ORACLE
DevLive

Boas práticas para desenvolvimento de aplicações modernas usando JSON e Java

em Oracle Database 23c.

Mario Barduchi
Sr. Database Systems Engineer
Dell Technologies, DCWS US & LATAM
April 03, 2024

Speaker

Mario Barduchi

Sr. Database Systems Engineer
Dell Technologies

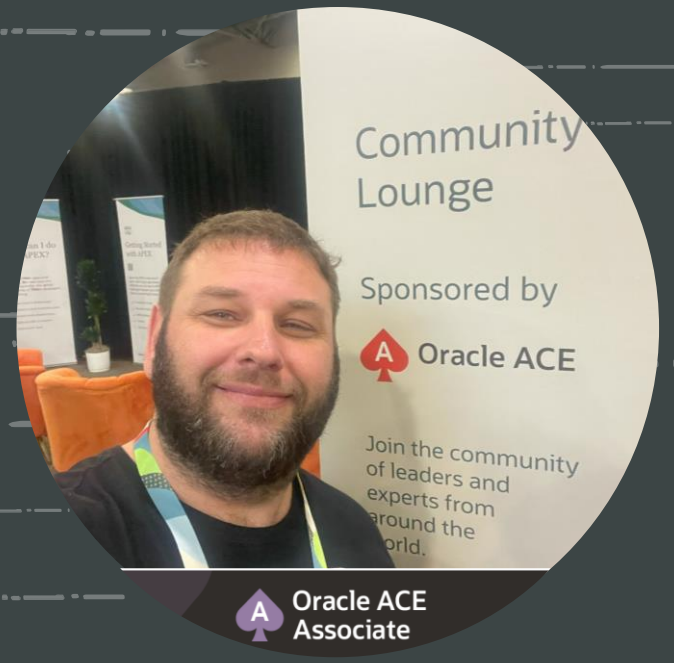
I’ve worked in IT for more than 24 years. I’m Oracle Database Administrator since 2003. MongoDB enthusiast since 2022.

Sr. Database Systems Engineer at Dell Technologies since 2022.

I have a degree in Software Engineering from Metodista University. I completed a MBA in Cloud Architecture and Engineering at FIAP and a postgraduate degree in Oracle Database Administration also at FIAP.

I’ve been an Oracle ACE Associate since 2023. I’m the coordinator and organizer of DBA Brasil Data & Cloud and Regional Coordinator of GUOB.

I'm married to Vanessa, father to Luna “The Cat” DBA and I love to travel and tattoos !!!



Mario Barduchi

Sr. Database Systems Engineer
Dell Technologies

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.

Oracle ACE

400+ technical experts helping peers globally

The **Oracle ACE Program** recognizes and rewards community members for their technical and community contributions to the Oracle community

3 membership tiers

Oracle ACE Director

Oracle ACE Pro

Oracle ACE Associate

For more details on Oracle ACE Program:
ace.oracle.com

Oracle ACE

Nominate

yourself or someone you know:

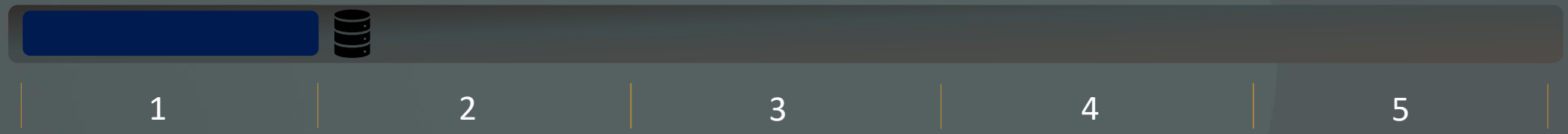
ace.oracle.com/nominate

Connect: **aceprogram_ww@oracle.com** Facebook.com/OracleACEs @oracleace

Agenda

Agenda

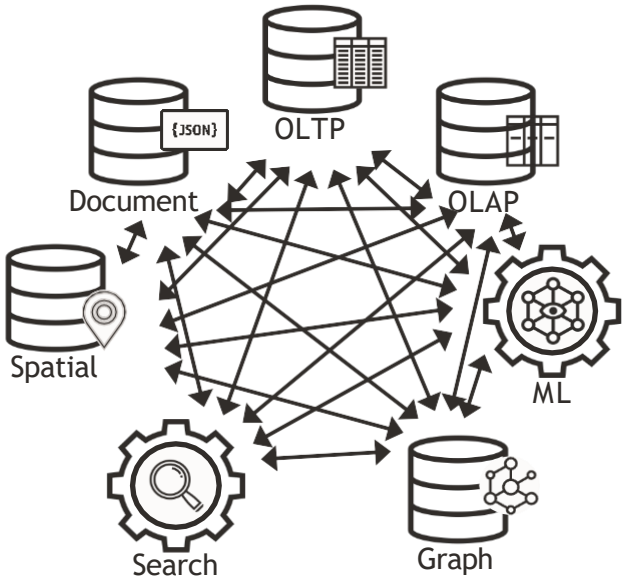
1. **Why?** Por que armazenar dados em JSON?
2. **With what?** JSON Datatype & Oracle Binary JSON (OSON)
3. **Where?** Oracle JSON-Relational Duality Views
4. **How?** Autonomous Database & JSON Oracle JDBC
5. **Which?** Spring Data JDBC



Por que armazenar dados em JSON?

Oracle Converged Database
O futuro é agora!

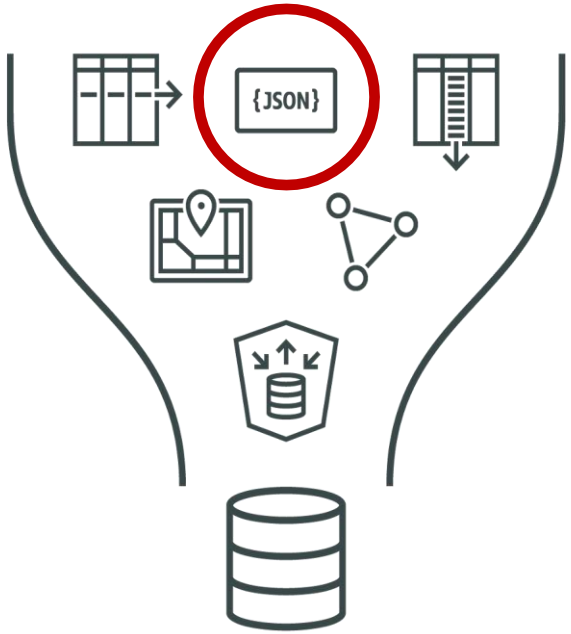
Single-purpose databases



Um para cada data type e workload

Multiplos security models, languages, skills, licenses, etc

Converged Database Architecture



Para **qualquer** data type ou workload

Oracle Converged Database
O future é agora!

Specialized Databases Trends

IMS | Relational DBs | Object DBs | Document, Key-value, Columnar, etc. | Graph DBs | Next DB?

1960s | 1970s | 1980s | 1990s | 2000s | 2010s | 2020s | Future

Object Types

XML Type
SQL/XML
XQuery

JSON Datatype
SQL/JSON

SQL with Graph

JSON Relational Duality Views

ORACLE
Converged Database

Armazenando dados em JSON

```
{
   'movie_id'   : 1234,
   'title'      : 'Iron Man',
   'date'       : '2010-05-07',
   'cast"       :
    [
        'Robert Downey Jr.',
        'Gwyneth Paltrow',
        'Jon Favreau'
    ]
}
```

```
class Movie {

   int movie_id;
   String title;
   Date LocalTime;

   Movie() {
     ...

}
```

```
db.movies.insertOne(
                movieValue
);



db.movies.find(
                {"movie_id" : 1653 }
);
```

Schema-flexible

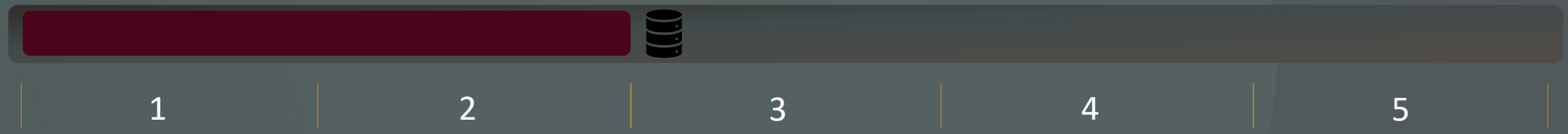
- ✓ Nenhum esquema inicial.
- ✓ Esquema definido pela aplicação.
- ✓ Modelo simplificado de dados.
- ✓ Evolução do esquema sem downtime ou alterações no lado do servidor

Baixa incompatibilidade com objetos de domínio

- ✓ Mapeia os objetos das aplicações.
- ✓ Suporte a estruturas aninhadas (nested structures).
- ✓ Leitura/gravação sem joins.
- ✓ Single-digit latency para leituras e gravações.

NoSQL

- ✓ Uma representação em todos os níveis
- ✓ REST e JSON são onipresentes
- ✓ Sem incorporação de código SQL.
- ✓ Escalabilidade
- ✓ Compatibilidade de JDBC com aplicativos existentes



JSON Datatype & Oracle Binary JSON (OSON)

O melhor dos dois mundos!

Oracle Database – as melhores features para storage para **JSON NoSQL document…** … com todos os **benefícios do Oracle Database Relacional.**

Elastic compute & storage

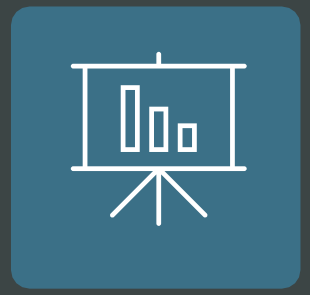
- ✓ **Scalable compute**
- ✓ **Scalable storage**

Single-digit latency reads & writes

- ✓ **Baixa latência**
- ✓ **Alto throughput**
- ✓ **SQL optional**
- ✓ **APIs Collection**
- ✓ **Schema-flexible JSON**

Highly available & Security

- ✓ **Alta disponibilidade**
- ✓ **Secure by default**
- ✓ **ACID transactions**
- ✓ **ANSI SQL / JSON**



Modern & Analytics

- ✓ **Full-text search**
- ✓ **Mission critical use cases**
- ✓ **Advanced Analytics**
- ✓ **Real-time analytics**
- ✓ **In-memory columnar**
- ✓ **Resource management**

JSON Storage History in the Oracle Database

Oracle Database 12c

Suporte para armazenamento de dados JSON-Text diretamente no banco de dados.
Suporte a query processing (clob, blob, varchar2).
Utilização de tipos de dados nativos (JSON datatype).

Oracle Database 18c

Melhorias de performance e facilidade de uso para manipulação de dados JSON.
SQL Enhancements for JSON
SQL/JSON query e functions podem retornar resultados como um LOB data.
Materialized views criadas com json_table podem ser sincronizadas automaticamente.

Oracle Database 12c Release 2 (12.2)

Aprimoramentos no suporte JSON, incluindo índices para consultas JSON, consultas SQL sobre dados JSON e suporte a transações ACID nas operações CRUD com JSON.

JSON Storage History in the Oracle Database

Oracle Database 21c

Tipo de dados JSON nativo e collections apoiadas por OSON (todos os tipos de banco de dados).
New Oracle SQL Function JSON_TRANSFORM.
JSON_SCALAR Syntax Improvements.
Multivalue Index for JSON.

