ORACLE

多模数据库与微服务应用

Oracle Database 19c

刘志宇

高级解决方案工程师

Safe harbor statement

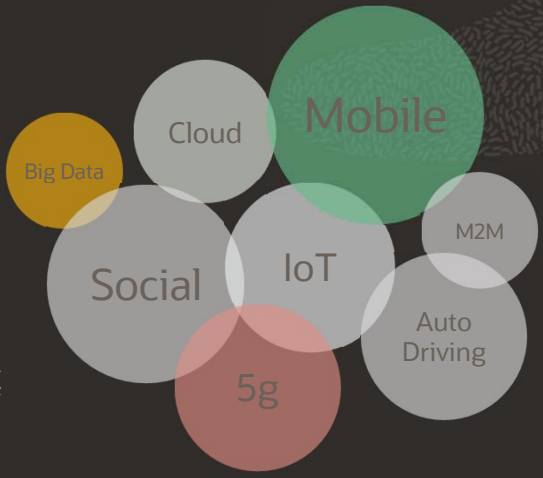
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, 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.

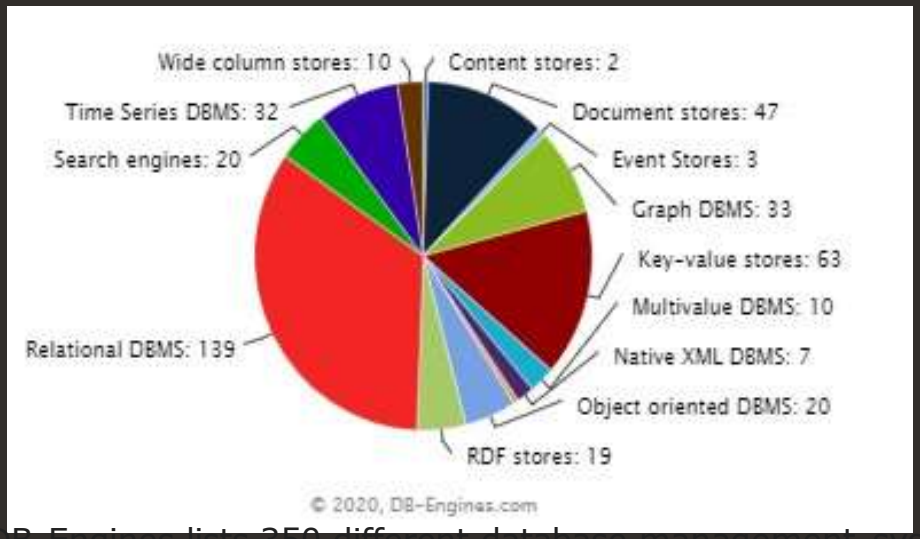
议程

- 为什么需要多模数据库
- Oracle多模数据库概览
- 如何利用Oracle数据库支持微服务架构
- Lab

企业数据应用的变化

- 现代应用程序需求的多样性：
  - 数据类型 –关系型，文档，空间，图文档、XML、JSON、多媒体、web 内容，以及诸如卫星图像和医学影像、地图和地理信息、传感器数据及图形结构等专业信息
  - 应用程序类型 –交易，分析，关系，地理位置，机器学习，物联网等





DB-Engines lists 350 different database management systems

The future is:

