ORACLE

Oracle JSON 数据库介绍

SE-Hub DM Team
Xu Guang

Oracle 免责申明

以下内容旨在概述 Oracle 产品的总体发展方向。该内容仅供参考，不可纳入任何合同。本文档不承诺提供任何材料、代码或功能，也不应将其作为购买决策的依据。

此处所述有关 Oracle 产品任何特性或功能的开发、发布、时间安排和定价可能会发生变更，且均由甲骨文公司自行决定。

Agenda

1. 多种数据交换格式及JSON的兴起
2. Oracle 多模数据库与JSON
3. 面向开发人员与DBA
4. Oracle JSON数据库优势
5. 客户案例介绍
6. Why Oracle JSON DB?
7. Q&A

多种数据交换格式及JSON的兴起

JSON的诞生

现今存在的流行数据格式

JSON 诞生历史

诞生于 2001年的数据交换格式 by Douglas Crockford
JSML （JavaScript Message Language）
JavaScript Object Notation

刚开始的反应

- 从来未听说过
- 我们只用XML 交换数据
- 不够标准

JSON.org （设计原则）

- Minimal
- Textual
- Subset of JavaScript
- Free and unencumbered

RFC 4627: MIME Media Type
application/json

Delivered to a hidden frame

```
<html><head><script>
document.domain = "fudco.com";
parent.session.receive({
    to:"session",
    do:"test",
    text:"Hello world"
});
</script></head></html>
```

```
{"firstName": "John"}
```

JSON 看起来更像数据

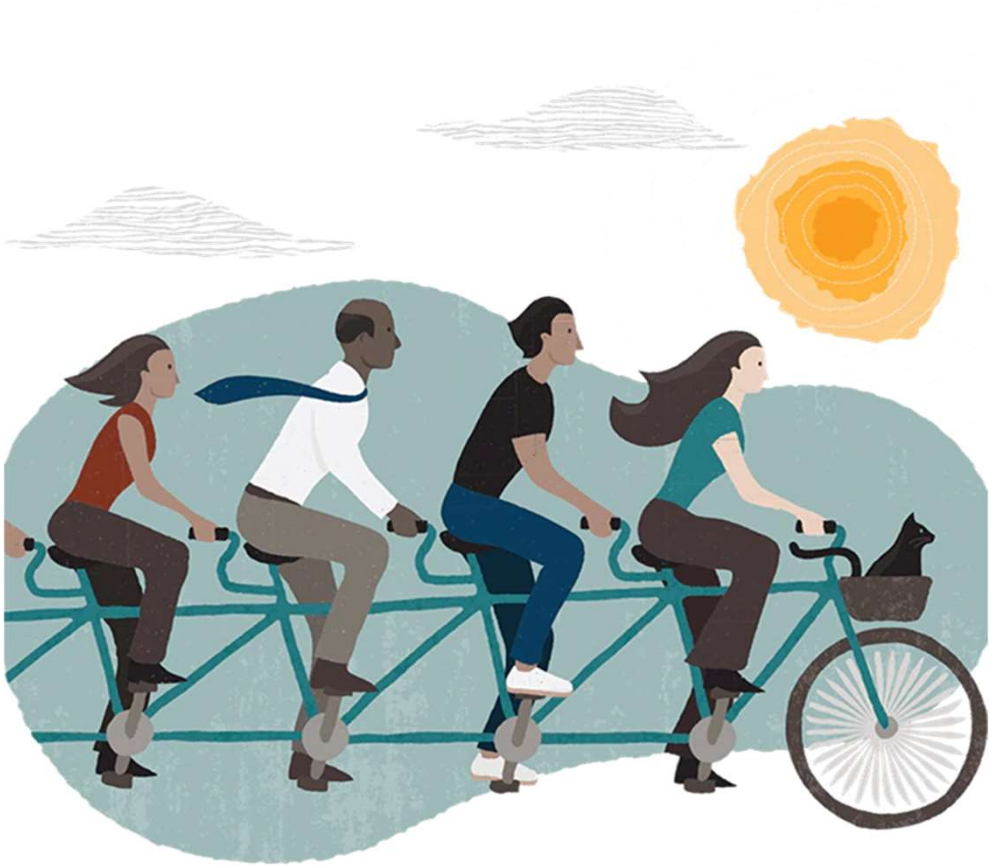
- 基本类型的value更贴近变成语言中的类型
- 编程语言易于解析
- JSON Object可映射一条记录，结构，对象等等
- JSON Array可映射array，vector，sequence, list

JSON 的版本 （就是没有版本）

- 非常简单
- 非常稳定

JSON的大规模使用

JSON vs XML



JSON 诞生历史-流行原因

Ajax (Asynchronous JavaScript and XML) 的出现

1996, 在IE5中开始支持iframe，并提供XMLHttpRequest scripting对象

(2004年和2005年在Gmail和Google Maps)

2005 Jesse James Garrett 发表了一篇文章第一次提到Ajax

A New Approach to Web Applications

2006 W3C 发布第一个规范草案

Web 2.0 – 单页面应用程序

SPA无需在每次用户交互时重新加载整个页面，
而是在浏览器中动态重写当前页面

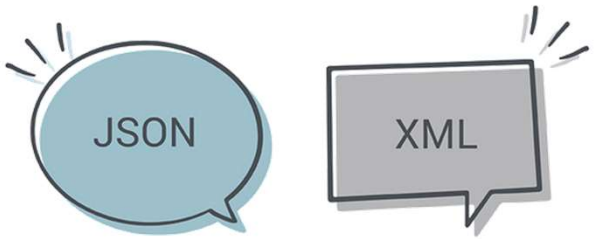
JSON 与 XML 比较

JSON

- 诞生之初就是用于数据交换
- 对于人/机都易于阅读
- 数据结构对于编程语言友好
  - Object
  - Array
  - String
  - Number
  - Boolean(true/false)
  - Null
- 弱约束（约束-需要第三方开源Lib）
- 尚无标准化的schema定义

XML

- 用于数据表示交换
- 对于人/机都易于阅读（有时冗长）
- 数据表述需要schema定义
- 强约束（基于schema）
- 标准化组织支持



常用的企业数据格式

企业数据交换及存储

企业级应用数据交换及存储格式

企业级应用特点
- 稳定健壮
- 业务逻辑复杂
- 数据结构复杂
- 强调数据一致性
- 系统集成要求高（异构系统多）

XML作为数据交换格式
- 具有统一标准
- 可以描述复杂结构
- 自约束
- 数据本身验证与应用解耦
- 非常适合于系统之间交互的契约格式

JSON作为数据交换及存储格式
- 灵活
- 简单易用
- 可以描述复杂结构
- 数据本身验证应用耦合

为什么要把JSON存数据库

JSON是将业务对象映射到其中的一种很好的格式，大多数编程语言都支持JSON

- **1）JSON 无 Schema**
  - 可以随意添加属性，结构
  - 无需更改数据库即可完成应用程序更改
  - 客户端/应用程序的更新可以解耦
- **2）JSON 可以避免规范化操作（表的分解/连接）**
  - 仅对一个字段的 insert/update/delete 就相当于一个业务的对象的操作
  - 低延时的插入/查询操作，提高复杂模式的性能
- **3）支持文档存储类型的document-store 编程接口(optional)**
  - 基于JSON文档和集合的编程模型
  - 不需要SQL就能完成简单的 CRUD 操作
  - 减少错误，提高生产性