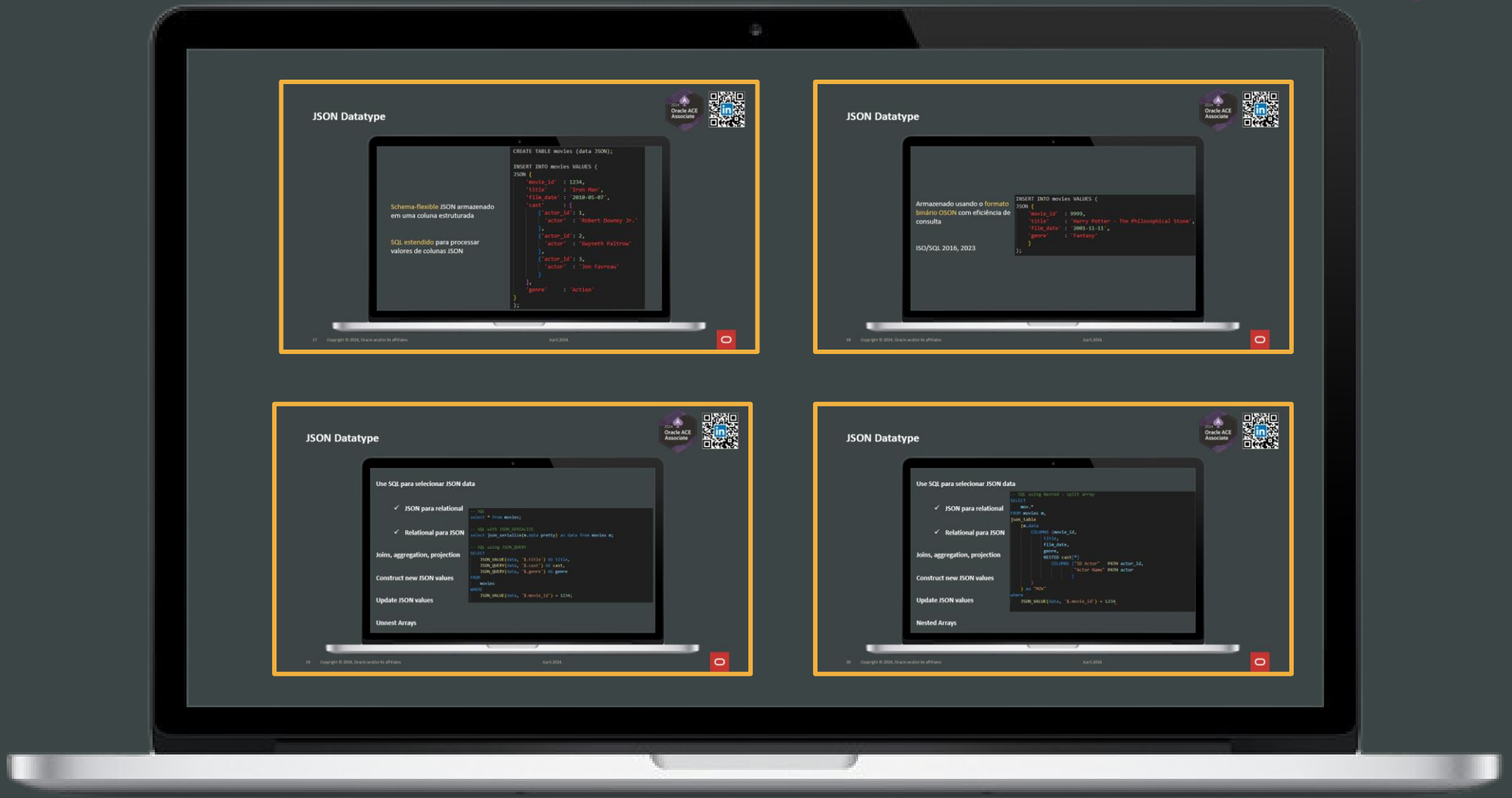
Oracle Database 19c

Binary JSON storage (OSON).
Mongo API support added for Autonomous Databases (in BLOB columns).
Materialized View Support for Queries with JSON_TABLE.
JSON Update Operations.
SQL/JSON Syntax Simplifications.
JSON Object Mapping.
New SQL/JSON Function JSON_SERIALIZE.

Oracle Database 23c

Oracle JSON-Relational Duality
JSON Schema
XML and JSON Search Index Enhancements
DBMS_AQ Support for JSON Arrays
Enhancement to JSON_TRANSFORM
JSON Type Support for External Tables
JSON_ARRAY Constructor by Query
Predicates for JSON_VALUE and JSON_QUERY

JSON Datatype



JSON Datatype

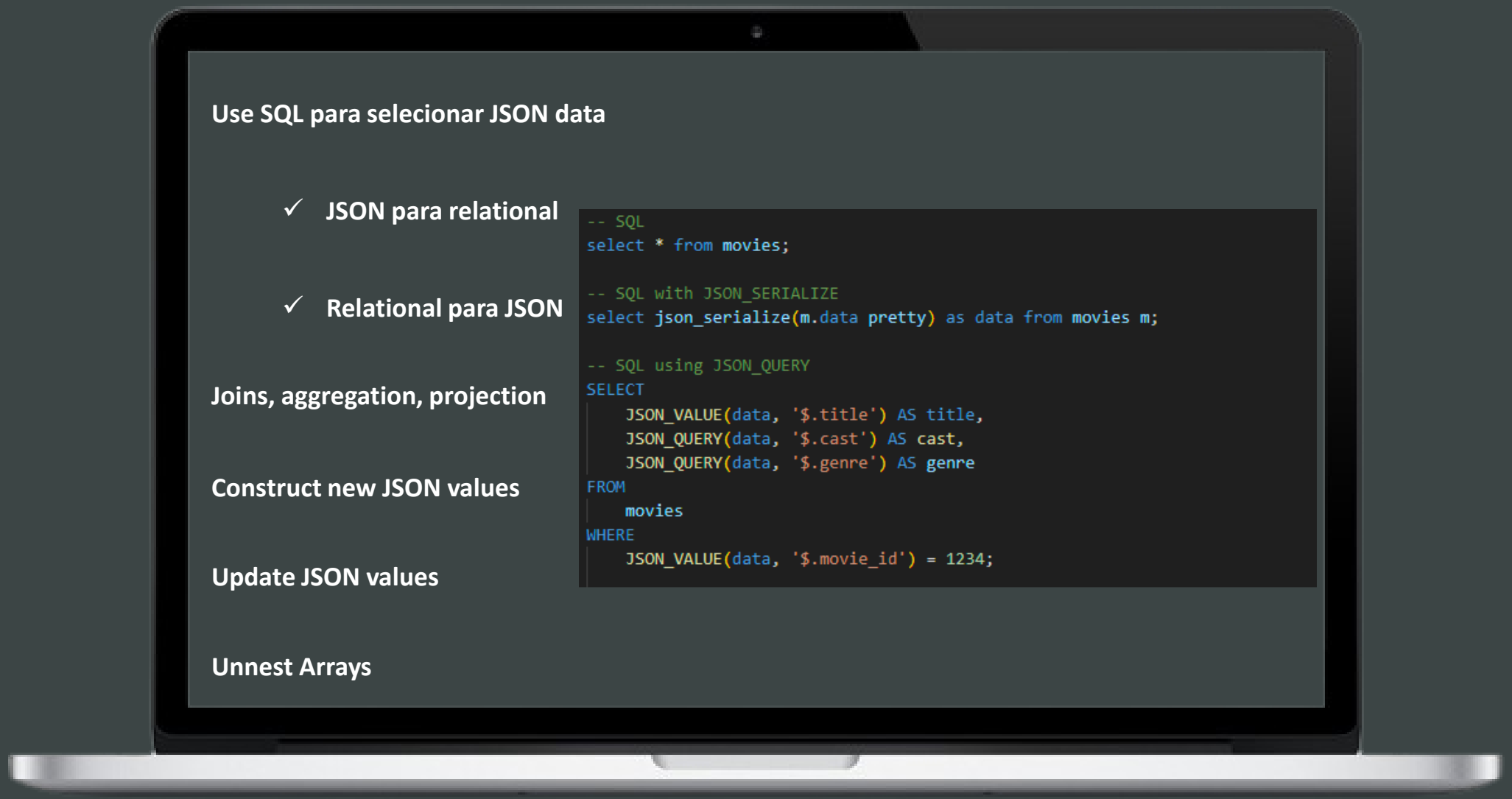
JSON Datatype

Armazenado usando o formato binário OSON com eficiência de consulta

ISO/SQL 2016, 2023

```
INSERT INTO movies VALUES (
JSON {
    'movie_id'   : 9999,
    'title'      : 'Harry Potter - The Philosophical Stone',
    'film_date'  : '2001-11-11',
    'genre'      : 'Fantasy'
    }
);
```

JSON Datatype



Use SQL para selecionar JSON data

- ✓ JSON para relational
- ✓ Relational para JSON

Joins, aggregation, projection

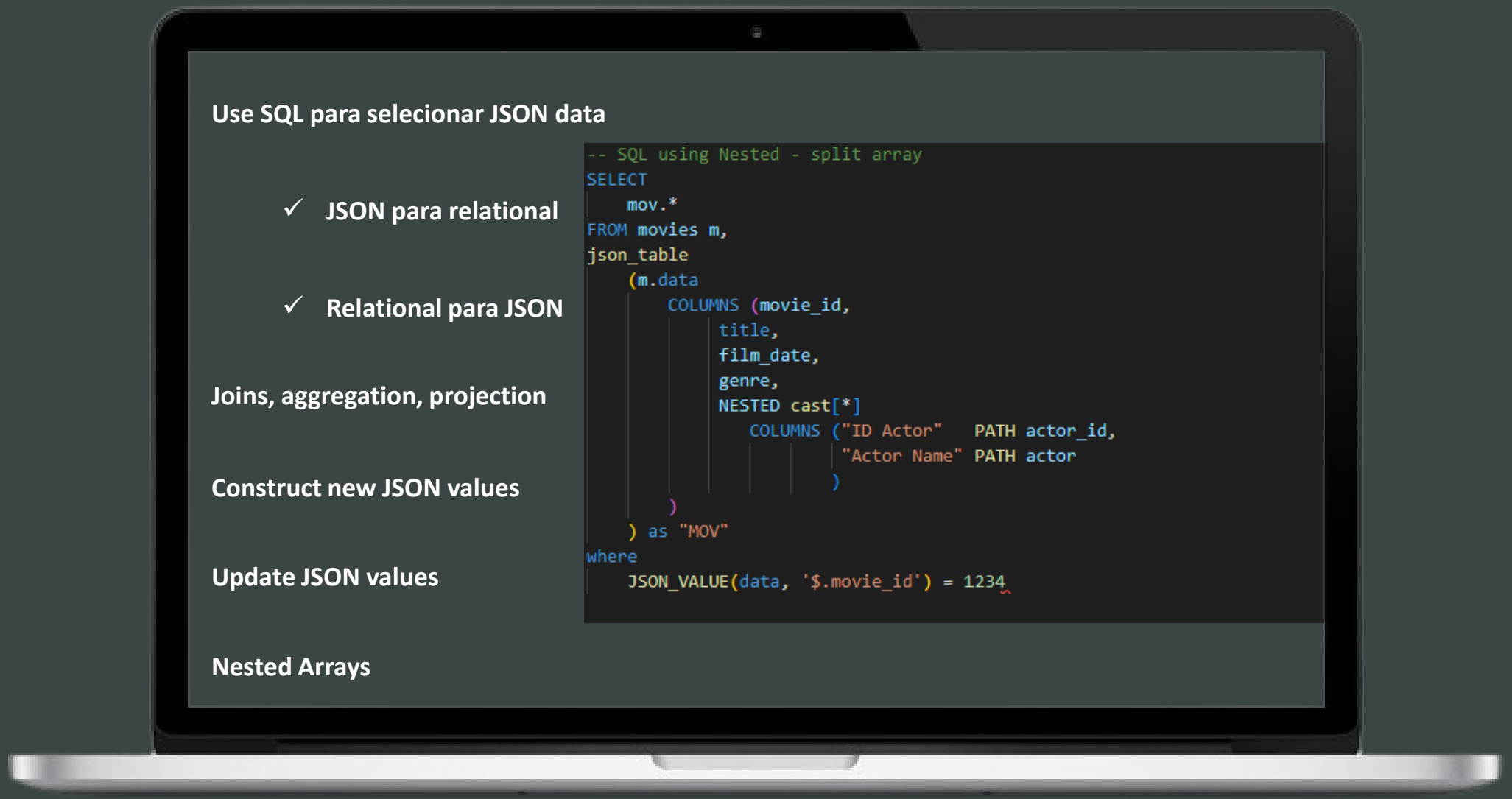
Construct new JSON values

Update JSON values

Unnest Arrays

```
-- SQL
select * from movies;

-- SQL with JSON_SERIALIZE
select json_serialize(m.data pretty) as data from movies m;

-- SQL using JSON_QUERY
SELECT
    JSON_VALUE(data, '$.title') AS title,
    JSON_QUERY(data, '$.cast') AS cast,
    JSON_QUERY(data, '$.genre') AS genre
FROM
    movies
WHERE
    JSON_VALUE(data, '$.movie_id') = 1234;
```

JSON Datatype



Use SQL para selecionar JSON data

- ✓ JSON para relational
- ✓ Relational para JSON

Joins, aggregation, projection

Construct new JSON values

Update JSON values

Nested Arrays

```
-- SQL using Nested - split array
SELECT
    mov.*
FROM movies m,
json_table
    (m.data
        COLUMNS (movie_id,
                 title,
                 film_date,
                 genre,
                 NESTED cast[*]
                    COLUMNS ("ID Actor"    PATH actor_id,
                             "Actor Name" PATH actor
                            )
        )
    ) as "MOV"
where
    JSON_VALUE(data, '$.movie_id') = 1234
```

JSON Datatype

JSON type é armazenado e transmitido usando o Oracle binary JSON format (OSON)

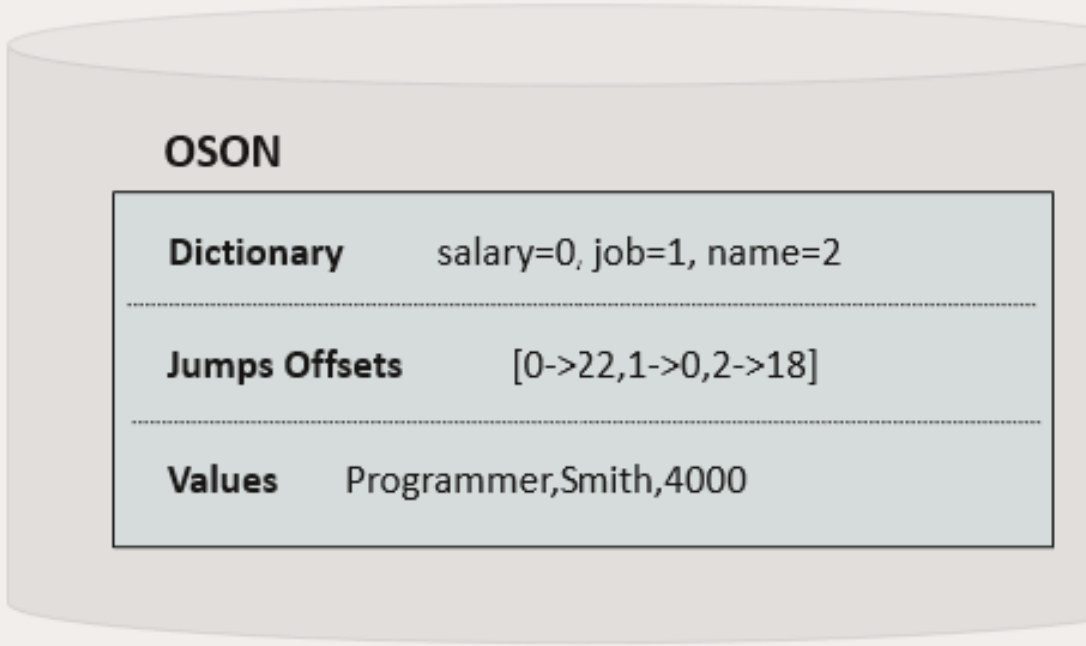
- Internal indexes suportam eficientemente acessos randômicos a objetos e arrays.
- Self-contained (autônomo), schema flexible.
- Object keys codificadas em um dicionário de dados com header autônomo.