~~NoSQL Databases~~

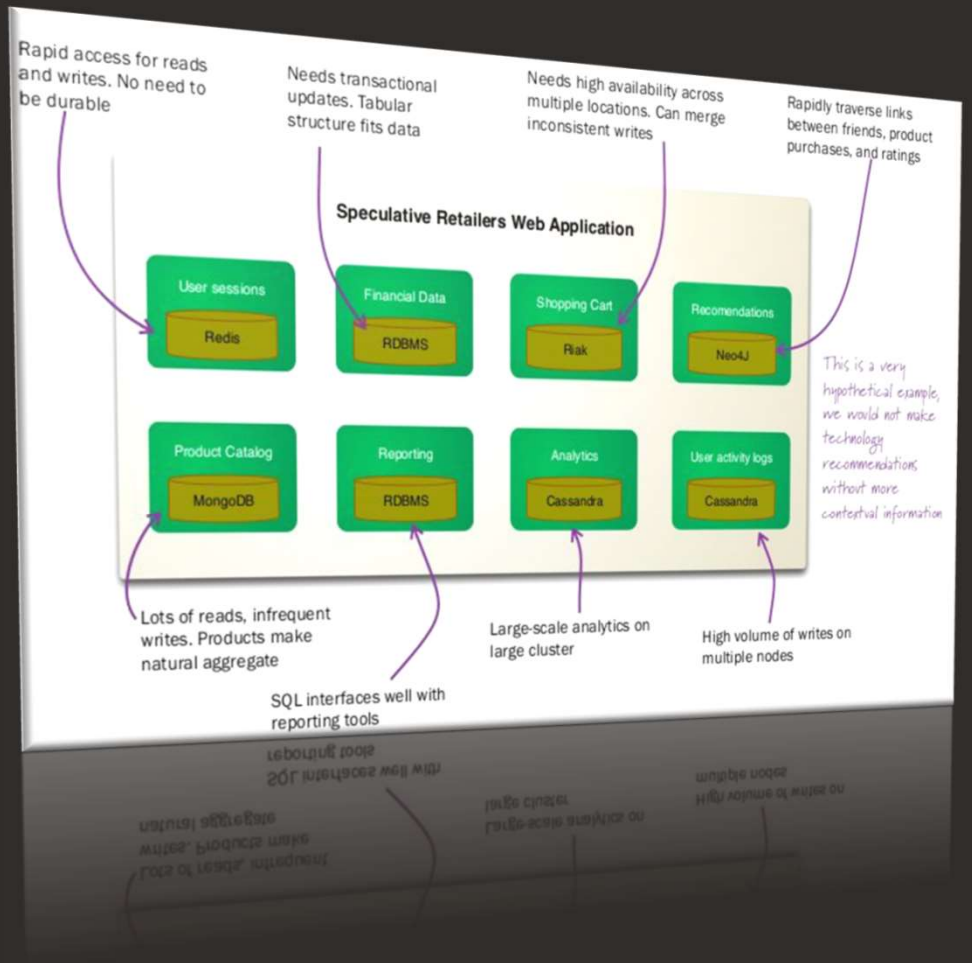
Polyglot Persistence

Martin Fowler

"Polyglot persistence will occur over the enterprise as different applications use different data storage technologies. It will also occur within a single application as different parts of an application's data store have different access characteristics."

Martin Fowler & Pramod Sadalage, Feb. 2012
http://martinfowler.com/articles/nosql-intro-original.pdf

多语言持久化 - Polyglot Persistence



- ✓ 多语言久化的思路是指，用户根据场景的不同需求分别选择使用合适的数据库，这样在一个完整的系统中，可能同时运行着多种不同的数据库。
- ✓ 多语言持久化一个显著的优点就是单一模块的性能提升，但缺点也同样的显而易见：以增加复杂性和学习成本为代价，在部署、使用及维护上带来了挑战。

什么是多模数据库 – Multimodel Database

多模型数据库 (Multi-model database), 是下一代新型数据库，与传统的数据库系统只支持单一数据模型不同，多模型数据库是一种在统一、综合的平台下同时支持多种不同的数据模型的数据库，这些数据模型可包括传统的关系模型和NoSQL数据模型 (文档模型，键值模型，图模型)，一个重要的特性是，多模型数据库拥有自己的一种或多种查询语言，可以非常灵活的方式访问多种不同数据模型，甚至是跨模型的JOIN操作，这使得数据组织，存储，操作较以往更加灵活，便捷。

一个标准的应用程序接口，一个安全模型，一套升级、恢复和可伸缩性程序，等等。

Oracle多语言持久性产品策略

支持两种模式 – 客户可依照场景来选择

单模式

- Oracle支持多个单模式数据存储
  - 关系型
  - Key/Value
  - XML & JSON
  - 空间
  - 图
  - OLAP

- Oracle 通过Big Data SQL集成了单模 型多语言环境

多模式

- Oracle 数据库支持多模式持久存储
  - 关系型
  - XML & JSON
  - Text
  - OLAP
  - 图 & 空间

- Oracle数据库提供对所有数据库对象 的集成访问

Oracle数据库数据存储模型

JSON

- 支持VARCHAR，CLOB和BLOB类型存储
- SODA API存取JSON
- SQL和PL/SQL统一接口
- Native JSON存储类型
- 快速扩展关系型存储

XML

- 支持所有关键的xml标准如：命名空间，DOM，XQuery，XLST
- SQL和PL/SQL统一接口
- 支持xml类型存储列和表

Graph

- Property Graphs
  - PGX in-memory graph engine
  - PGQL graph query language
  - 50+ Graph algorithms
  - Ease of development with rich UI
- RDF graph
  - Support for W3C and OGC standards
  - SPARQL graph query language
  - Java APIs via Apache Jena

Spatial

- 向量数据的空间对象类型，空间索引类型，数百个空间运算符和函数
- 全面支持坐标系与转换
- 原生的3D数据模型支持
- 支持partitioning, in-memory, distributed transactions, sharding
- 基于标准的 SQL, Java API, JSON和REST

无需额外的许可费用 FREE

如何利用Oracle数据库支持微服务架构

单体应用架构面临的挑战

- **可靠性差**：如果系统的某一功能无法正常工作，则整个系统将无法正常工作，缺乏故障隔离。
- **难于扩展**：在对应用程序横向扩展时不同模块对资源的需求可能是相互冲突的，服务器必须满足所有模块需求，某一模块负载大需要扩展整个应用。
- **难于快速交付**：从代码的提交到部署生产环境时间越来越长，需要多个协调多个团队项目。
- **过度复杂**：整个系统过于庞大和复杂以至于任何一个开发者很难理解全部，修复BUG或开发新功能变得困难耗时。
- **技术栈锁定**：必须长期使用一套相同技术栈，向后兼容性也是一项挑战。
- **开发缓慢**：由于系统过于复杂开发人员在本地构建并运行单体应用需要花费大量时间调试。