关于模式灵活性的注意事项

- 没有绝对的No-Schema，Schema隐式的存在于数据库层面或应用程序代码中
- **JSON 不会替代关系型数据!**

Oracle 融合数据库与JSON

多个专有数据库 vs 融合数据库

多个单一用途数据库 vs. 一个融合数据库

数据策略的不同选择

Amazon 等一些数据库供应商 策略

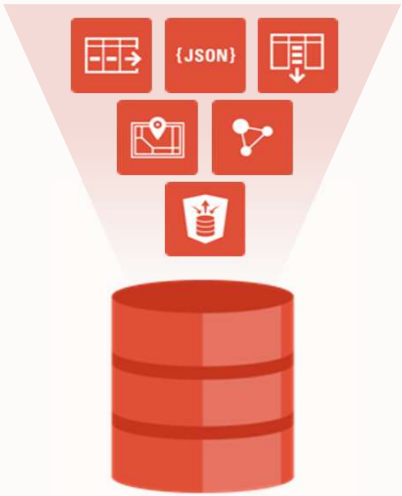
针对每种数据类型和工作量运行

单一用途 数据库

Oracle 策略

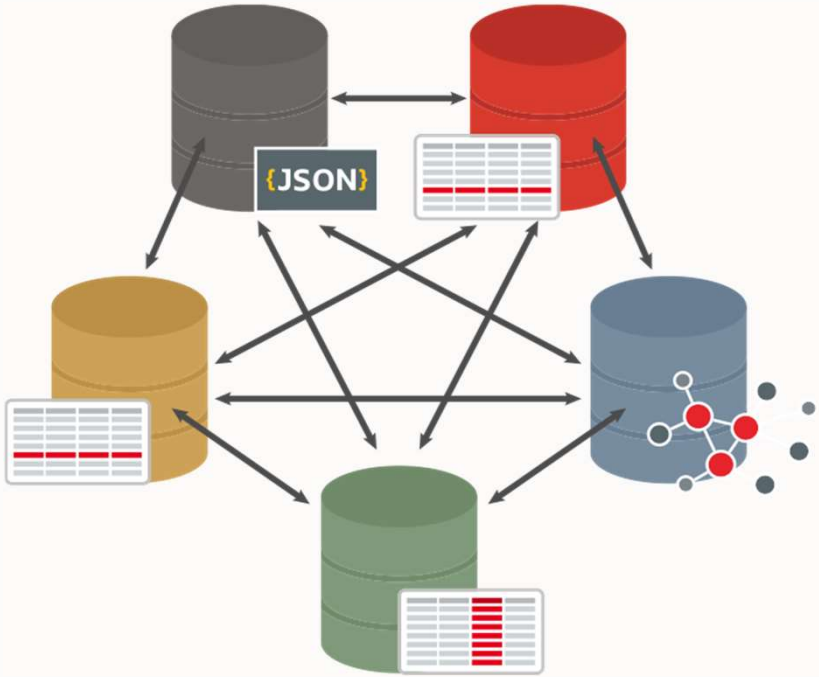
针对多种数据类型和工作负载运行

融合开放式 数据库

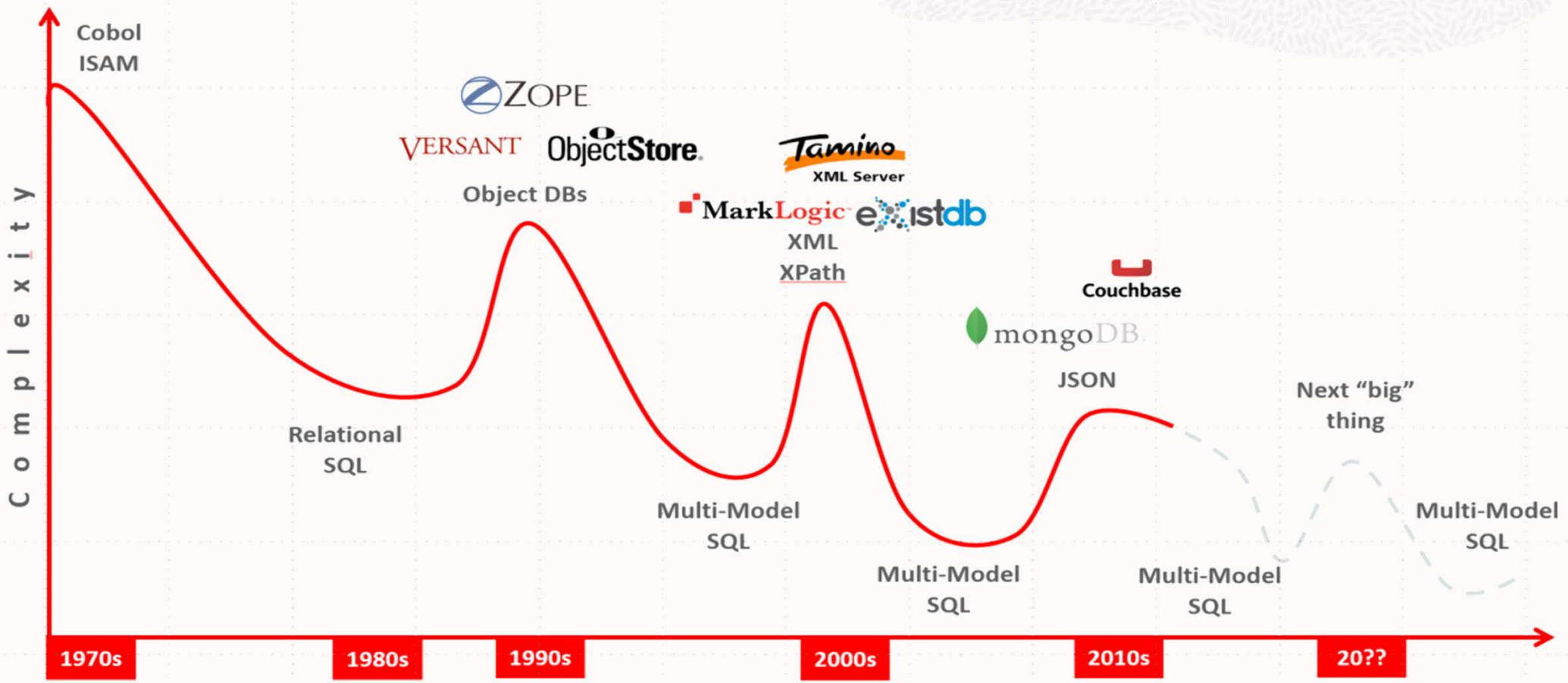


多个专用数据库导致数据碎片化

- 部署的多个单一用途数据库都会使企业数据体系结构碎片化
- 集成碎片化的数据使分析和应用程序开发极为复杂
- 多种类型的单一数据库会带来更多的运营和安全复杂性以及风险