Inserção/recuperação

- Compressão de documentos devido a codificação do dicionário de dados.
- Com isso, reduz o IO e os custos de transferência.
- Acesso In-place possível com baixa latência e baixo consumo de memória.
- Não é necessário copiar dados para estruturas de dados intermediárias (hash tables, arrays, etc).

Query & Update

- A avaliação da consulta não requer análise de texto, podendo navegar diretamente para valores aninhados (nested values).
- Avaliação de path sem usar comparações de strings nas chaves.
- **Update -** suporta fine-grained updates sem necessidade de substituição do document todo.



Database using OSON

OSON - Oracle binary JSON

Oracle binary JSON (OSON)

Otimização da Oracle do formato binário JSON, adicionando tipos escalares (data e double) que não fazem parte do padrão JSON.

Baseado em uma codificação de árvore (tree encoding).

Todos os documentos JSON no Oracle Database são armazenados automaticamente no formato OSON.

Grandes benefícios de desempenho para aplicativos JSON

Desempenho de consulta mais rápido.

Atualizações mais eficientes.

Tamanhos de armazenamento reduzidos.

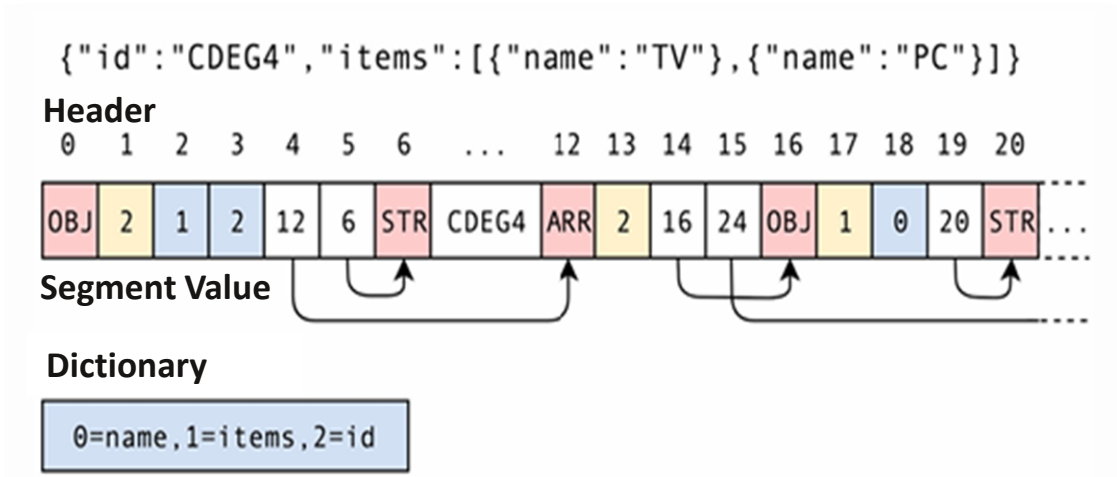
Totalmente transparente para as aplicações

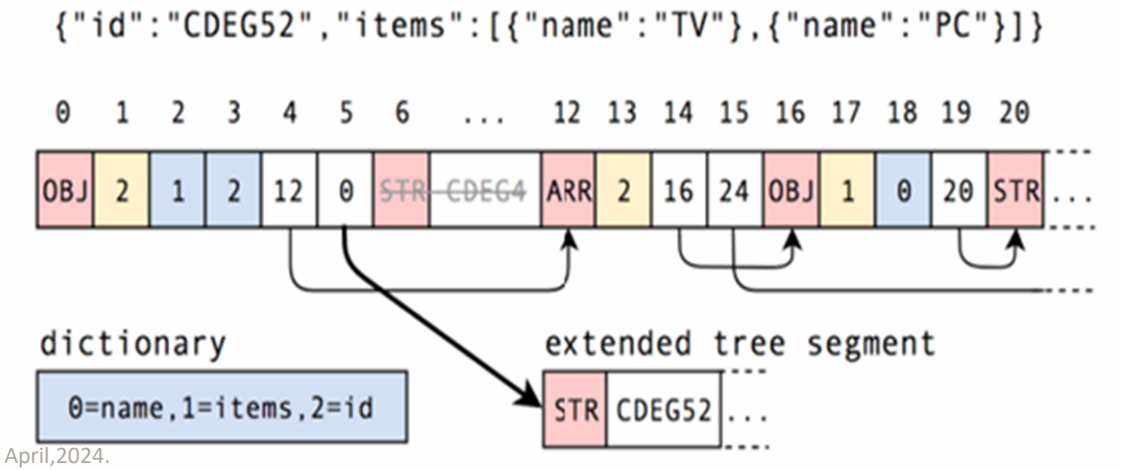
As aplicações sempre usa documentos JSON padrão (por exemplo, cadeias de texto).

Todas as operações de banco de dados em documentos JSON são otimizadas de forma automática por meio desse formato binário.

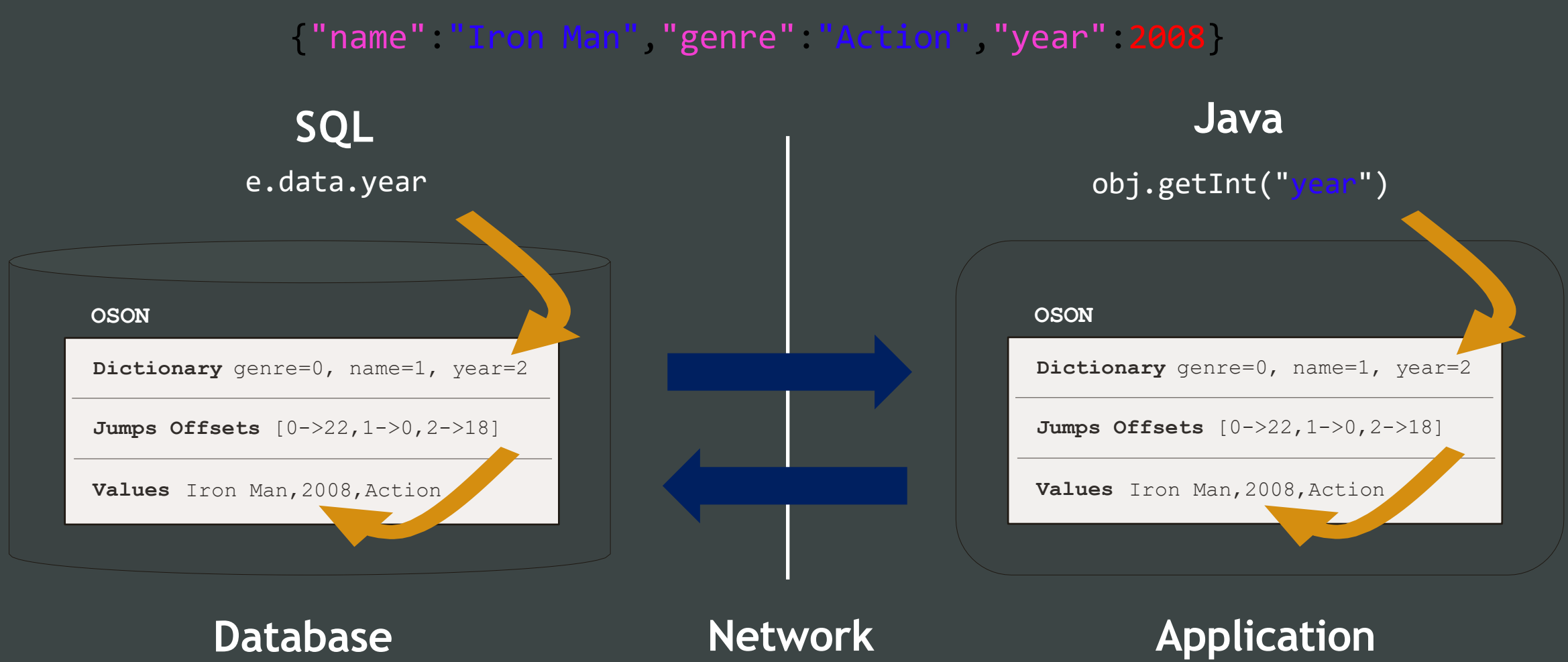
Baseado em uma codificação de árvore

Ao lado, uma codificação OSON de um documento JSON simples como uma matriz de bytes OSON serializada com ponteiros de árvore representados como offsets de navegação de salto (jump navigation offsets).





OSON - Oracle Binary JSON



OSON - Oracle binary JSON

Standard JSON

- ✓ OBJECT { }
- ✓ ARRAY [ ]
- ✓ STRING
- ✓ NUMBER
- ✓ TRUE/FALSE
- ✓ NULL

Extended SQL types

- ✓ BINARY_FLOAT
- ✓ BINARY_DOUBLE
- ✓ TIMESTAMP/DATE
- ✓ INTERVALDS/INTERVALYM
- ✓ RAW

Fidelity with relational data

```
CREATE TABLE orders VALUES (
    oid          NUMBER,
    created      TIMESTAMP,
    status       VARCHAR2(10),
);


{
  "oid":123,
  "created":"2020-06-04T12:24:29Z",
  "status":"OPEN"
}
```

OSON - Oracle binary JSON

Standard JSON

- ✓ OBJECT { }
- ✓ ARRAY [ ]
- ✓ STRING
- ✓ NUMBER
- ✓ TRUE/FALSE
- ✓ NULL

Extended SQL types

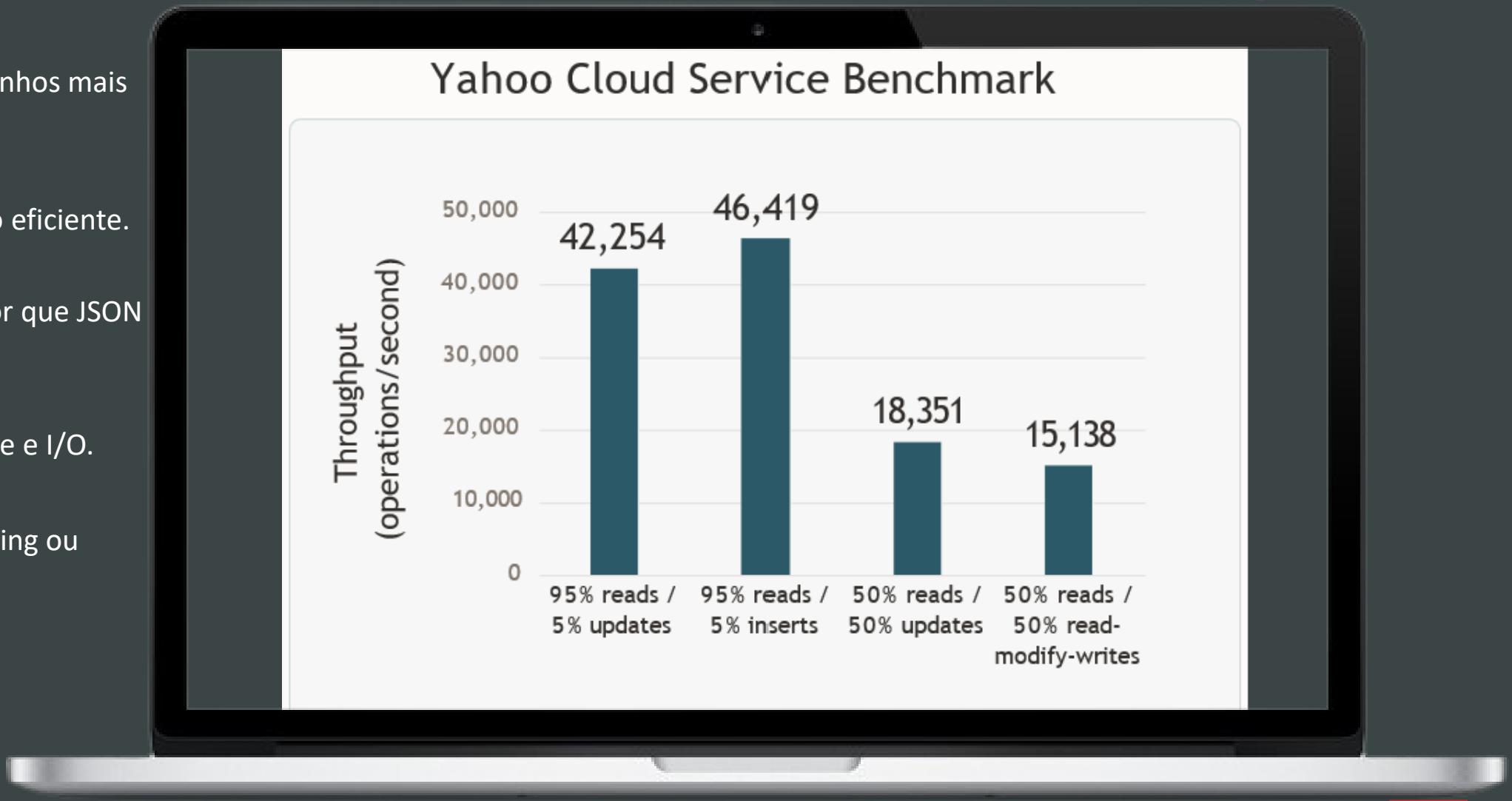
- ✓ BINARY_FLOAT
- ✓ BINARY_DOUBLE
- ✓ TIMESTAMP/DATE
- ✓ INTERVALDS/INTERVALYM
- ✓ RAW