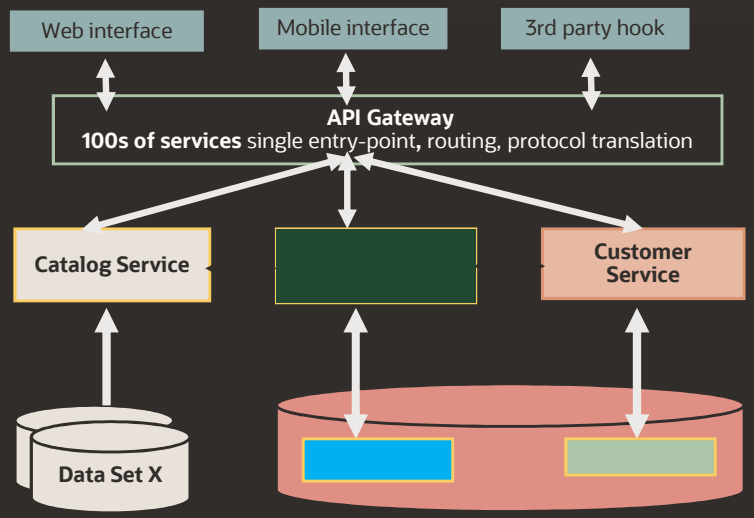
微服务架构的收益与挑战

收益

- 使大型的复杂应用程序可以持续交付和持续部署
- 每个服务都相对较小并容易维护
- 服务可以独立部署
- 服务可以独立扩展
- 微服务架构可以实现团队的自治
- 更容易实验和采用新技术
- 更好地容错性

挑战

- 服务的拆分和定义是一项挑战
- 分布式系统带来的各种复杂性，使开发、测试和部署变得困难
- 管理数据的强一致性变得困难，不得不采用最终一致性方案
- 大量的服务使得运维管理变得越来越复杂
- 可能需要引入额外消息中间件，增加整体系统复杂性



Oracle数据库哪些特性可以支持微服务架构

- 创新的多租户架构允许微服务具有单独的数据库(PDB)，但仍将它们作为一个数据库(CDB)进行管理
- Oracle Advanced Queuing消息队列为微服务提供事务性消息传递，不需要引入额外的消息队列
- Oracle Rest Data Service快速开启数据的REST APIs
- 真正的多模型引擎，支持关系，OLAP，JSON，XML，图形，空间
- 物化视图，多分片和跨PDB查询支持实时分析
- 业界领先的安全性，可伸缩性，分析和高可用性

利用Oracle多租户特性支持微服务架构

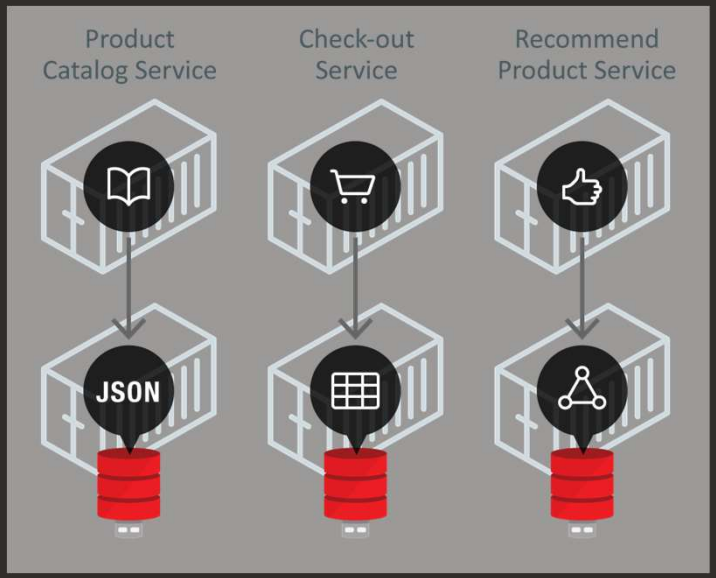
为开发新应用减少成本和数据管理的复杂性,提高敏捷性

利用PDB作为独立数据库

- 给每个微服务分配它自己的数据库 (can be a PDB, Schema, or a subset)
- 每个微服务可以有自己的数据模型如 Text, JSON, xml, graph

多租户容器数据库提供

- 统一管理多个数据库 (Patch, Upgrade, HA, Backup)
- PDB中的数据被安全的隔离
- 便捷的夸PDB分享与查询数据
- 利用克隆快速生成新的PDB库
- 快速迁移到新CDB
- 精细化控制如安全策略、资源分配等
- 分享内存和后台进程最大化利用资源



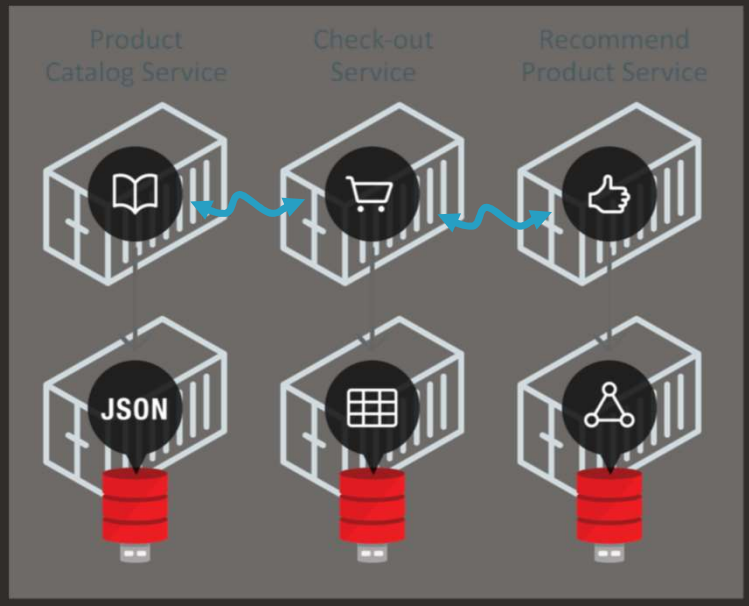
Oracle Advanced Queuing

Transactional Event Queuing System for Microservices (21c)