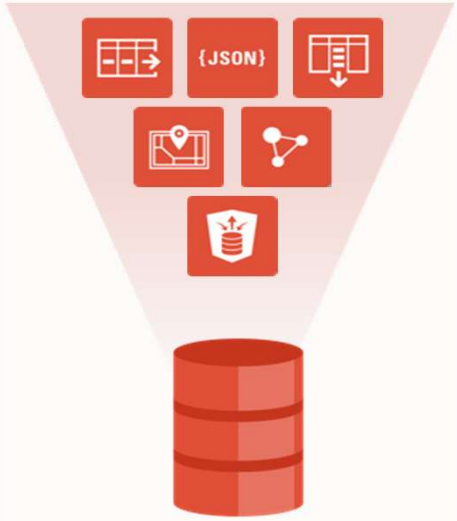
融合数据库的发展时间线



Oracle 数据库业界所处位置

支持融合数据库的更多创新

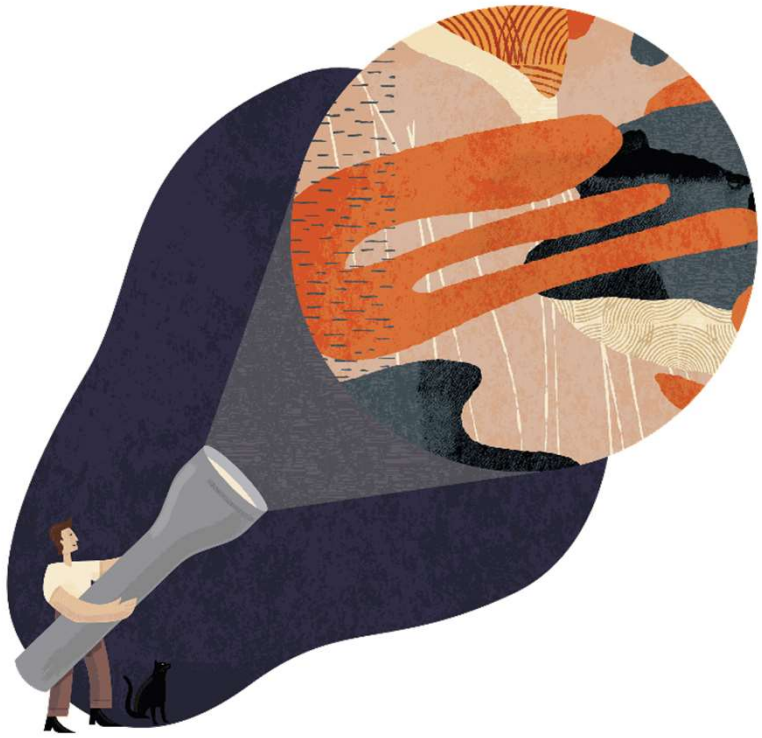
21c



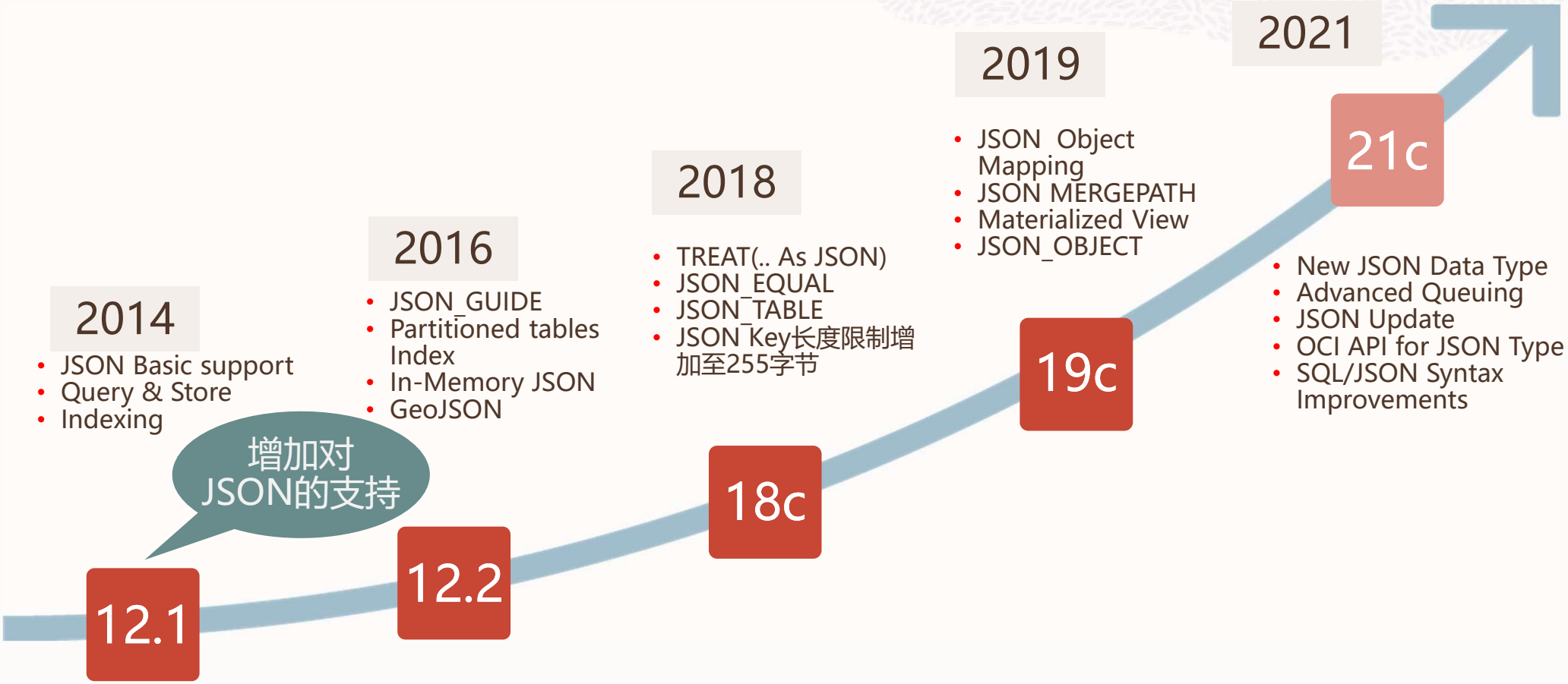
- 原生JavaScript
- Spatial
- JSON
- In-Memory 改进
- 多租户Data Guard
- SQL 宏
- 持久化内存存储
- 原生区块链表
- Graph
- 自动机器学习
- Sharding 增强
- 多租户安全
- 安全增强

JSON 数据库的介绍

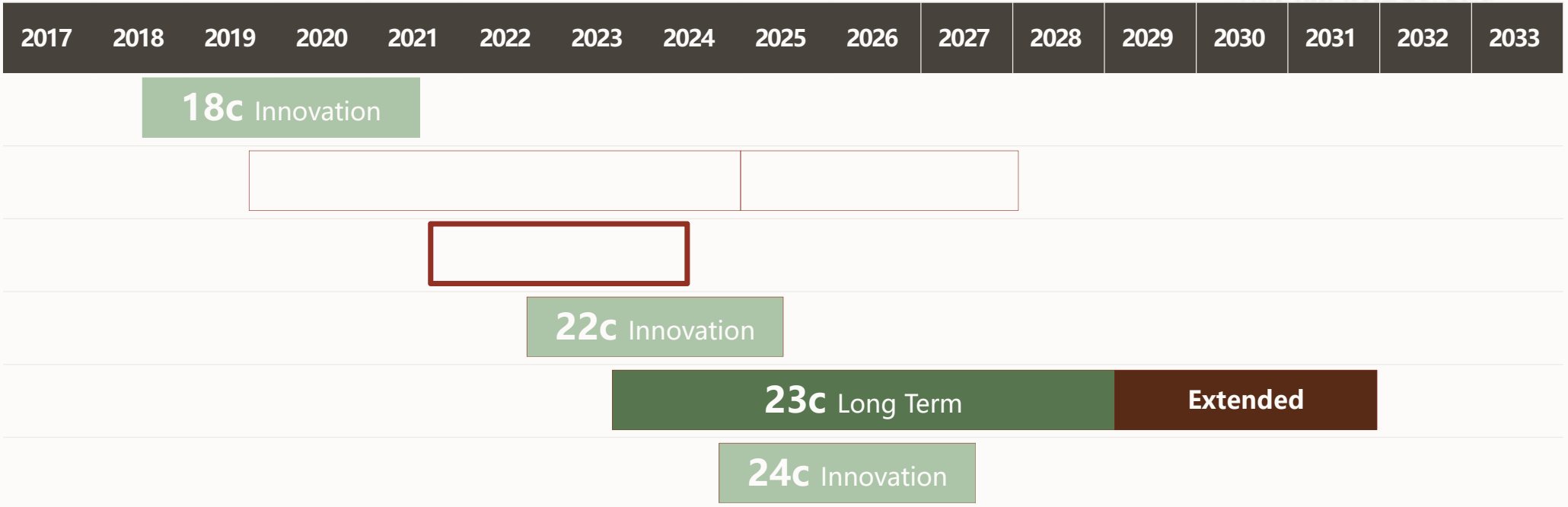
JSON 在 Oracle DB中的特性



Oracle JSON 数据库发展历史



Oracle 数据版本以及迭代周期



- **Innovation Release** - 2 years of Premier Support, and no Extended Support
- **Long Term Release** - 5 years of Premier Support, and 3 years of Extended Support