ID	UTF8 (b)	BSON (b)	OSON (b)	Dict	vsUTF8	vsBSON
D1	613	764	524	5%	0.9x	0.7x
D2	1,782	1,813	1,950	30%	1.1x	1.1x
D3	2,608	3,094	2,160	16%	0.8x	0.7x
D4	2,943	3,293	2,476	6%	0.8x	0.8x
D5	8,842	8,440	5,591	19%	0.6x	0.7x
D6	40,285	37,526	20,486	18%	0.5x	0.5x
D7	76,861	75,195	38,383	11%	0.5x	0.5x
D8	141,051	133,307	103,897	0%	0.7x	0.8x
D9	682,228	No Data	483,053	0%	0.7x	No Data
D10	3,374,379	3,303,387	2,167,101	0%	0.6x	0.7x
D11	41,548,995	37,352,414	13,801,333	0%	0.3x	0.4x

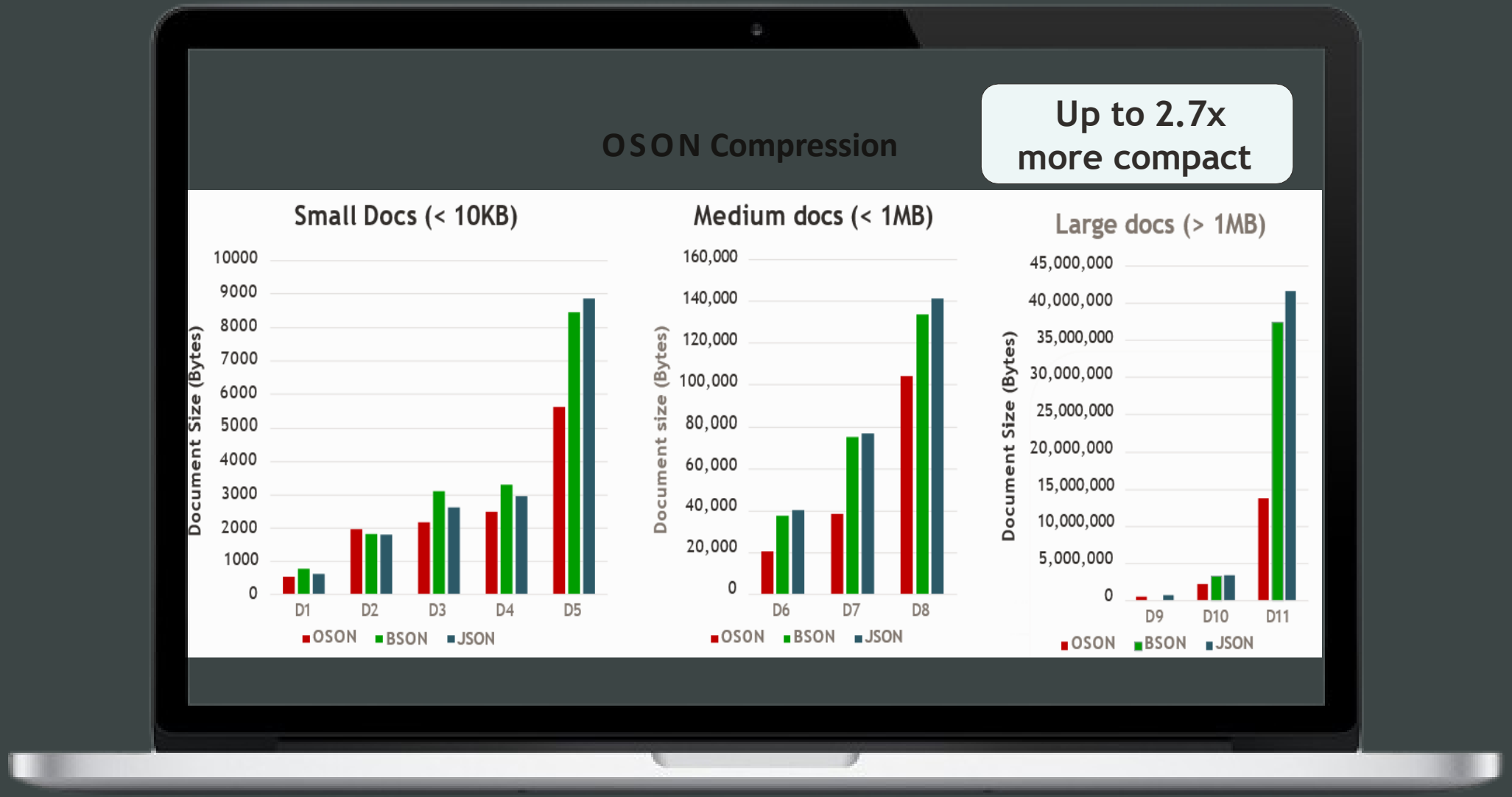
OSON - Oracle binary JSON

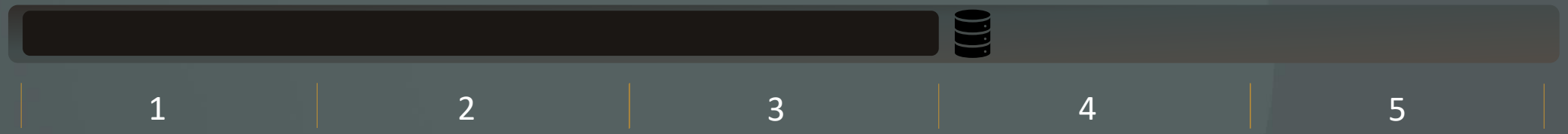
Performance

- ✓ Avaliação de caminhos mais rápidos.
- ✓ Acesso randômico eficiente.
- ✓ Throughput menor que JSON text e BSON.
- ✓ Menor uso de rede e I/O.
- ✓ Nenhum text parsing ou serialização.



OSON - Oracle binary JSON

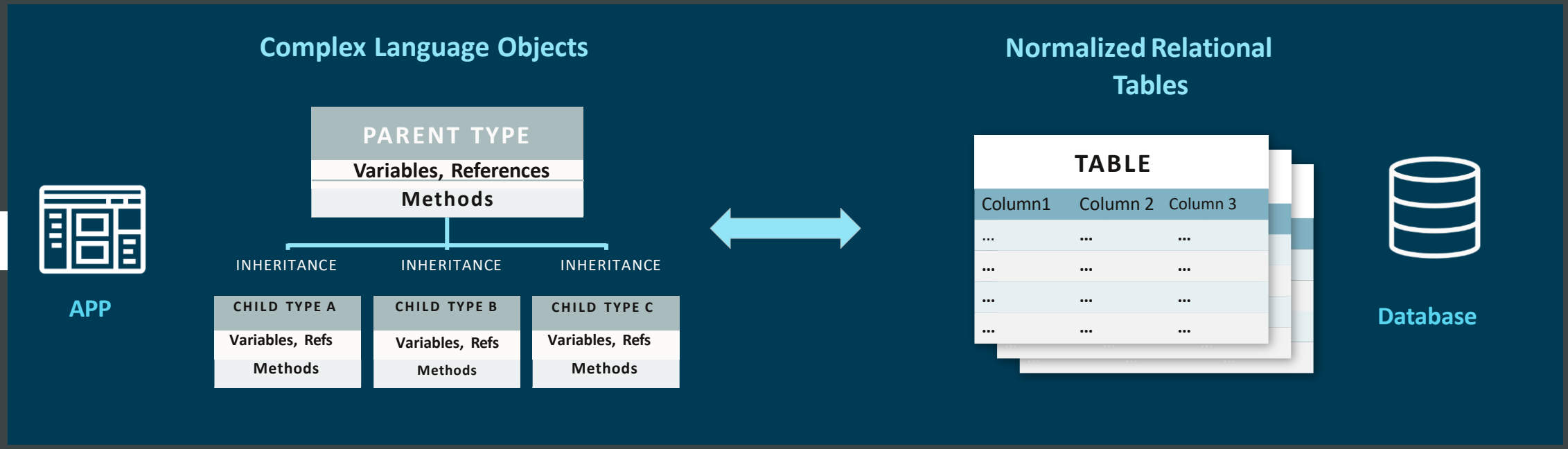




Oracle JSON-Relational Duality Views

Desafio: Traduzir o mundo relacional para a linguagem do APP

Nem sempre é simples para os desenvolvedores traduzir o mundo relacional para um APP-TIER Objects, já que o database foi modelado pensando na normalização.



- ✓ Muitas chamadas ao database para manipular um único app-tier object;
- ✓ Muitas vezes são ineficientes porque não tiram proveito das features do database engine;
- ✓ Não gerenciam bem os controles de concorrência de transações;

Oracle JSON-Relational Duality Views

Documents e relacional totalmente unificados. Você teria todos os benefícios do relacional + todos os benefícios do JSON!

Oracle JSON-Relational Duality Views

Relacional

- ✓ Sem duplicação de dados
- ✓ Flexibilidade de casos de uso
- ✓ Capacidade de consulta
- ✓ Consistência
- ✓ Normalização de dados (1ºFN, 2ºFN, 3ºFN, Boyce-Codd)
- ✓ Modelos de dados rígidos baseado em tabelas, índices, linhas, colunas e relacionamentos
- ✓ Modelo de dados voltado a normalização, nunca pensando na query

JSON

- ✓ Mapeamento fácil para operações de aplicativos
- ✓ Desenvolvimento ágil sem esquema
- ✓ Formato de dados hierárquico
- ✓ Mapeamento fácil para classes de dados de aplicativos
- ✓ Construção de APP-TIER Objects facilitadas
- ✓ Object Relational Mapping (ORM) não é necessário
- ✓ Desenvolvimento orientado às necessidades das queries e resultados

JSON Duality Views

Transações ACID.

Possível manipular dados de vários tipos, incluindo JSON no mesmo database. Combina benefícios do relacional (storage, consistência, eficiência) com a flexibilidade dos documents JSON.

Document-centric – Oracle Database API for MongoDB, Oracle REST Data Services (ORDS), JSON document functions ou SQL. Suporte nativo a JSON.

Eliminam a necessidade do uso do ORM.

Estatísticas, Analytics, auditória, fine-grained access control e features de segurança avançados.

Oracle JSON-Relational Duality Views

DBA

- ✓ Sem duplicação de dados
- ✓ Flexibilidade de casos de uso
- ✓ Capacidade de consulta
- ✓ Consistência
- ✓ Normalização de dados (1ºFN, 2ºFN, 3ºFN, Boyce-Codd)
- ✓ Modelos de dados rígidos baseado em tabelas, índices, linhas, colunas e relacionamentos
- ✓ Modelo de dados voltado a normalização, nunca pensando na query

JSON

- ✓ Mapeamento fácil para operações de aplicativos
- ✓ Desenvolvimento ágil sem esquema
- ✓ Formato de dados hierárquico
- ✓ Mapeamento fácil para classes de dados de aplicativos
- ✓ Construção de APP-TIER Objects facilitadas
- ✓ Object Relational Mapping (ORM) não é necessário
- ✓ Desenvolvimento orientado às necessidades das queries e resultados

JSON Duality Views

Transações ACID.

Possível manipular dados de vários tipos, incluindo JSON no mesmo database. Combina benefícios do relacional (storage, consistência, eficiência) com a flexibilidade dos documents JSON.

Document-centric – Oracle Database API for MongoDB, Oracle REST Data Services (ORDS), JSON document functions ou SQL. Suporte nativo a JSON.

Eliminam a necessidade do uso do ORM.