Oracle高级队列是基于数据库表实现的，所以数据库的高可用性，可伸缩性和可靠性的所有操作优势也适用于队列数据。

微服务可以利用Advanced Queuing发送/接收消息

- 支持 Point-To-Point模式和Publish/Subscribe 模式
- Persistent messaging, Buffered Messaging
- JMS+AQ 提供稳定丰富的异步消息能力
- 消除了以REST同步方式消息数据交互的紧耦合



分布式事务跨数据库表和消息不需要两阶段提交，DML操作和消息在同一本地事务

System-Level Access Control，Queue-Level Access Control

通过Messaging Gateway整合IBM WebSphere MQ

事务会话确保没有消息丢失和精准一次送达

高性能: 12c Sharded Queues 新架构对比Classic AQ提升10倍吞吐量

AQ支持Saga模式的分布式事务，支持Kafka Client API

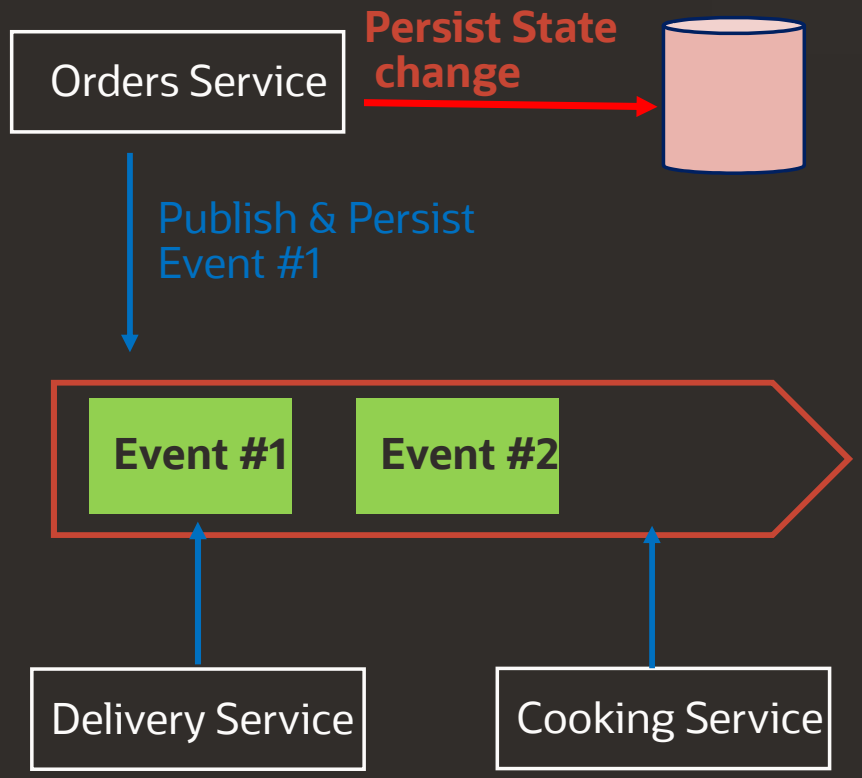
消息持久化与数据状态的原子性挑战

几种不同技术：

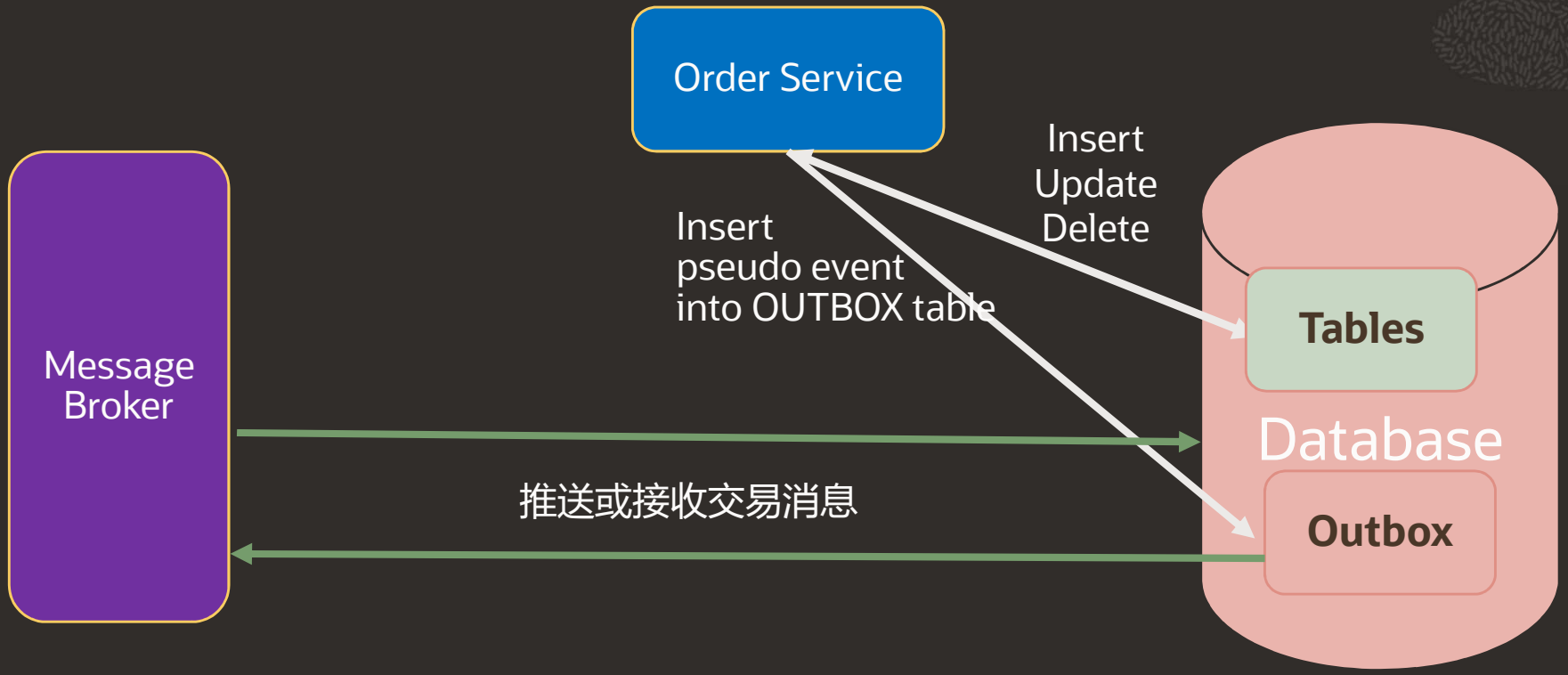
数据库表作为消息队列

Outbox Table

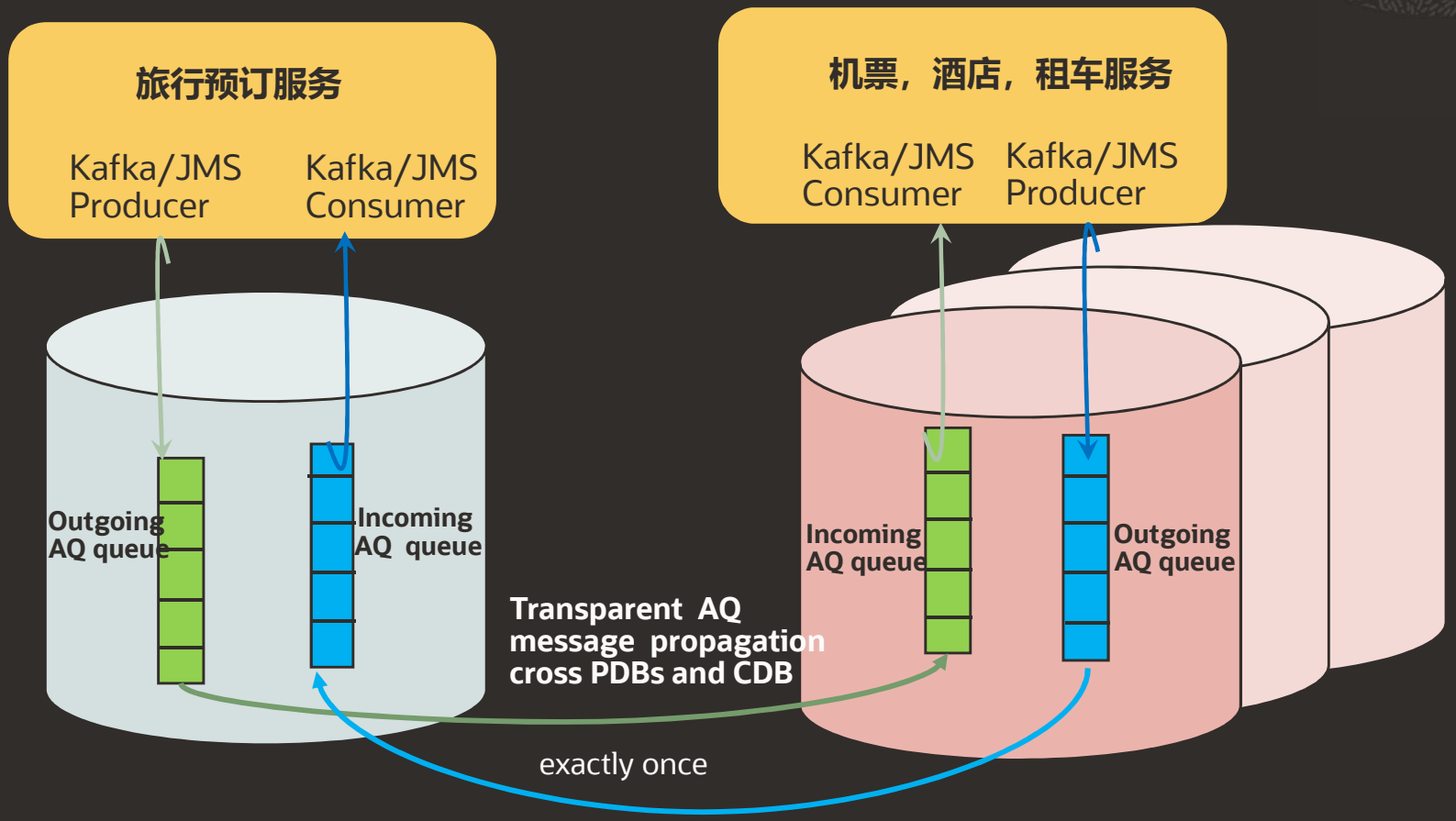
Oracle Advanced Queue



Outbox Table模式



每个微服务可以通过AQ传播事件到其他微服务中



Order inserted in table and AQ message sent in same local transaction

OracleDBOrderService.java

```
172 @    public static javax.jms.Queue insertOrderAndSendMessage(String message) {