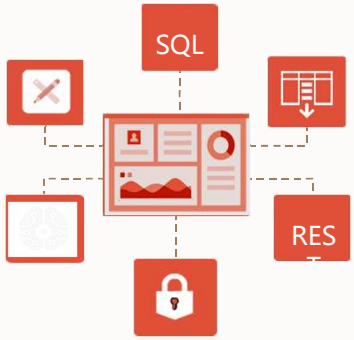
Oracle JSON 数据库

Existing Oracle DB & DB Cloud Service

只要安装Oracle数据库就能使用JSON并且
无需支付额外费用，无论是

- 在本地安装的 Oracle 数据库
- Exadata
- 云上的DBaaS

AJD 全新Oracle Autonomous JSON DB



提供与NoSQL文档存储相同的优势特点

弹性计算和存储

单位毫秒级
低延迟读写

高可用

低廉价格

Oracle 的 JSON 数据支持类型及全新的原生JSON数据类型

可以使用的 JSON Column 类型

VARCHAR2

- 4000 Bytes
- 32767 Bytes

LOBs (> 32767 Bytes)

- BLOB (Oracle 推荐的存储数据类型)
  - CLOB 存在字符集转换（AL32UTF8 > UCS2）
    导致存储容量翻倍
  - BLOB 不存在字符集转换
- CLOB
  - BLOB 以文本形式查看时，需要显示的转换为CLOB
  - 更新数据时，需要显示的将字符串转换为BLOB

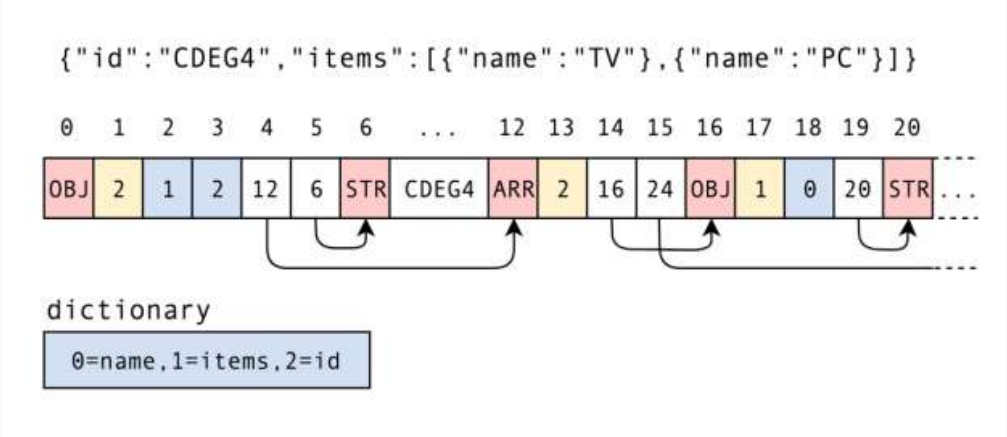
原生 JSON 二进制类型

21c

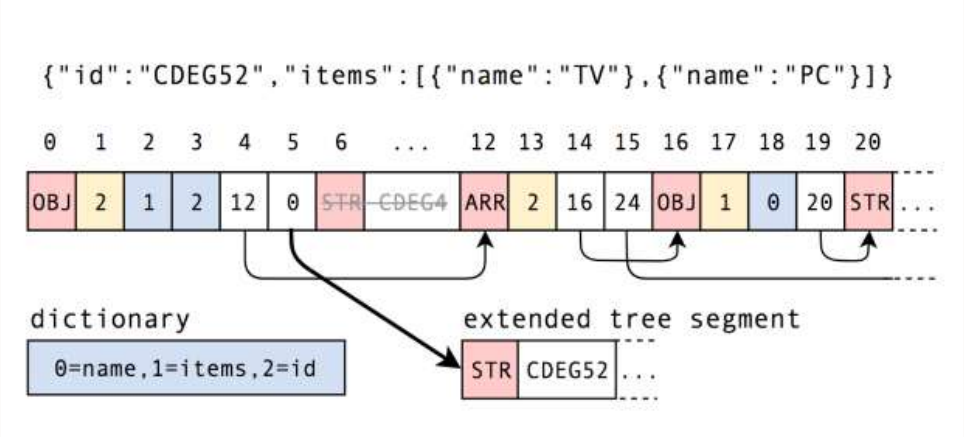
- 同时支持在 SQL, PLSQL中使用
- OCI, JDBC 的原生的支持
- 基于 OSON – 经过优化的二进制表示
  - Self-contained format
  - 快速查找
  - 局部更新
- 扫描速度提升 2x 以上
- 更新速度提升 10x 以上

