automatic scaling, backups, and management
- Faster and cheaper than MongoDB Atlas

- Always-free service Oracle Autonomous JSON is FedRAMP high
  - MongoDB is only FedRAMP
- Moderate
  - Federal Risk and Authorization Management Program

Autonomous JSON Database

Elastic compute and storage

Single-digit latency reads and writes

Highly available

Low-price, always-free tier

Autonomous JSON Database
More than a NoSQL document store

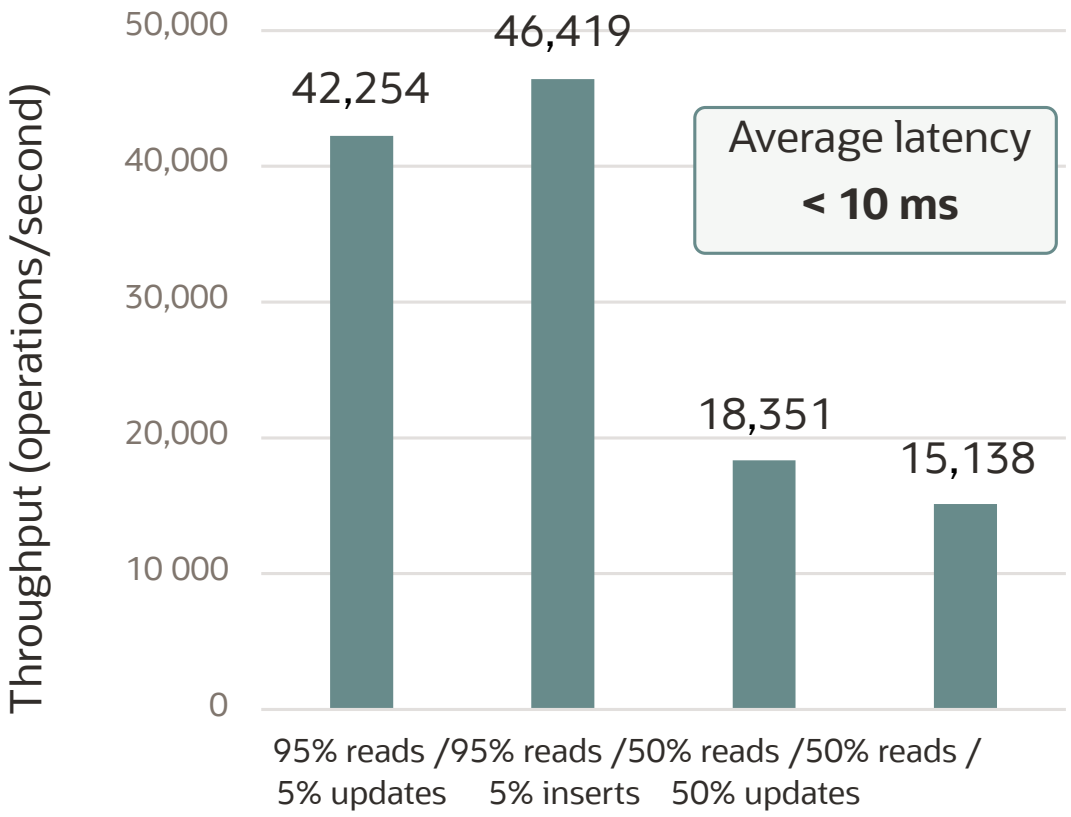
Native and comprehensive SQL support

- ANSI SQL/JSON
- Joins
- Advanced analytics
- Full-text search

Powered by the Oracle Database

- ACID compliant transactions
- Secure by default
- Mission-critical

YCSB NoSQL Benchmark



** Autonomous JSON Database with 8 OCPU running in San Jose region, Mongo API

Leverage your MongoDB skills and work with your data in whole new ways

JSON Columns and SQL/JSON

Enabling technology for Oracle's native document store support and gateway to enrich any MongoDB application

Relational Model

- A **schema** contains **tables**
- A **table** contains **rows**
- A table is **flat**
- Rows are **structured**
- Data is accessed with **SQL**
- Related rows are joined

Flat, structured tables

id	title	date
123	Iron Man	2010-05-07
345	Thor	2022-07-08

Accessed with SQL

```
SELECT m.title, m.date
FROM   movies m m.id=
WHERE  123
```