173          try {
174              System.out.println("-----> OracleDBController insertOrderAndSendMessage enter");
175              OracleDataSource aqDataSource = getAQDataSource();
176              QueueConnectionFactory q_cf = AQjmsFactory.getQueueConnectionFactory(aqDataSource);
177              QueueConnection q_conn = q_cf.createQueueConnection();
178              Session session = q_conn.createQueueSession(true, Session.CLIENT_ACKNOWLEDGE);
179              Connection dbConnection = ((AQjmsSession) session).getDBConnection();
180              System.out.println("-----> OracleDBController insertOrderAndSendMessage dbConnection:" +
181              dbConnection.createStatement().executeUpdate(
182                      "insert into orders values ('" + message+ "', 'myinventoryid1', 1, 'mystatus1') "
183              Queue queue = ((AQjmsSession) session).getQueue(queueOwner, queueName);
184              ((AQjmsDestination) queue).start(session, true, true);
185              QueueSender sender = ((AQjmsSession) session).createSender(queue);
186              Message msg = session.createTextMessage(message);
187              sender.send(msg);
188              session.commit();
189              System.out.println("-----> OracleDBController insertOrderAndSendMessage committed:" +
190                      message);
191              session.close();
192              q_conn.close();
193              return queue;
194          } catch (Exception e) {
195              e.printStackTrace();
196              return null;
197          }
198      }
```

ORDS快速开启你的数据REST服务

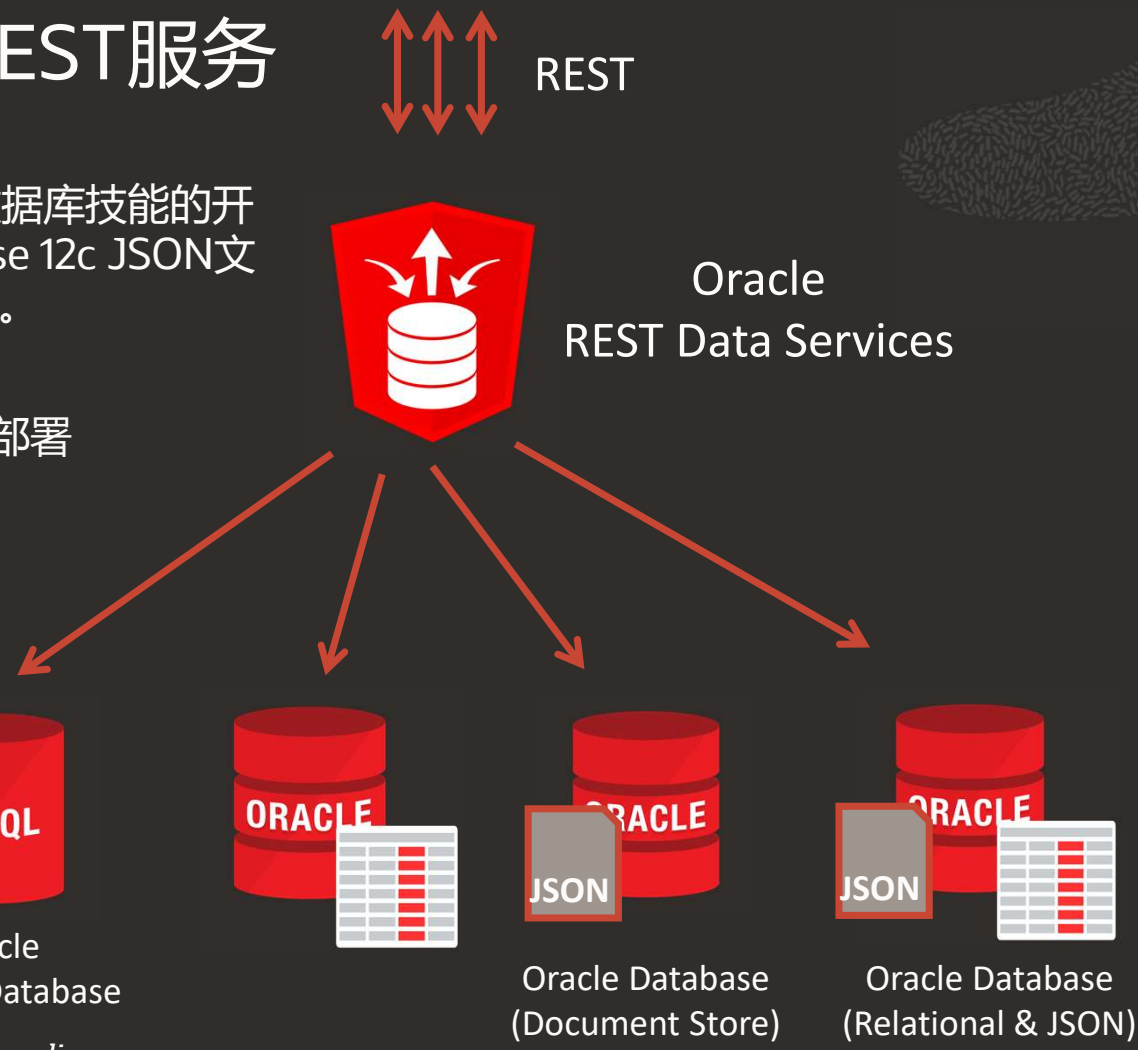
ORDS是一个Java应用程序，使具有SQL和数据库技能的开发人员能够为Oracle数据库，Oracle Database 12c JSON文档存储和Oracle NoSQL数据库开发REST API。