21c Oracle JSON 原生数据类型存储结构

内部存储结构



更新时的结果



JSON, BSON Vs OSON

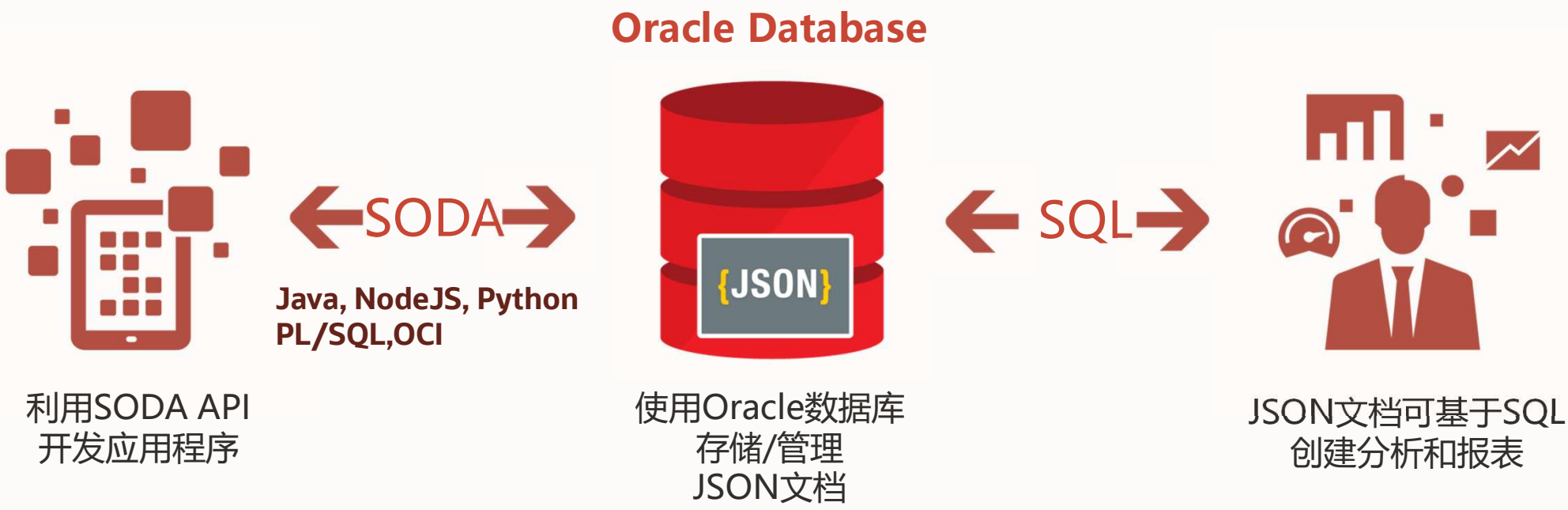
数据压缩比

ID	JSON in UTF8	BSON	OSON	vsUTF8	vsBSON	#Object	#Keys(tot/distinct)	#Array
D1	613	764	524	0.9x	0.7x	20	33/5	1
D2	1,782	1,813	1,950	1.1x	1.1x	4	56/55	0
D3	2,608	3,094	2,160	0.8x	0.7x	26	100/32	14
D4	2,943	3,293	2,476	0.8x	0.8x	46	100/19	14
D5	8,842	8,440	5,591	0.6x	0.7x	38	307/74	29
D6	40,285	37,526	20,486	0.5x	0.5x	81	1,435/246	9
D7	76,861	75,195	38,383	0.5x	0.5x	490	3,300/282	23
D8	141,051	133,307	103,897	0.7x	0.8x	1,688	6,620/40	52
D9	3,374,379	3,303,387	2,167,101	0.6x	0.7x	14,712	112,356/90	12,738
D10	41,548,995	37,352,414	13,801,333	0.3x	0.4x	100,141	1,839,847/73	1

面向开发人员与DBA

面向开发人员与DBA

Oracle 的 JSON 数据访问方式



SODA (Simple Oracle Data Access)

```
SODA allows schemaless application development using the JSON data model.
        SODA create <collection_name>
             Create a new collection

        SODA list
             List all the collections

        SODA get <collection_name> [-all | -f | -k | -klist] [{<key> | <k1> <k2> ... > | <qbe>}]
             List documents the collection
             Optional arguments:
                   -all    list the keys of all docs in the collection
                   -k      list docs matching the specific <key>
                   -klist  list docs matching the list of keys
                   -f      list docs matching the <qbe>

        SODA insert <collection_name> <json_str | filename>
             Insert a new document within a collection

        SODA drop <collection_name>
             Delete existing collection

        SODA count <collection_name> [<qbe>]
             Count # of docs inside collection.
             Optional <qbe> returns # of matching docs

        SODA replace <collection_name> <oldkey> <new_{str|doc}>
             Replace one doc for another

        SODA remove <collection_name> [-k | -klist |-f] {<key> | <k1> <k2> ... | <qbe>}
             Remove doc(s) from collection
             Optional arguments:
                   -k     remove doc in collection matching the specific <key>
                   -klist remove doc in collection matching the list <key1> <key2> ... >
                   -f     remove doc in collection matching <qbe>
```

SQLcl

- 适用Oracle数据库的更易用的SQL命令行开发界面
- 提供
  - 内联编辑，语句完成，命令调用...
  - SODA 命令

SODA – 支持多种语言

Node.js

```
conn = await oracledb.getConnection(…);
db = conn.getSodaDatabase();
col = await
db.createCollection("purchase_orders");
await col.drop();
```

Java

```
OracleClient client = new OracleRDBMSClient();
db = client.getDatabase(jdbcConn);
OracleCollection col =
db.admin.createCollection("purchase_orders");
col.admin().drop();
```

Python

```
conn =cx_Oracle.connect(…);
db = conn.getSodaDatabase();
col = db.createCollection("purchase_orders");
col.drop();
```

PL/SQL (and Oracle Application Express)

```
col :=
dbms_soda.create_collection('purchase_orders');
select
dbms_soda.drop_collection('purchase_orders')
from dual;
```

Oracle 创建 JSON 表

SODA vs. SQL

SQL

```
CREATE TABLE  JSON_DOCUMENTS (
  ID    RAW(16) NOT NULL,
  DATA  BLOB,
  CONSTRAINT JSON_DOCUMENTS_PK PRIMARY KEY
  (ID),
  CONSTRAINT JSON_DOCUMENTS_JSON_CHK CHECK
  (DATA IS JSON)
);
```

SODA (java source code)

```
OracleClient client = new OracleRDBMSClient();
db = client.getDatabase(jdbcConn);
OracleCollection col =
db.admin.createCollection("JSON_DOCUMENTS");
```

表结构

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	RAW	No	(null)	1	(null)
2	DATA	BLOB	No	(null)	2	(null)

表结构

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	VARCHAR2(255 BYTE)	No	(null)	1	(null)
2	CREATED_ON	TIMESTAMP(6)	No	sys_extract_utc(SYSTIMESTAMP)	2	(null)
3	LAST_MODIFIED	TIMESTAMP(6)	No	sys_extract_utc(SYSTIMESTAMP)	3	(null)
4	VERSION	VARCHAR2(255 BYTE)	No	(null)	4	(null)
5	JSON_DOCUMENT	BLOB	Yes	(null)	5	(null)

在Oracle中使用JSON (面向开发人员)

SODA metadata- customization

```
{
   "schemaName" : "mySchemaName",
  "keyColumn" :
   {
      "name" : "ID",
      "sqlType" : "VARCHAR2",
      "maxLength" : 255,
      "assignmentMethod" : "UUID"
   },
   "contentColumn" :
   {
      "name" : "JSON_DOCUMENT",
      "sqlType" : "BLOB",
      "compress" : "NONE",
      "cache" : true,
      "encrypt" : "NONE",
      "validation" : "STRICT"
   },
   "versionColumn" :
   {
     "name" : "VERSION",
     "type":"String",
     "method":"SHA256"
   },
   "lastModifiedColumn" :
   {
     "name":"LAST_MODIFIED"
   },
   "creationTimeColumn":
   {
      "name":"CREATED_ON"
   }
}
```

根据业务自定义

- Table primary key column 的类型
- Content 字段的 Type
- Version 字段的 Type
- .....

Collection 可以与普通的table进行Join查询

简单的 SQL 查询语法（面向开发人员）

Field Access

```
SQL> select j.PO_DOCUMENT
  2         from J_PURCHASEORDER j
  3         where j.PO_DOCUMENT.PONumber = 1600;
```

Collection unnesting

```
 SQL> select *
  2          from CUSTOMER NESTED jcol.orders.lineitems[*]
  3                 COLUMNS (lineid, quantity, prodid, upc, comments);
```

JSON Generation

```
SQL> select JSON_OBJECT(c.jcol.orders.lineitems FORMAT JSON)
   2         from CUSTOMERS c;
```

SQL/JSON 操作说明（面向开发人员）

SQL Operators	JSON Functions	Returns
SELECT	JSON_QUERY()	Fragments of a JSON document
	JSON_VALUE()	Scalar value
	CASE JSON_EXISTS()	False if no JSON value is matched
FROM	JSON_TABLE() (row source)	Specific JSON data projected to relational columns and new rows in case of JSON array
WHERE	JSON_EXISTS()	False if no JSON values are matched
	JSON_QUERY()	Fragments of a JSON document
	JSON_VALUE()	Scalar value
	JSON_TEXTCONTAINS()	Row(s) matching the Full-Text search expr.
	IS JSON	make it enforce strict JSON syntax
GROUP BY	JSON_VALUE()	Scalar value
ORDER BY	JSON_VALUE()	Scalar value

JSON字段索引 (SODA vs SQL)