Relational Model + JSON Columns

- **Schema-flexible** JSON stored within a **structured** column
- SQL **extended** to process JSON column values
- Stored using query-efficient **OSON** binary format

```
CREATE TABLE movies (data JSON);

INSERT INTO movies VALUES
       ( JSON{
                 'id'     : 123,
                 'title' : 'Iron Man'
               }
        );
```

JSON Collections and the MongoDB API

A native MongoDB API compatible document store

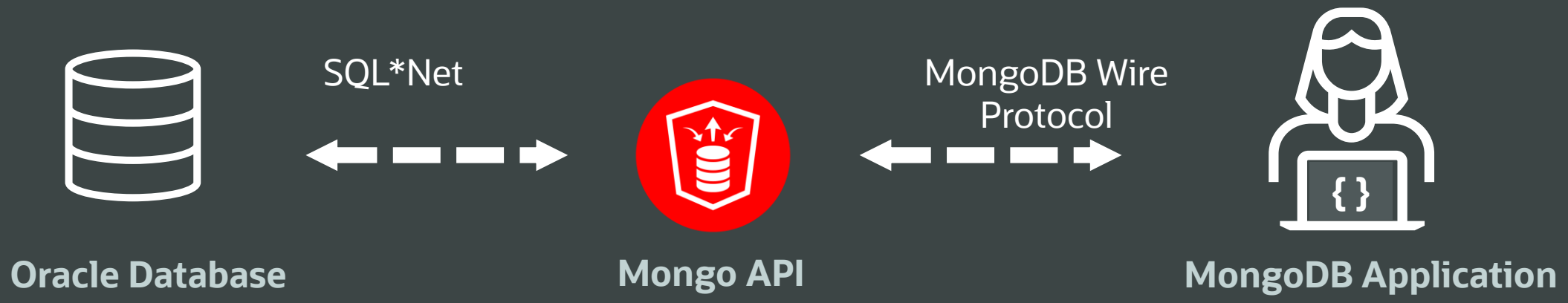
Oracle Database API for MongoDB

Connect MongoDB client drivers and tools to Oracle Database

MongoDB does not have tables – it stores collections of JSON documents

- Transparency simplifies migrations from MongoDB to Oracle

MongoDB developers keep using the same skills, tools, and frameworks



Oracle Database API for MongoDB

Connect MongoDB client drivers and tools to Oracle Database

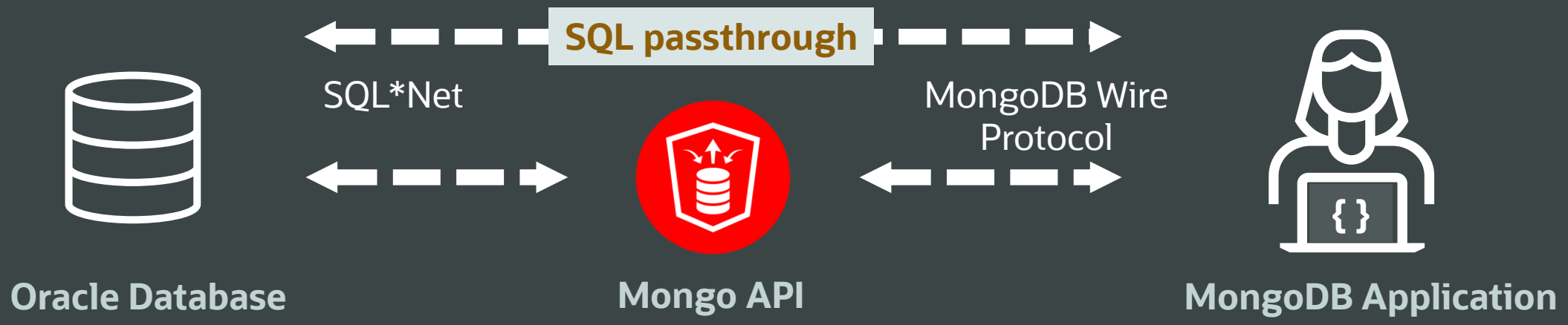
MongoDB does not have tables – it stores collections of JSON documents

- Transparency simplifies migrations from MongoDB to Oracle

MongoDB developers keep using the same skills, tools, and frameworks

Enhance applications with SQL passthrough

- **Statements and data**



JSON Collections

A **document** is a JSON value

Structure is flexible

A **collection** contains documents

Supports insert, get, update, filter

A **database** contains collections

Access data programmatically – "No SQL"