支持WebLogic，Tomcat， Jetty嵌入式独立部署
支持云端和本地环境

Auto REST for Tables

Custom REST Service for SQL/PLSQL

支持Basic Auth 与 OAuth2.0安全认证

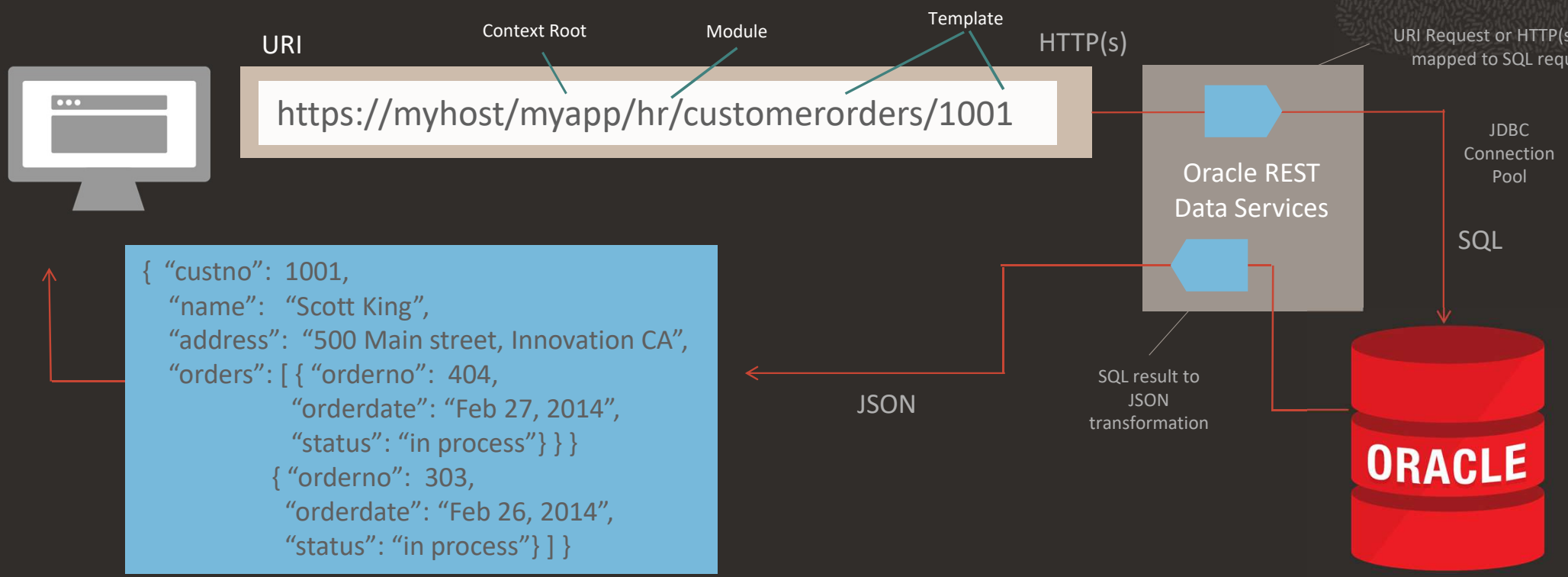


FREE

** No additional cost with any Oracle Database license*

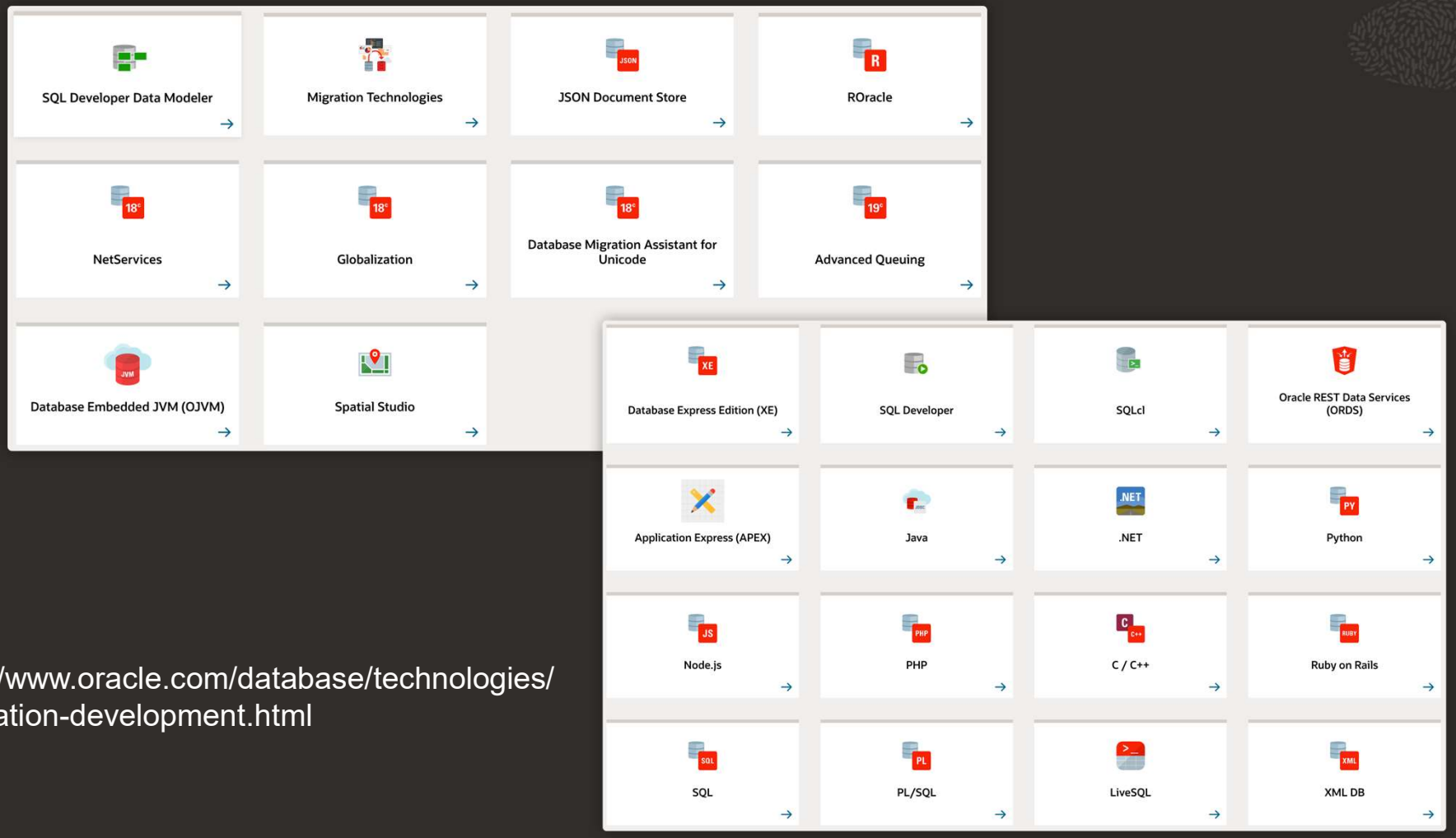
ORDS - 数据请求响应机制

HTTP(s) API App-Dev with Relational Tables in Oracle Database



ORDS maps standard URI requests to corresponding relational SQL (not schemaless): e.g. SQL SELECT from customers and orders table.