创建索引

Oracle JSON 中的索引

- B-Tree Index
- Bitmap Index
- Composite B-tree Index
- Search Index

Function-Based Index

在Oracle中使用JSON
创建索引 - SODA Base

Function-Based

```
OracleDatabase db = new
OracleRDBMSClient().getDatabase(Conn);
OracleCollection col = =
db.openCollection(collection);
doc = db.createDocumentFromString(indexSpec);
col.admin().createIndex(doc);
```

```
indexSpec
{ "name"    : "ZIPCODE_IDX",
  "fields" : [ { "path"      :
"address.zip",
                 "datatype" : "number",
                 "order"    : "asc" } ]}
```

Search Index

```
OracleDatabase db = new
OracleRDBMSClient().getDatabase(Conn);
OracleCollection col =
db.openCollection(collection);
col.admin().createJsonSearchIndex(indexSpec);
```

```
indexSpec
{ "name" :
"SEARCH_AND_DATA_GUIDE_IDX" }
```

在Oracle中使用JSON (面向DBA)

创建索引 - SQL Base

Bitmap Index

```
CREATE BITMAP INDEX has_zipcode_idx
       ON j_purchaseorder (
       json_exists(po_document,
'$.ShippingInstructions.Address.zipCode'));

CREATE BITMAP INDEX cost_ctr_idx ON
j_purchaseorder (
json_value(po_document, '$.CostCenter'));
```

Function-Based

```
CREATE UNIQUE INDEX po_num_idx1 ON
j_purchaseorder (json_value(po_document,
'$.PONumber' RETURNING NUMBER ERROR ON ERROR));
```

Composite B-tree

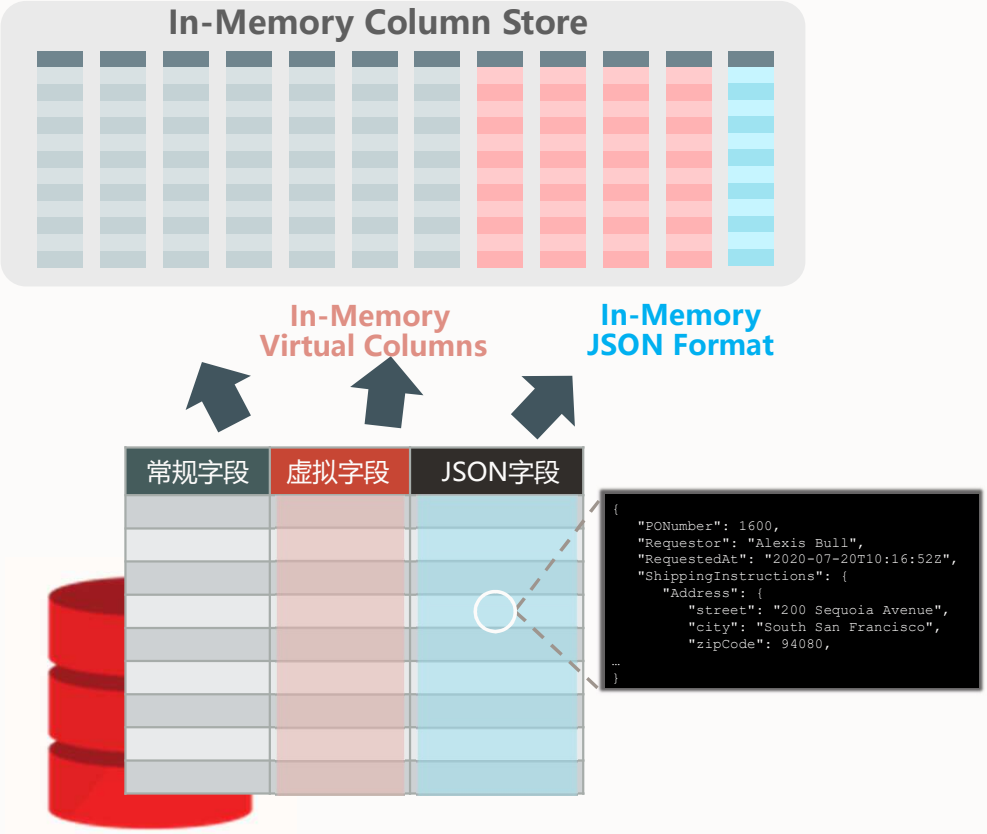
```
CREATE INDEX user_cost_ctr_idx on
j_purchaseorder(userid, costcenter);
```

Search Index

```
CREATE SEARCH INDEX po_search_idx ON
j_purchaseorder (po_document) FOR JSON;
```

JSON 与 In-Memory (面向DBA)

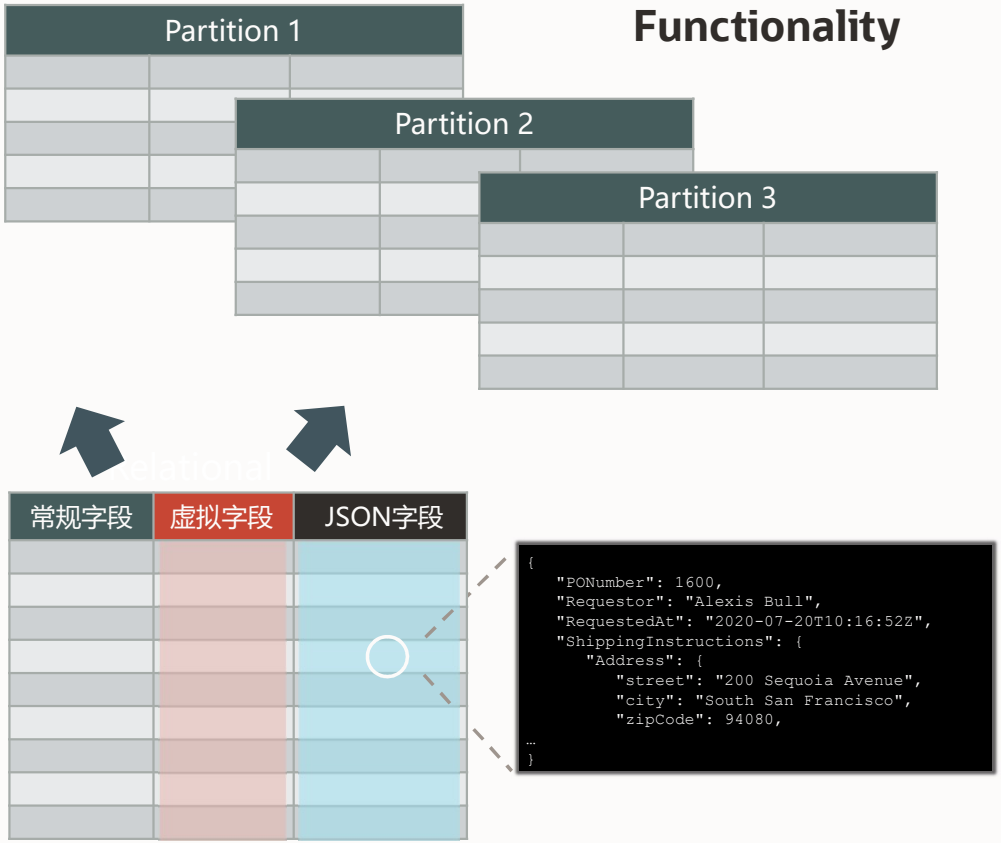
使分析类查询运行的更快



- **可以将完整JSON Document装载到in-memory**
  - 可以利用添加虚拟列的方式将单独的property装载到in-memory
- **在查询JSON内容时，优化器将自动定位到in-memory中的数据**
  - 例如. 满足如下条件的ShippingInstructions.Address.street contains “Avenue”的数据