MongoDB Example

```
MongoClient  mongoClient  =  MongoClients.create(connString);
MongoDatabase database = mongoClient.getDatabase("admin");

MongoCollection<Document> coll =
  database.createCollection("movies");

Document movie = Document.parse(json);
coll.insertOne(movie);

Bsonfilter  =eq("title",  "Iron Man");
MongoCursor<Document> cursor = coll.find(filter).cursor();

Document doc=cursor.next();
```

JSON Collections

Database => Schema

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

MongoDB Example

```
MongoClient  mongoClient  =  MongoClients.create(connString);
MongoDatabase database = mongoClient.getDatabase("admin");

MongoCollection<Document> coll =
  database.createCollection("movies");

Document movie = Document.parse(json);
coll.insertOne(movie);


Bsonfilter  =eq("title",  "Iron Man");
MongoCursor<Document> cursor = coll.find(filter).cursor();

Document doc=cursor.next();
```

JSON Collections

Collection => Table

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

```
create table
movies (
    ID VARCHAR2,
    DATA JSON
);
```

MongoDB Example

```
MongoClient  mongoClient  =  MongoClients.create(connString);
MongoDatabase database = mongoClient.getDatabase("admin");

MongoCollection<Document> coll =
  database.createCollection("movies");

Document movie = Document.parse(json);
coll.insertOne(movie);

Bsonfilter  =eq("title",  "Iron Man");
MongoCursor<Document> cursor = coll.find(filter).cursor();

Document doc=cursor.next();
```

JSON Collections

Document => Row

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

```
insert into movies
    (data)
  values
    (:1)
;
```

MongoDB Example

```
MongoClient  mongoClient  =  MongoClients.create(connString);
MongoDatabase database = mongoClient.getDatabase("admin");

MongoCollection<Document> coll =
  database.createCollection("movies");

Document movie = Document.parse(json);
coll.insertOne(movie);

Bsonfilter  =eq("title",  "Iron Man");
MongoCursor<Document> cursor = coll.find(filter).cursor();

Document doc=cursor.next();
```

JSON Collections

Filter => Query

Filter expressions are executed as SQL over the backing table. Fully utilizes core Oracle Database features such as indexing, cost-based optimization, etc.

```
select data from movies e
where e.data.title =
'Iron Man'
```

MongoDB Example

```
MongoClient  mongoClient  =  MongoClients.create(connString);
MongoDatabase database = mongoClient.getDatabase("admin");

MongoCollection<Document> coll =
  database.createCollection("movies");

Document movie = Document.parse(json);
coll.insertOne(movie);

Bsonfilter  = eq("title", "Iron Man");
MongoCursor<Document> cursor = coll.find(filter).cursor();

Document doc=cursor.next();
```

SQL or Document Store APIs – whenever you need it…

JSON Collections

```
movies.insertOne({
  "_id" : 123,
  "title"  : "Iron Man"
});
```

Simple, flexible persistence for applications, microservices

movies

```
{
  "_id" : 123,
  "title" : "Iron Man"
}
```

SQL/JSON

```
select t.data.title.string()
from movies t
where t.data._id = 123;
```

Powerful analytics and reporting directly over collections

SQL or Document Store APIs – whenever and wherever you need it…

SQL/JSON

```
db.aggregate([{
 $sql :
   `select * from movies`
}]);
```

Transparent SQL

JSON Collections

```
movies.insertOne({
 "_id" : 123,
 "title"  : "Iron Man"
});
```

Simple, flexible persistence for applications, microservices

movies

```
{
  {
    {
     "_id" : 123,
     "title" : "Iron Man"
    }
  }
}
```

SQL/JSON

```
select t.data.title.string()
from movies t
where t.data._id = 123;
```

Powerful analytics and reporting directly over collections

SQL – but only when you need it…

JSON Collections

```
movies.insertOne({
"_id" : 123, "title" :
"Iron Man”
});
```

Simple, flexible persistence for applications, microservices

movies

```
{
  {
    {
      "_id" : 123, "title" :
      "Iron Man"
    }
  }
}
```

SQL/JSON

```
select t.data.title.string()
from movies t
where t.data._id = 123;
```

Powerful analytics and reporting directly over collections

Installing Database API for MongoDB for any Oracle Database
Pre-requisites

With the latest 24.1(even 22.3) release of Oracle REST Data Services (ORDS), the Database API for MongoDB is now available for unmanaged Oracle Database on-premises as well as in the cloud:

- It can be installed against any Oracle Database 23ai
- Any Autonomous Database, including ADB Serverless, ADB-Dedicated and ADB on Exadata Cloud@Customer, which all run Oracle Database 19c Possibly any Oracle Database 19c with a very recent patch set, but for that you need to talk to us …