Estatísticas, Analytics, auditória, fine-grained access control e features de segurança avançados.

Oracle JSON-Relational Duality Views

DBA

- ✓ Sem duplicação de dados
- ✓ Flexibilidade de casos de uso
- ✓ Capacidade de consulta
- ✓ Consistência
- ✓ Normalização de dados (1ºFN, 2ºFN, 3ºFN, Boyce-Codd)
- ✓ Modelos de dados rígidos baseado em tabelas, índices, linhas, colunas e relacionamentos
- ✓ Modelo de dados voltado a normalização, nunca pensando na query

JSON

- ✓ Mapeamento fácil para operações de aplicativos
- ✓ Desenvolvimento ágil sem esquema
- ✓ Formato de dados hierárquico
- ✓ Mapeamento fácil para classes de dados de aplicativos
- ✓ Construção de APP-TIER Objects facilitadas
- ✓ Object Relational Mapping (ORM) não é necessário
- ✓ Desenvolvimento orientado às necessidades das queries e resultados

JSON Duality Views

Transações ACID.

Possível manipular dados de vários tipos, incluindo JSON no mesmo database. Combina benefícios do relacional (storage, consistência, eficiência) com a flexibilidade dos documents JSON.

Document-centric – Oracle Database API for MongoDB, Oracle REST Data Services (ORDS), JSON document functions ou SQL. Suporte nativo a JSON.

Eliminam a necessidade do uso do ORM.

Estatísticas, Analytics, auditória, fine-grained access control e features de segurança avançados.

Oracle JSON-Relational Duality Views

Criação de tabelas seguindo o modelo relacional.

Benefícios:

- ✓ **Auditoria**
- ✓ **Estatísticas**
- ✓ **Analytics**
- ✓ **Fine-grained access control**
- ✓ **Features de segurança avançados;**

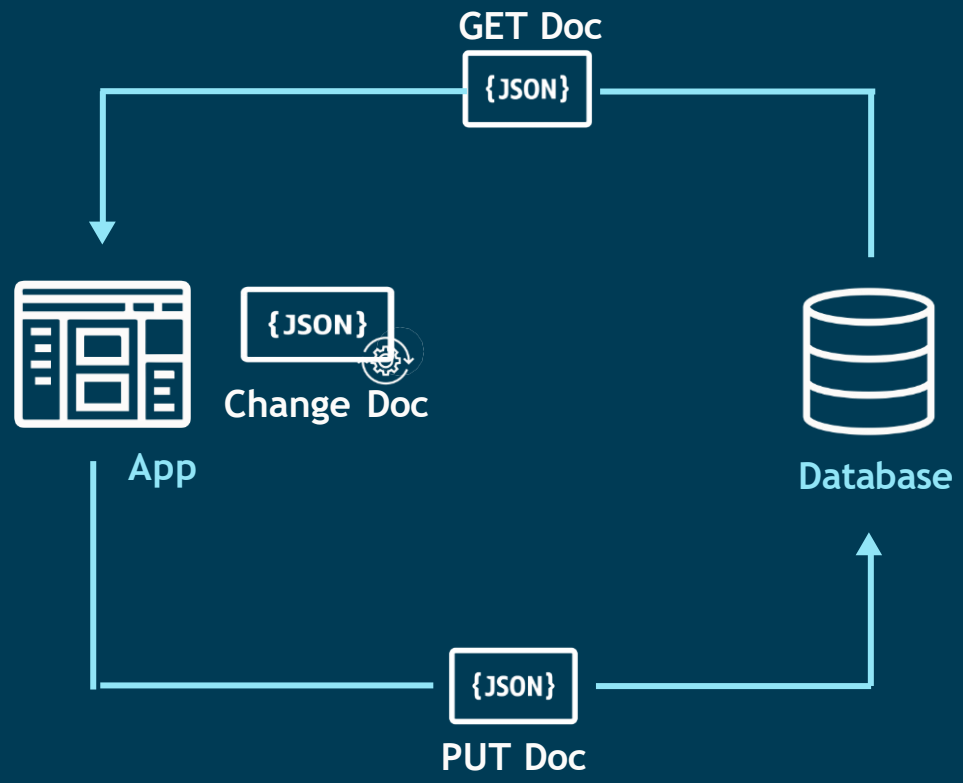
Utilizando Oracle JSON-Relational Duality Views com AutoREST

JSON-Duality Views asão acessadas de forma extremamente simples usando REST:

- ✓ Obter (GET) o document da view.
- ✓ Fazer as alterações necessárias no document.
- ✓ Retornar (PUT) o document para a view.

O banco de dados automaticamente detecta as mudanças no novo document e modifica as linhas envolvidas.

- ✓ Todas as duality views que compartilham os mesmos dados refletem imediatamente essa alteração.
- ✓ Os desenvolvedores não precisam mais se preocupar com inconsistências.



Utilizando Oracle JSON-Relational Duality Views com AutoREST

Lock-Free Concurrency Control

Precisamos certificar que a versão/estado do document não foi alterado após sua chamada. E em “operações sem estado” como PUT, GET, os algoritmos de lock não mantém bloqueios efetivos.

Para as DVs é utilizado o “**Lock-free concurrency control algorithm**". Esse algoritmo permite que atualizações consistentes sejam realizadas em operações sem estado.

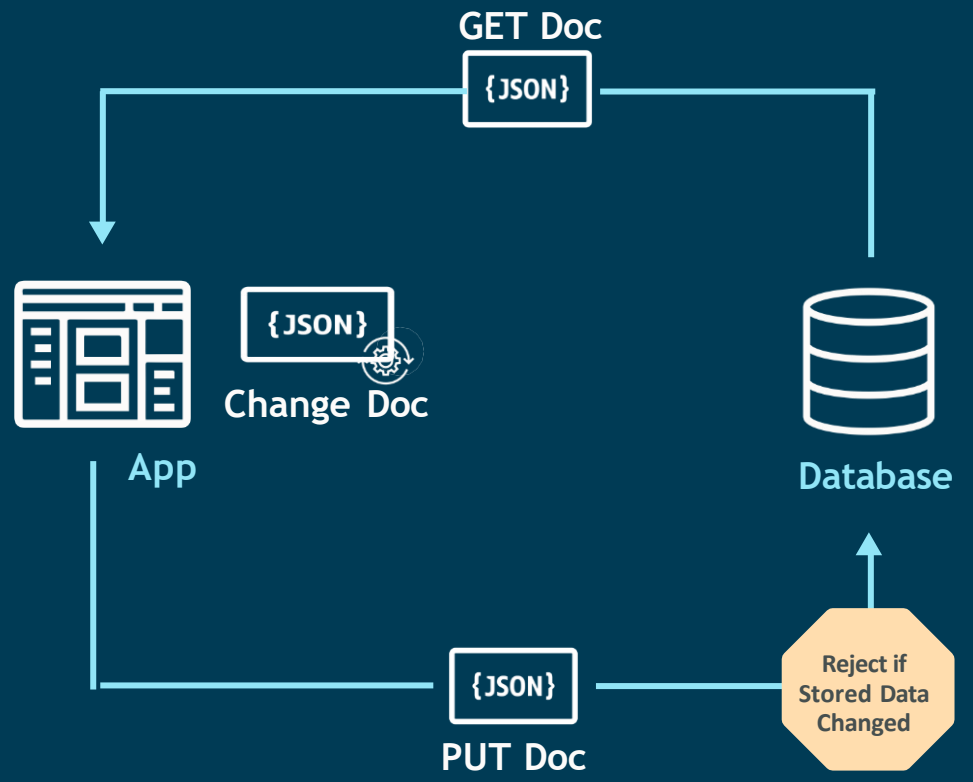
Por default, todo document em uma DV registra o seu estado no **Entity Tag (ETAG)**. Ele é um HASH calculado pelo conteúdo do document, e mais algumas informações, e é renovado automaticamente sempre que esse document é recuperado.

Quando uma atualização é realizada, o JSON devolvido contém um ETAG calculado e uma comparação com o document atual é feita. **Se os ETAGs forem diferentes, o document original foi modificado e a atualização será rejeitada.**

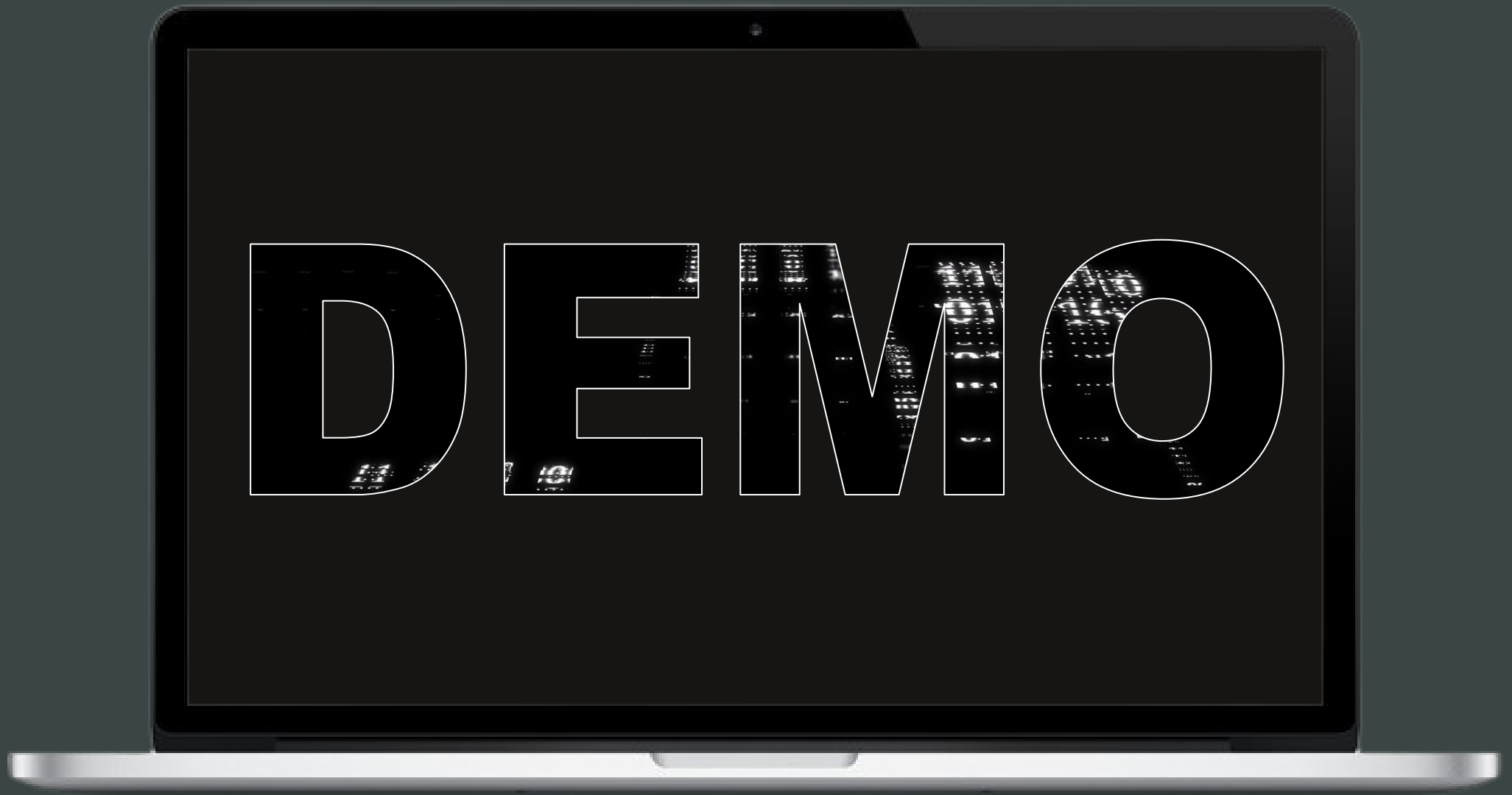
Sendo assim, temos a garantia de que não ocorreram alterações no document, assegurando a atomicidade e a consistência a nível do document.

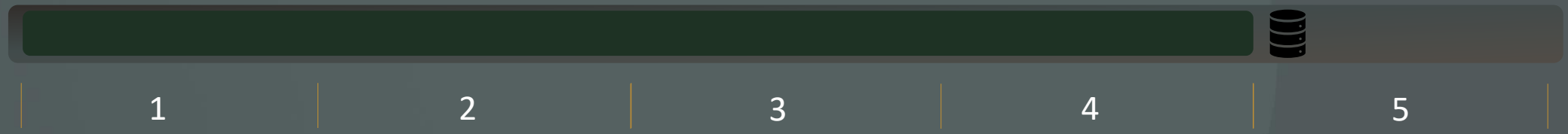
Para excluir um determinado campo do cálculo, usamos o NOCHECK. Se todas as colunas forem NOCHECK, nenhum campo será validado. Isso pode melhorar muito o desempenho para documents maiores. Você pode querer excluir as verificações do ETAG em:

- ✓ Um APP que tem seu próprio controle de consistência;
- ✓ Um APP Single thread - não é possível fazer modificações simultâneas.



Lock-Free Concurrency Control





Autonomous Database JSON & Oracle JDBC

Java API for JSON in JDBC

Facilidades para leitura, escrita e modificação de valores de Binary JSON com JAVA.