ORDS also transforms the SQL results into the highly popular JavaScript Object Notation (JSON), other formats include HTML, binary and CSV.
Fully committed to supporting any and all standards required by Fusion / SaaS / FMW; we are actively engaged in the ongoing dialog.

Oracle 数据库支持丰富的应用开发工具

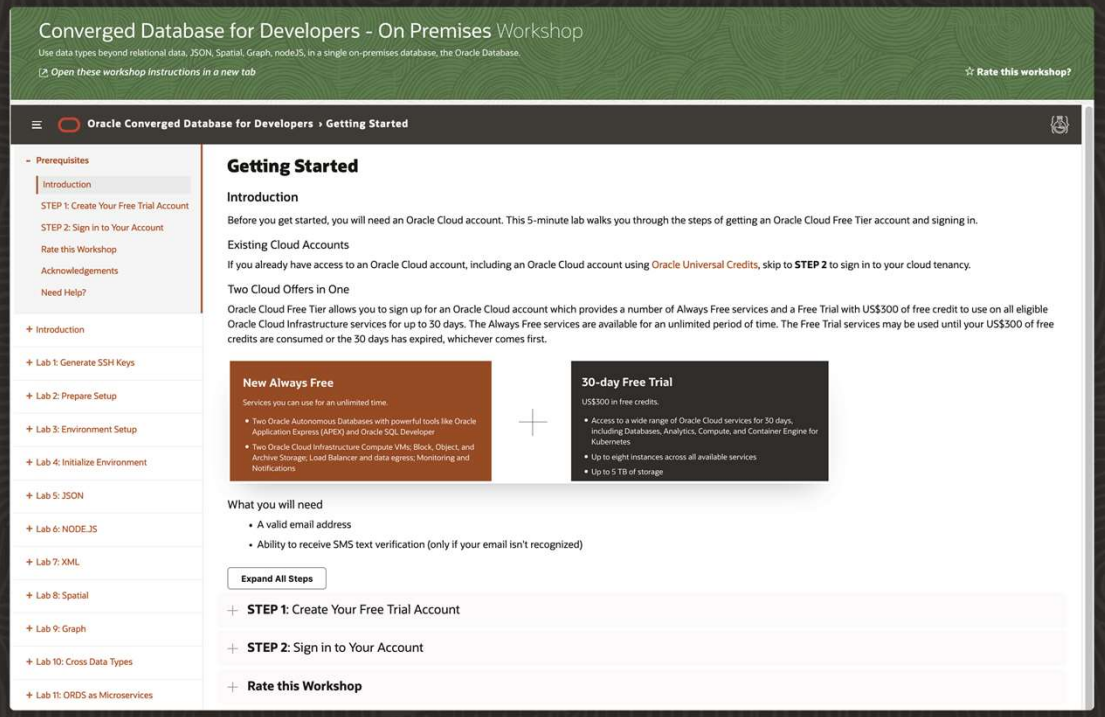


https://www.oracle.com/database/technologies/application-development.html

Lab

I. 利用Sql 操作 JSON, Spatial数据类型

II. 开发基于ORDS RESTful服务



https://oracle.github.io/learning-library/developer-library/converged-db/converged-db-on-premises/workshops/freetier/

Thank you