Installing Database API for MongoDB for any Oracle Database
Pre-requisites

Before you start you should ensure you have the following:

- An Oracle Database 21c or later
- The main instructions will assume this is an on-premise database, ADB cloud databases will be
- described at the end Java version 11 or later Knowledge of your TNS connection details for the database: this can either be Server/Port/Service name, or a TNS alias string Knowledge of an administrative user with 'AS SYSDBA' level access -normally SYS -and the password for that user MongoDB client tools such as Mongo Shell or Compass installed somewhere to test the API

Installing Database API for MongoDB for any Oracle Database

Steps

Customers on Autonomous Database are able to configure their own unmanaged ORDS service with the Database API for MongoDB. Here are the steps for unmanaged ORDS:

Step 1: Download

Step 2: Create Directories

Step 3: Unzip the ORDS download

Step 4: Set Environment Variables

Step 5: Run the ORDS installer

Step 6: Configure ORDS to enable MongoDB API

Step 7: Restart ORDS

Step 8: Configure a database user

Step 9: (Optional) Run Database Actions by opening http://localhost:8080/ords/sql-developer

Step 10: Configure Firewall

3 Key Takeaways

1 Store, use, and manage collections, JSON documents, and relational data in a single converged database. Unified management, security, consistency model

2 Comprehensive document-store APIs and language support for Java, Python, node.js, and others, supporting MongoDB and Oracle SODA. No knowledge of Oracle or SQL required

3 Leverage existing MongoDB skills and easily move your applications and data to a single converged database and work with your data in whole new ways

Let’s dive more into JSON in your app dev environment

1. What specific data formats and structures are utilized in your business applications, and how does JSON fit into this landscape?
2. Can you describe the types of data your business typically needs to exchange or share between different systems or partners?
3. How do you currently handle data transmission and integration between disparate systems or platforms?
4. Are there any scenarios where you need to transmit structured data over networks or store it in a lightweight, text-based format?

Try it for free!

free-oracle.github.io

cloud.oracle.com/free

oracle.com/xe

livelabs.oracle.com

Get Hands On with JSON

livelabs.oracle.com

Store query, and process JSON documents in collections using MongoDB API and SQL/JSON

Use SQL to query, generate and process JSON data

Configure the Mongo API to query or manipulate data in the Oracle Database

Learn the newest SQL Enhancements to work with JSON data

LiveLabs | Search Workshops and Sprints... | Event Code | Sign In

SQL, JSON, and MongoDB API: Unify worlds with Oracle Database 23ai Free

Share | Start

1 hour, 30 minutes

Outline

- Store, query, and process JSON documents in collections using MongoDB API and SQL/JSON
- Use SQL to query, generate, and process JSON data
- Configure the Mongo API to query or manipulate data in the Oracle Database
- Learn the newest SQL Enhancements to work with JSON data

About This Workshop

In this workshop, you will experience Oracle's JSON capabilities using both relational and document-store APIs, namely the Oracle Database API for MongDB. The workshop loosely follows the Moviestreams theme, a series of workshops that demonstrate Oracle converged database capabilities. You will work on our movies library throughout the workshop.

This lab is organized into different topics, each topic consists of multiple steps. After completing this workshop a user has a very good understanding of what JSON features are available in Oracle Database and when to use them. You will work against the same data using both SQL and using the Mongo DB API and will experience why Oracle database is better suited for JSON Development than MongoDB, etc.

You can complete this entire workshop using your web browser. There is no need to install any extra software on your local machine. When writing a real

Prerequisites

- An Oracle Database 23ai Free Developer Release or one running in a LiveLabs environment
- Familiarity with Oracle Database is desirable, but not required
- Familiarity with Mongo API is desirable, but not required
- Some understanding of database terms is helpful

ORACLE

Where To Get More Information

- LiveLabs: Developing with JSON and SODA
- LiveLabs: Using the Database API for MongoDB
- LiveSQL: SQL/JSON features
- Blog: Oracle Database API for MongoDB
- O.com: Autonomous JSON Database
- Documentation: Overview of Oracle Database API for MongoDB
- Documentation: Configure the Oracle Database API for MongoDB
- @bch_t @OracleDatabase
- DW-PM_us@oracle.com

It’s now time for Q&A

Got any questions?

Your Name
email@oracle.com
linkedin.com/in/yourlinkedin