Features

- ✓ Mutable tree/object model
- ✓ Event-based parser e generator
- ✓ Acesso a extended SQL/JSON types (TIMESTAMP, DATE, etc.)
- ✓ Suporte a JSON text e OSON
- ✓ Integração opcional com JSON-P / JSR-374

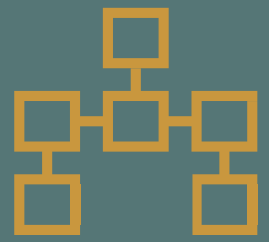
Onde encontrar…

JAR: ojdbc11.jar

- ✓ OTN
- ✓ Maven Central Repository
  (groupId = com.oracle.database.jdbc, artifactId = ojdbc11)

Package: oracle.sql.json

Package oracle.sql.json



Tree Model

OracleJsonObject

OracleJsonArray

OracleJsonString

OracleJsonDecimal

OracleJsonDouble

OracleJsonTimestamp

OracleJsonBinary

OracleJsonIntervalDS

OracleJsonIntervalYM

Event Model

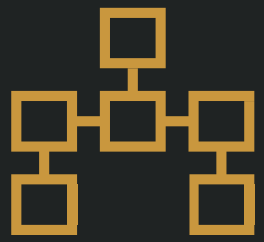
OracleJsonParser

OracleJsonGenerator

Factory

OracleJsonFactory

Package oracle.sql.json



Tree Model

OracleJsonObject

OracleJsonArray

OracleJsonString

OracleJsonDecimal

OracleJsonDouble

OracleJsonTimestamp

OracleJsonBinary

OracleJsonIntervalDS

OracleJsonIntervalYM

Event Model

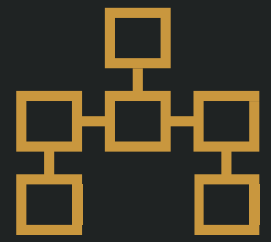
OracleJsonParser

OracleJsonGenerator

Factory

OracleJsonFactory

Package oracle.sql.json



Tree Model

OracleJsonObject

OracleJsonArray

OracleJsonString

OracleJsonDecimal

OracleJsonDouble

OracleJsonTimestamp

OracleJsonBinary

OracleJsonIntervalDS

OracleJsonIntervalYM

Event Model

OracleJsonParser

OracleJsonGenerator

Factory

OracleJsonFactory

OracleJsonFactory Methods

Read/write OSON

createJsonBinaryGenerator(OutputStream)
createJsonBinaryValue(ByteBuffer)
createJsonBinaryValue(InputStream)
createJsonBinaryParser(ByteBuffer)
createJsonBinaryParser(InputStream)

Read/write JSON text

createJsonTextGenerator(OutputStream)
createJsonTextGenerator(Writer)
createJsonTextValue(InputStream)
createJsonTextValue(Reader)
createJsonTextParser(InputStream)
createJsonTextParser(Reader)

In-memory tree model creation

createObject() createObject(OracleJsonObject)
createArray() createArray(OracleJsonArray)
createString(String) createDecimal(BigDecimal)
createDecimal(int) createDecimal(long)
createDouble(double) createTimestamp(Instant)

....

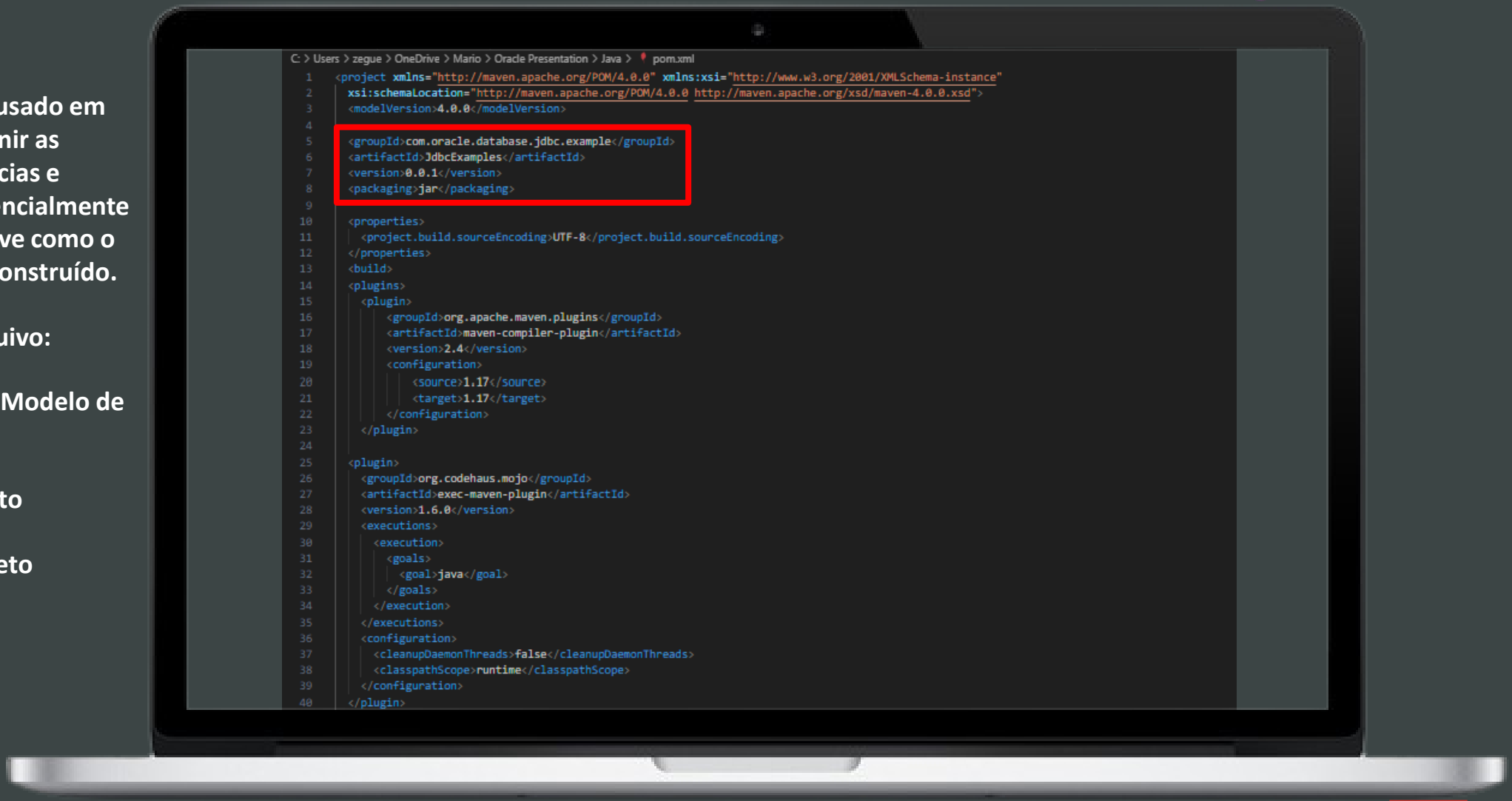
Package oracle.sql.json & Autonomous Database JSON (ADJ)

pom.xml

Arquivo de configuração usado em projetos Maven para definir as configurações, dependências e plugins do projeto. É essencialmente um manifesto que descreve como o projeto Maven deve ser construído.

Principais sessões do arquivo:

- ✓ **Possui Declaração de Modelo de Projeto**
- ✓ **Informações do Projeto**
- ✓ **Propriedades do Projeto**
- ✓ **Dependências**
- ✓ **Plugins**
- ✓ **Repositórios**

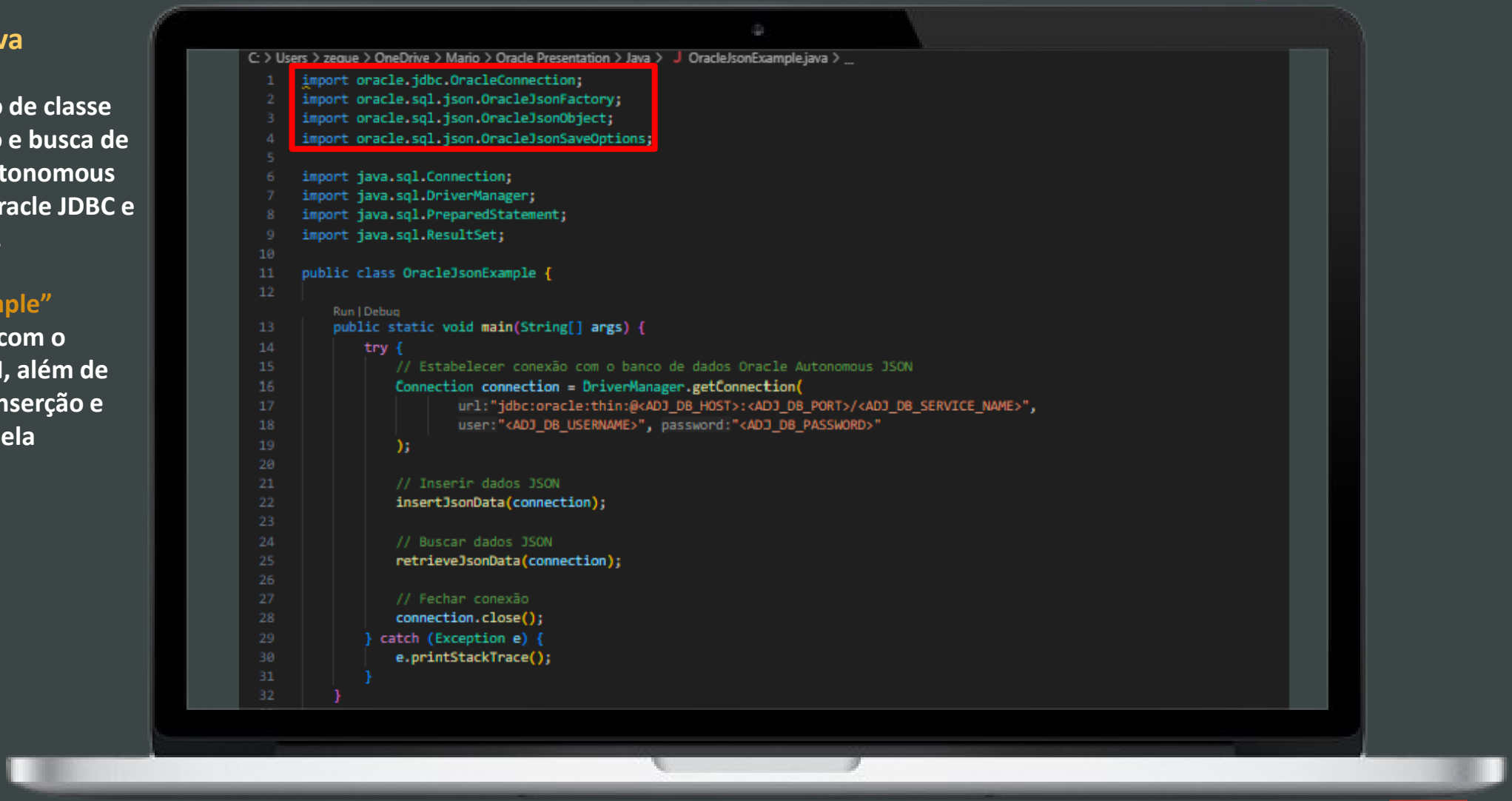


Package oracle.sql.json & Autonomous Database JSON (ADJ)

OracleJsonExample.java

Arquivo com um exemplo de classe com métodos de inserção e busca de dados JSON no Oracle Autonomous Database JSON usando Oracle JDBC e a package oracle.sql.json.

A classe “OracleJsonExample” estabelece uma conexão com o Oracle Autonomous JSON, além de definir os métodos para inserção e listagem de dados na tabela “produtos”.



Package oracle.sql.json & Autonomous Database JSON (ADJ)

OracleJsonExample.java

Arquivo com um exemplo de classe com métodos de inserção e busca de dados JSON no Oracle Autonomous Database JSON usando Oracle JDBC e a package oracle.sql.json.

A classe "OracleJsonExample" estabelece uma conexão com o Oracle Autonomous JSON, além de definir os métodos para inserção e listagem de dados na tabela "produtos".