- **20 - 60x 观察到的性能提升**

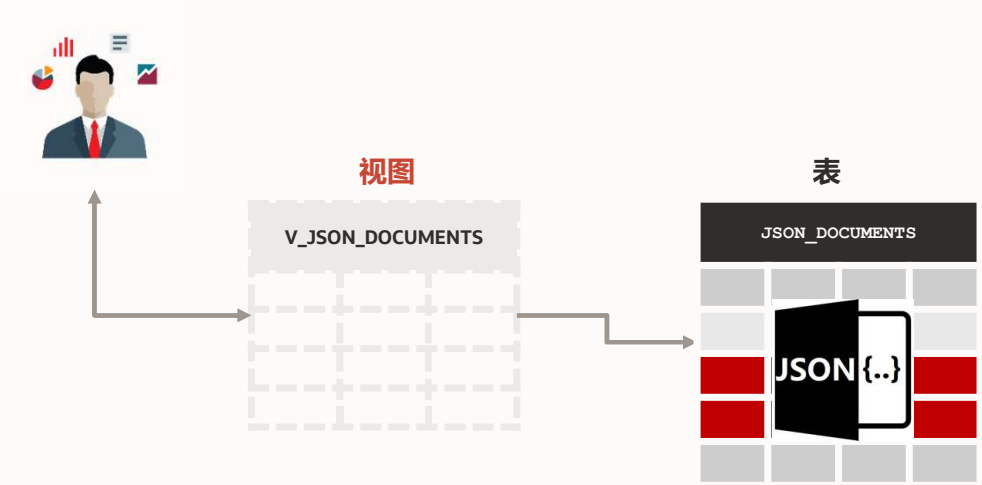
创建分区表 (面向DBA)



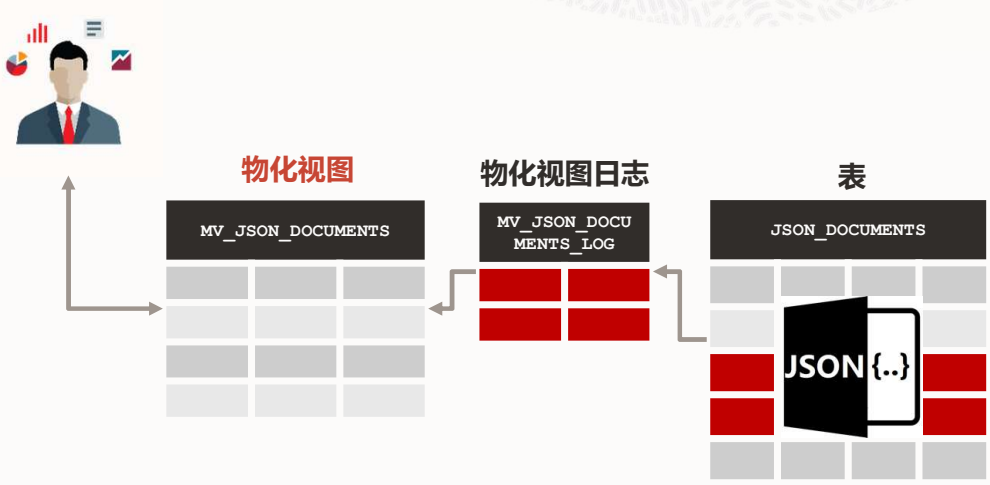

```
CREATE TABLE j_purchaseorder_partitioned
  (id VARCHAR2 (32) NOT NULL PRIMARY KEY,
   date_loaded TIMESTAMP (6) WITH TIME ZONE,
   po_document JSON,
   po_num_vc NUMBER GENERATED ALWAYS AS
    (json_value (po_document, '$.PONumber'
     RETURNING NUMBER))
  )
  PARTITION BY RANGE (po_num_vc)
    (PARTITION p1 VALUES LESS THAN (1000),
     PARTITION p2 VALUES LESS THAN (2000));
```

JSON 数据视图和物化视图 (面向DBA)

创建视图 Vs. 物化视图




```
CREATE VIEW V_JSON_DOCUMENTS
AS
SELECT   ID,
         JSON_VALUE(JSON_DATA,
            '$.ShippingInstructions.name') NAME,
         JSON_VALUE(JSON_DATA,
           '$.ShippingInstructions.Address.city')ADDRESS_CITY,
         JSON_VALUE(JSON_DATA,
            $.ShippingInstructions.Phone[1].number') MOBILE_PHONE
from JSON_DOCUMENTS;
```




```
CREATE MATERIALIZED VIEW LOG ON JSON_DOCUMENTS;
CREATE MATERIALIZED VIEW MV_JSON_DOCUMENTS REFRESH FAST
AS
SELECT   ID,
         JSON_VALUE(JSON_DATA,
            '$.ShippingInstructions.name') NAME,
         JSON_VALUE(JSON_DATA,
           '$.ShippingInstructions.Address.city')ADDRESS_CITY,
         JSON_VALUE(JSON_DATA,
            $.ShippingInstructions.Phone[1].number') MOBILE_PHONE
from JSON_DOCUMENTS;
EXEC DBMS_MVIEW.REFRESH('MV_JSON_DOCUMENTS','FAST');
```