C: > Users > zegue > OneDrive > Mario > Oracle Presentation > Java > J OracleJsonExample.java > ...

```
11  public class OracleJsonExample {
33
34      private static void insertJsonData(Connection connection) throws Exception {
35          // Criar um objeto JSON para inserção
36          OracleJsonFactory factory = new OracleJsonFactory();
37          OracleJsonObject json = factory.createObject();
38          json.put("id", 1);
39          json.put("name", "Produto A");
40          json.put("price", 10.0);
41          json.put("quantity", 100);
42
43          // Preparar a instrução SQL para inserção
44          String sql = "INSERT INTO produtos (json_column) VALUES (?)";
45          PreparedStatement statement = connection.prepareStatement(sql);
46          statement.setObject(parameterIndex:1, json);
47
48          // Executar a inserção
49          statement.executeUpdate();
50          System.out.println(x:"Dados JSON inseridos com sucesso.");
51      }
52
53      private static void retrieveJsonData(Connection connection) throws Exception {
54          // Preparar a consulta SQL para buscar os dados JSON
55          String sql = "SELECT json_column FROM produtos";
56          PreparedStatement statement = connection.prepareStatement(sql);
57
58          // Executar a consulta
59          ResultSet resultSet = statement.executeQuery();
60
61          // Processar os resultados
62          while (resultSet.next()) {
63              // Obter o objeto JSON da coluna
64              OracleJsonObject json = resultSet.getObject(columnLabel:"json_column", type:OracleJsonObject.class);
65
66              // Imprimir os dados JSON
67              System.out.println("ID: " + json.getInt("id"));
68              System.out.println("Nome: " + json.getString("name"));
69              System.out.println("Preço: " + json.getDouble("price"));
70              System.out.println("Quantidade: " + json.getInt("quantity"));
71          }
72      }
73  }
```

Package oracle.sql.json & Autonomous Database JSON (ADJ)

Performance Tips

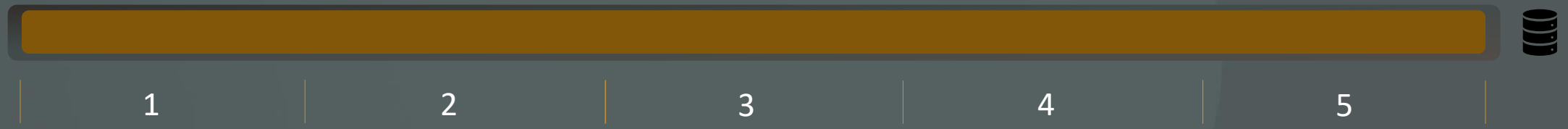
Reutilize instâncias **OracleJsonFactory** entre solicitações. Instâncias locais de thread são melhores. Instância global é aceitável, pois usa estruturas de dados seguras de thread sem bloqueio.

Não converta (de e para) **JSON text** sem necessidade. Acesse valores como **OracleJsonValue, OracleJsonParser, JSONP, JSONB** em vez de String, Reader, InputStream, etc.

Ao vincular um JSON, setObject() por exemplo, especifique o **OracleTypes.JSON**. Se o driver não souber que é um JSON, o servidor fará a codificação OSON em alguns casos.

Use universal connection pooling (UCP)

- ✓ Criar novas conexões "custa caro"
- ✓ Threads podem compartilhas uma conexão ao mesmo tempo, mas isso não é eficiente. Oracle Universal Connection Pool Developer's Guide, 21c
- ✓ Ative o statement caching e use bind variables



Spring Data JDBC

Spring Data JDBC

Spring Data JDBC tem como objetivo ser muito mais simples conceitualmente que o **Java Persistence API (JPA)**.

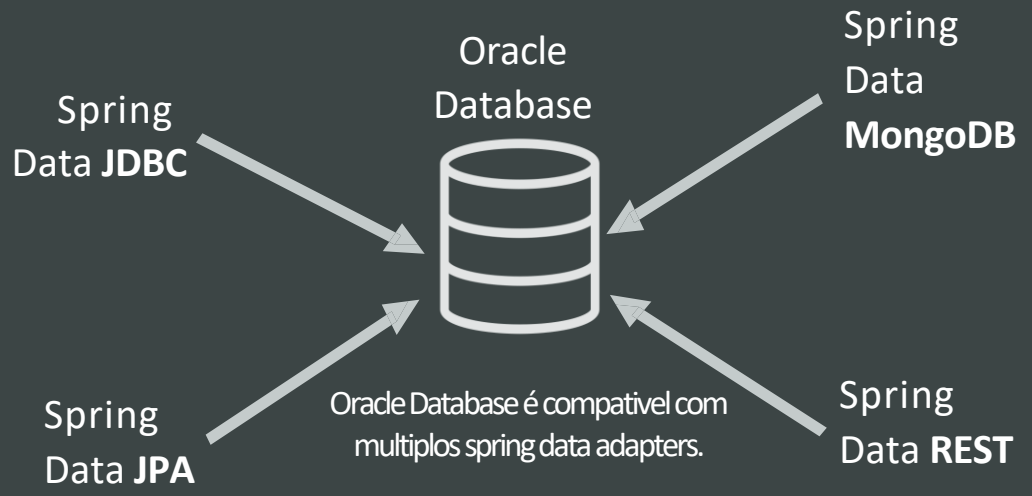
Fornece uma abstração de alto nível simplificando o trabalho com JDBC. Automatiza tarefas de baixo nível envolvidas no trabalho com API.

- ✓ Interação com objetos de negócios como um repositório de objetos (sem SQL ou JSON diretamente)
- ✓ Criação e o gerenciamento de conexões
- ✓ Preparação e a execução de instruções SQL
- ✓ Mapeamento de resultados para objetos Java

Evita a necessidade de adicionar um **JPA** como o **Hibernate** nos projetos, oferecendo uma abordagem direta para acesso aos dados.

Flexibilidade total para escrever consultas SQL, tornando-o ideal para projetos que demandam controle total sobre operações de banco de dados.

Facilita o mapeamento de resultados para objetos Java, permitindo uma **implementação mais rápida e eficiente** do acesso a dados.




```
public class Movie { @Id
    Integer id;
    String name;
    String genre;
}
```

```
Movie m = movies.findById(1).get();
System.out.println(m.getName() + " " + m.getId()
);
```

Spring Data JDBC X Java Persistence API (JPA)

Spring Data JDBC

- ✓ Busca ser conceitualmente mais simples.
- ✓ Funciona com SQL puro.
- ✓ Não usa o lazy loading ou caching.
- ✓ Sem rastreamento de alterações ou sessão.
- ✓ Modelo simples de mapeamento de entidades para tabelas.
- ✓ Suporte limitado para personalização de estratégia.

Java Persistence API

- ✓ Principal API de persistência para bancos de dados relacionais em Java.
- ✓ Rastreia alterações nas entidades.
- ✓ Realiza lazy loading.
- ✓ Mapeia objetos para designs de banco de dados.

Spring Data JDBC X Java Persistence API (JPA)

Spring Data JDBC

- ✓ Busca ser conceitualmente mais simples.
- ✓ Funciona com SQL puro.
- ✓ Não usa o lazy loading ou caching.
- ✓ Sem rastreamento de alterações ou sessão.
- ✓ Modelo simples de mapeamento de entidades para tabelas.
- ✓ Suporte limitado para personalização de estratégia.

Java Persistence API

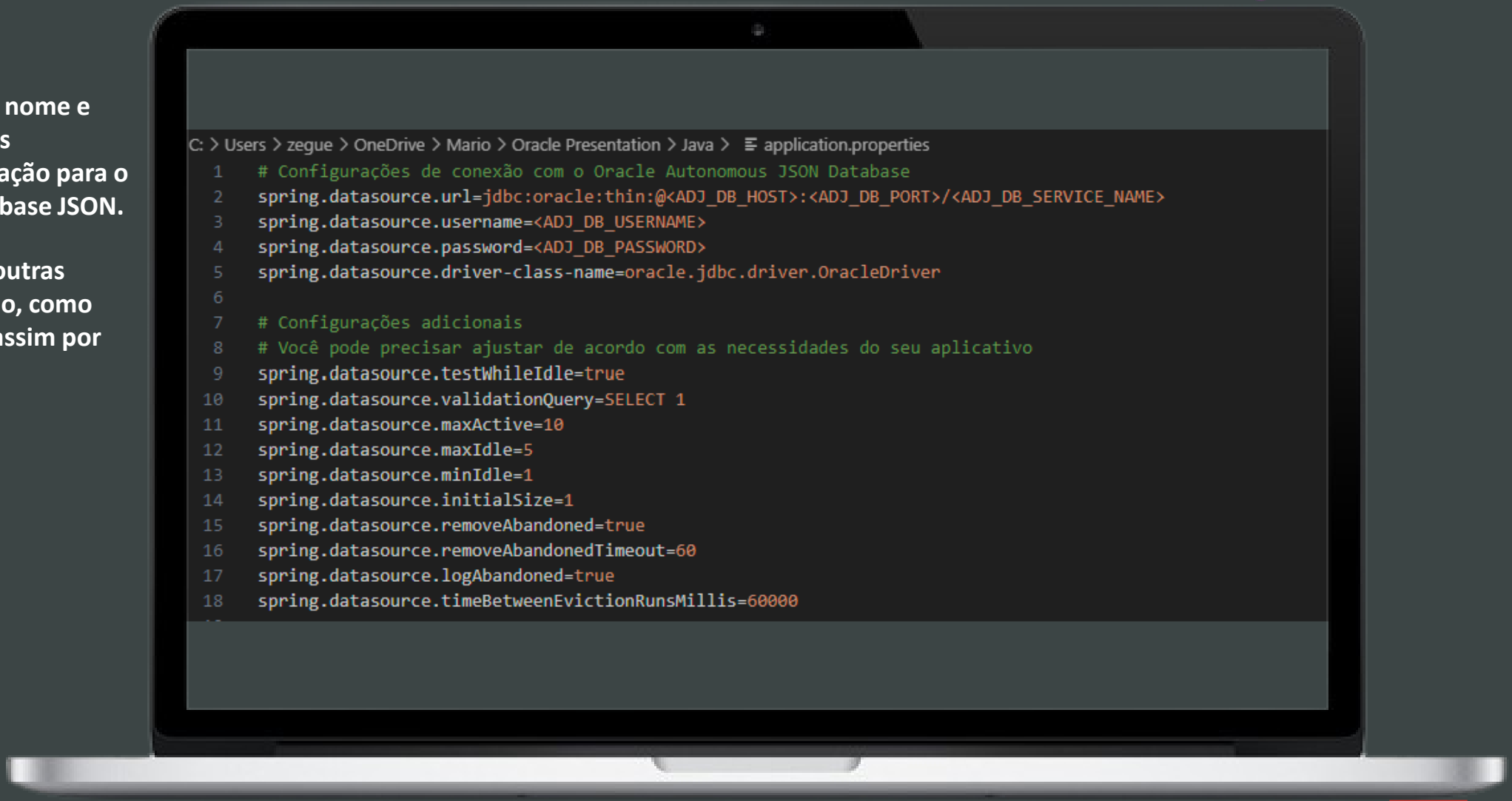
- ✓ Principal API de persistência para bancos de dados relacionais em Java.
- ✓ Rastreia alterações nas entidades.
- ✓ Realiza lazy loading.
- ✓ Mapeia objetos para designs de banco de dados.

Spring Data JDBC & Autonomous Database JSON (ADJ)

application.properties

Define a URL de conexão, nome e senha de usuário, e outras propriedades de configuração para o Oracle Autonomous Database JSON.

Além disso, pode conter outras configurações de aplicação, como portas de servidor, log e assim por diante.



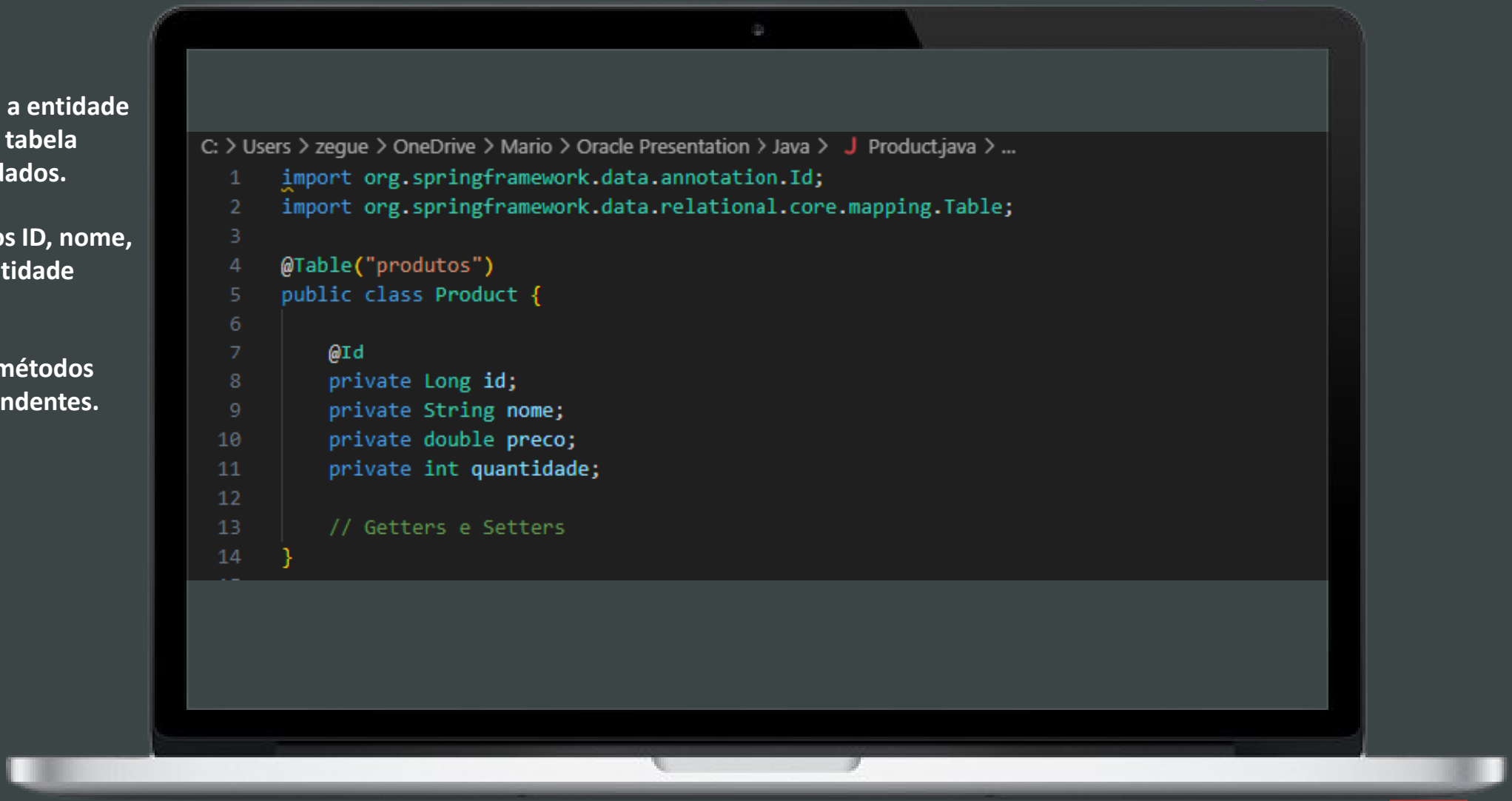
Spring Data JDBC & Autonomous Database JSON (ADJ)

Product.java

Esta classe é a representa a entidade Product que irá mapear a tabela "produtos" no banco de dados.

Aqui se define os atributos ID, nome, preço e quantidade da entidade Product.

Além disso definimos os métodos getters e setters correspondentes.

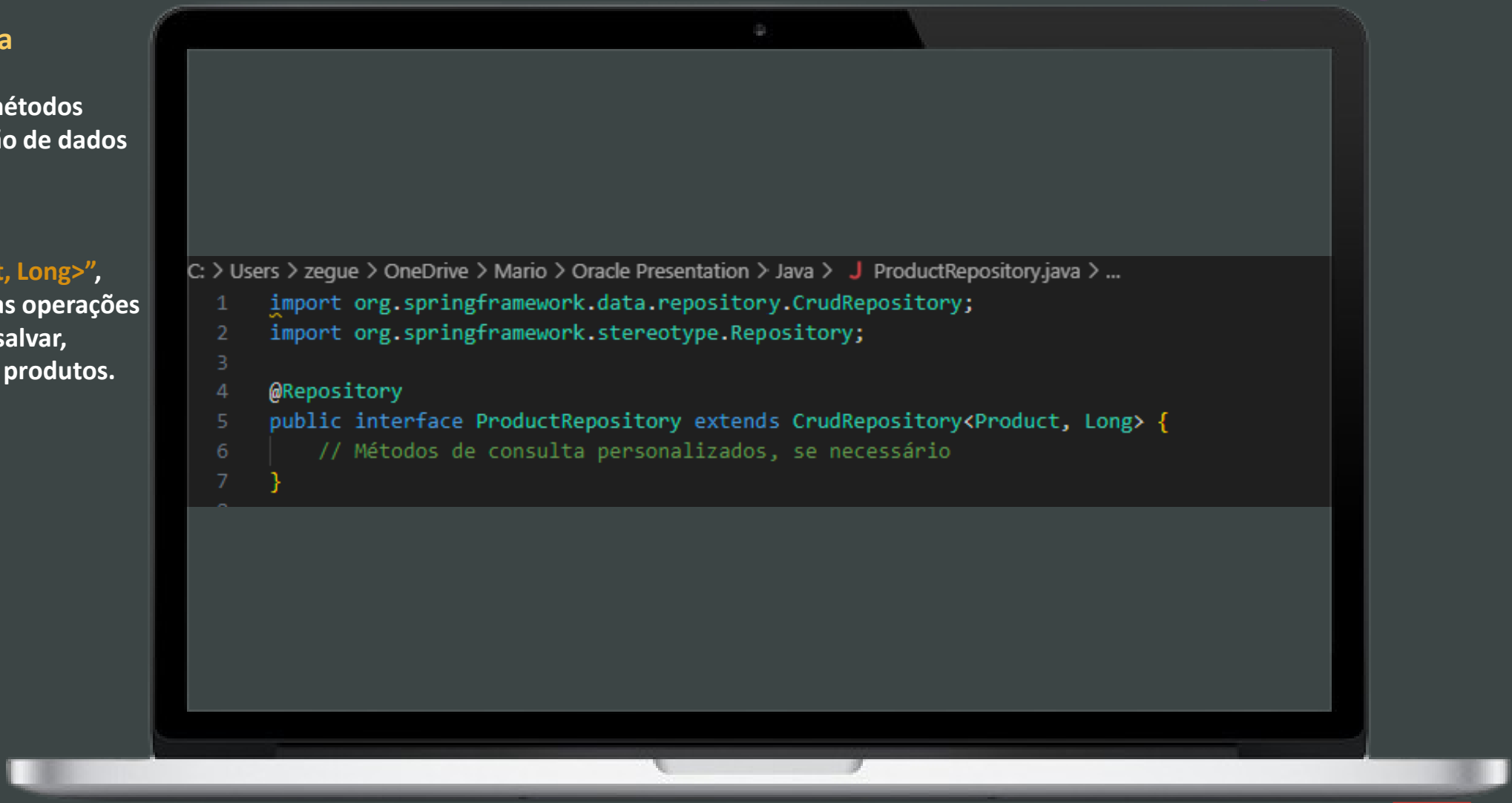


Spring Data JDBC & Autonomous Database JSON (ADJ)

ProductRepository.java

Esta interface define os métodos para acesso e manipulação de dados na tabela produtos.

Ele estende a interface "CrudRepository<Product, Long>", herdando métodos para as operações CRUD básicas, tais como salvar, buscar, atualizar e excluir produtos.



Spring Data JDBC & Autonomous Database JSON (ADJ)

ProductService.java

Esta classe contém toda a regra de negócios para manipulação de dados relacionados aos produtos.

Ela inclui os métodos para inserir um novo produto (insertProduct) e buscar todos os produtos (findAllProducts).

Toda a interação com o Autonomous Databa JSON é feita em conjunto com a "ProductRepository".

C: > Users > zegue > OneDrive > Mario > Oracle Presentation > Java > J ProductService.java > ...

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class ProductService {

    @Autowired
    private ProductRepository productRepository;

    // Método para inserir um novo produto no banco de dados
    public void insertProduct(Product product) {
        productRepository.save(product);
    }

    // Método para buscar todos os produtos no banco de dados
    public List<Product> findAllProducts() {
        return (List<Product>) productRepository.findAll();
    }
}
```

Spring Data JDBC & Autonomous Database JSON (ADJ)

Application.java

Esta é a classe principal da aplicação Spring Boot.

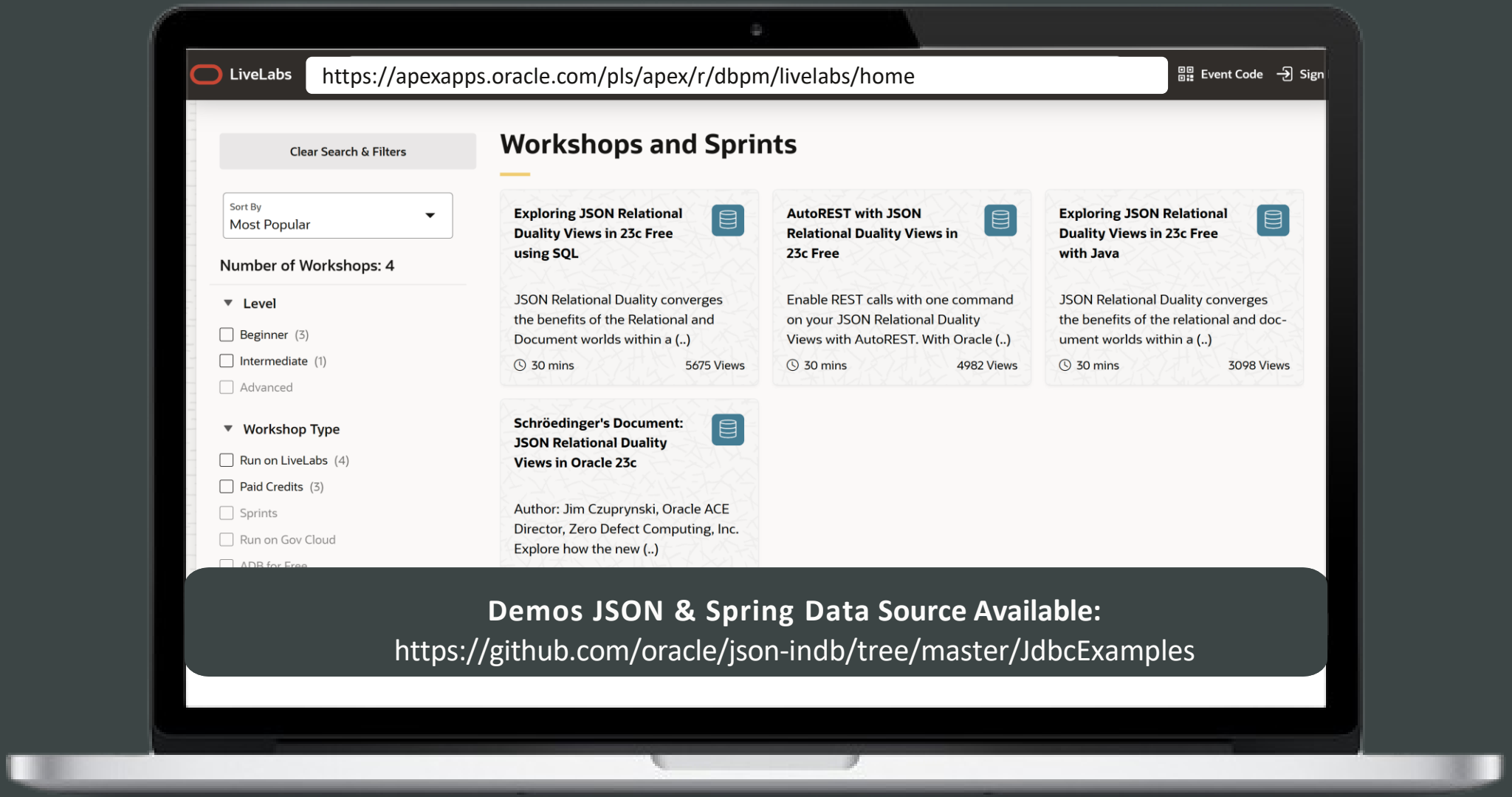
Ela inicia a aplicação Spring Boot e implementa a interface "CommandLineRunner" que permite a execução do código após o carregamento do contexto da aplicação.

No método "run", são usados os métodos "insertProduct" e "findAllProducts" que foram definidos em "ProductService".

C: > Users > zegue > OneDrive > Mario > Oracle Presentation > Java > Application.java > ...

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application implements CommandLineRunner {

    @Autowired
    private ProductService productService;

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        // Criar e salvar um novo produto
        Product product1 = new Product();
        product1.setName("Produto A");
        product1.setPrice(10.0);
        productService.insertProduct(product1);

        // Criar e salvar outro produto
        Product product2 = new Product();
        product2.setName("Produto B");
        product2.setPrice(20.0);
        productService.insertProduct(product2);

        // Buscar todos os produtos e imprimir
        System.out.println(x:"Todos os produtos:");
        productService.findAllProducts().forEach(System.out::println);
    }
}
```

LiveLabs & Demos

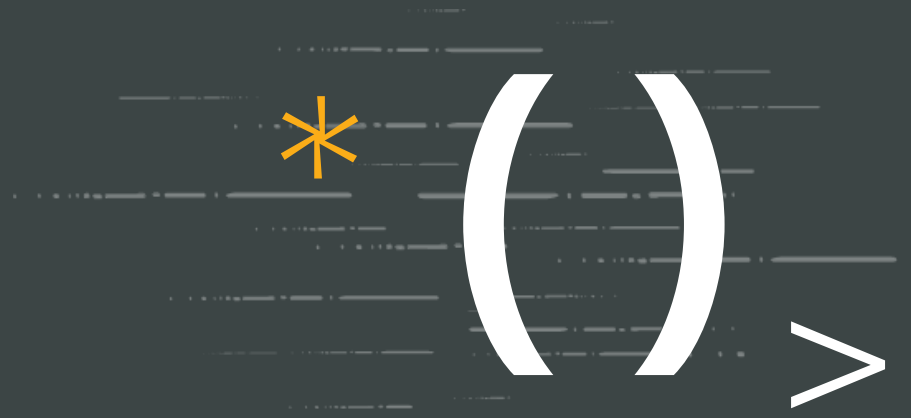


Q&A

Questions?

Mario Barduchi
Sr. Database Systems Engineer
Dell Technologies, DCWS US & LATAM
mario.barduchi@dell.com

Thank you



Mario Barduchi
Sr. Database Systems Engineer
Dell Technologies, DCWS US & LATAM
mario.barduchi@gmail.com


ORACLE