Oracle JSON数据库优势

Oracle JSON DB vs MongoDB

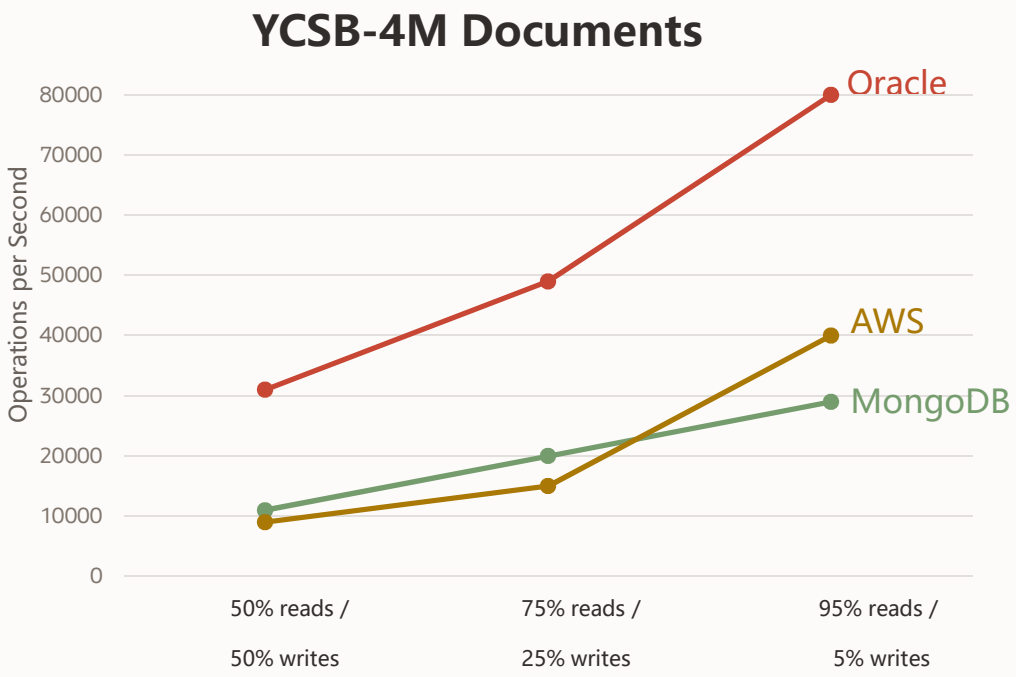
Oracle JSON 数据库与 MongoDB 比较

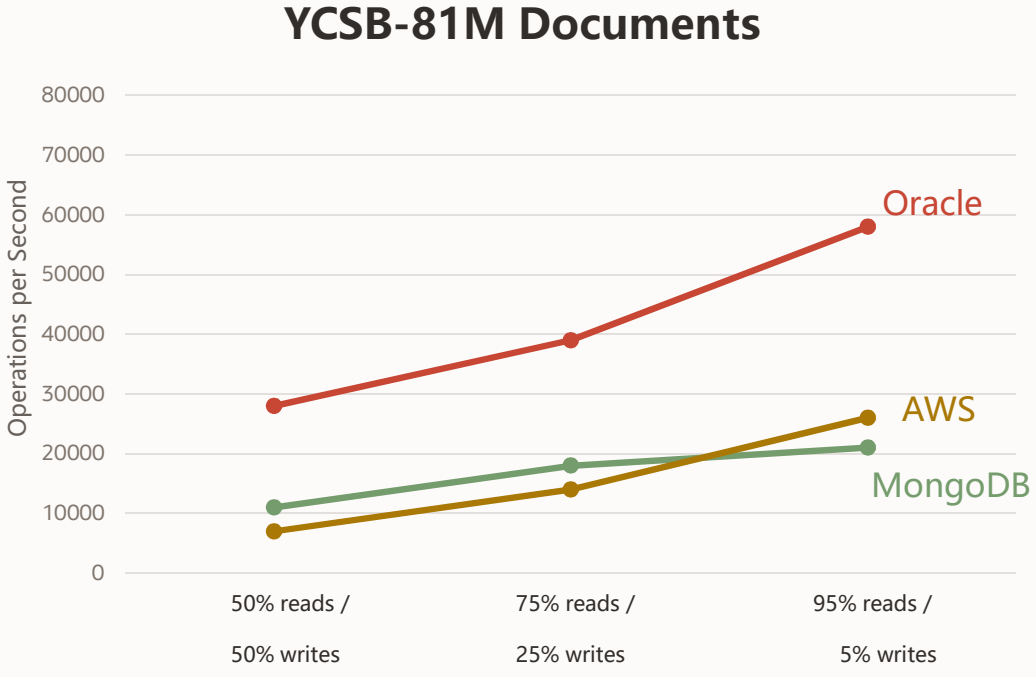
	Oracle JSON 数据库	MongoDB
最大文件尺寸		16 MB
文件的嵌套深度	1024 层	100层
每个集合的索引数量	无限制	64
复合索引字段	无限制(使用JSON SEARCH INDEX)	32
完整文件索引	JSON SEARCH Index	-
服务器端功能	Functions, Procedures, Triggers	不推荐（根据MangoDB官方文档）
多文件查询	始终支持ACID	仅在通过显式API调用请求时才能实现ACID
查询时间	无限制	默认60s
查询规模	无限制	推荐<= 1000个文件
聚合数据大小	无限制	100 MB RAM +显式allowDiskUse参数

Oracle Autonomous JSON Database与 MongoDB 比较

	Oracle Autonomous JSON Database	MongoDB Atlas
Serverless 自动缩放	✓	X
通过SQL访问JSON文档	✓	X
Cross-Collection分析	✓	X
全面security机制	✓	X

Oracle JSON数据库与 MongoDB 比较





Industry-standard Yahoo Cloud Serving Benchmark (YCSB)
Autonomous JSON Database with 8 OCPUs compared to: MongoDB Atlas on M60, AWS DocumentDB on R4.4xlarge
Source: https://www.mongodb.com/atlas-vs-amazon-documentdb/performance as of 8/12/2020

客户案例介绍

证券与保险

客户案例：证券交易所

所在行业: 金融
区域: 美国
行业地位：全球TOP5

客户案例：保险公司

所在行业: 保险
区域: 英国
行业地位：全球TOP5

客户案例 – 跨境电商

需求:

基于客户的购物信息（已经使用Oracle ADW +OAC 进行客户购物偏好等）+Web访问日志，进一步进行客户行为分析，提升客户访问的转化率

方案:

利用Oracle ADW中的JSON DB 能力，把Nginx的日志信息（JSON格式）快速加载到ADW中。分析层利用ADW中加载的日志信息，结合ADW中其他用户和业务数据进行用户访问行为分析，结合分析结果制定优化策略

Oracle方案的优势:

1：集业务处理和分析于单一数据库中，无需移动数据可以满足全部应用需求，从而快速满足业务需求。

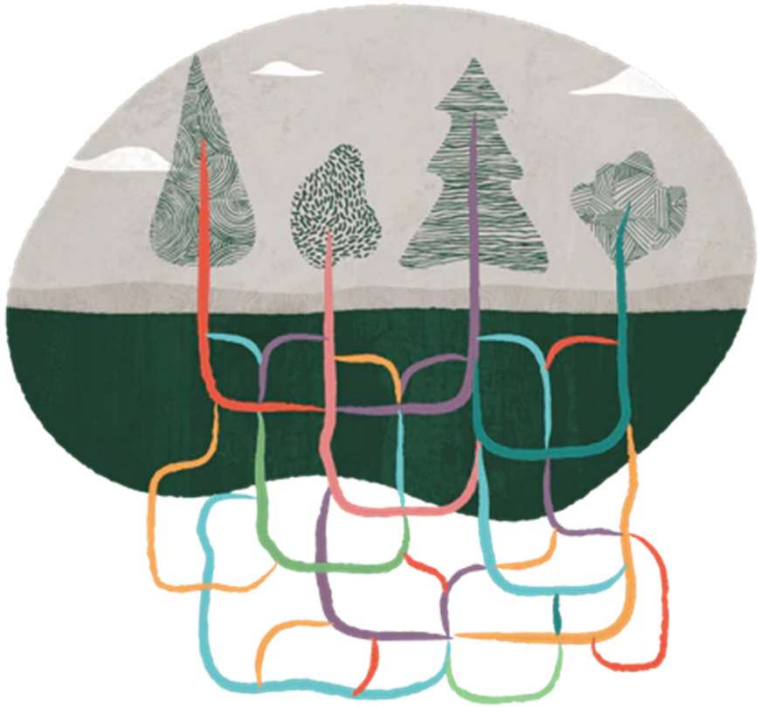
2：原生JSON数据类型支持，丰富的函数简化开发

同时支持基于SODA的API接口和SQL访问，给开发者最大灵活性

架构简单，运维方便，降低运维成本，提高系统稳定性

为什么选择 oracle JSON DB?

技术层面和业务层面



为什么选择 ORACLE – 技术层面

Oracle Database

- ✓ **可操作性**
  - 可以直接使用SQL/PLSQL 进行交互
  - 数据存储
- ✓ **事务一致性保障**
  - 可以跨多个collection和多个documents
  - 不需要在程序端中额外处理
- ✓ **更高级的数据库引擎**
  - 全面支持 SQL/JSON
  - 复杂查询优化
  - 智能处理
  - 任意场景
- ✓ **经过更优化的硬件设施**
  - Exadata Smart Scans
- ✓ **Oracle 平台支持**
  - Oracle 平台对JSON数据的广泛支持
  - 关键数据的管理保障
  - 企业级安全
  - Oracle工具集
  - Oracle Enterprise Manager

为什么选择 ORACLE – 业务层面

Oracle Database

- ✓ **减少额外投资**
  - 一个Oracle DB可以应对多种需求
  - JSON 以外还有 Graph, Spatial 等
- ✓ **无需额外费用**
  - 免费使用如JSON，Graph, Spatial
- ✓ **可快速对应开发**
  - 无需部署其他独立数据库，即刻尝试验证新的业务模型
  - 提供了免费的低代码开发平台APEX
- ✓ **更优秀的性能**
  - 更加紧凑，高效的JSON存储结构（OSON）
  - 基于分区表，In-memory
  - ***基于Exadata的极致性能***（Smart scan 等）
- ✓ **企业级售后服务**

Thank you