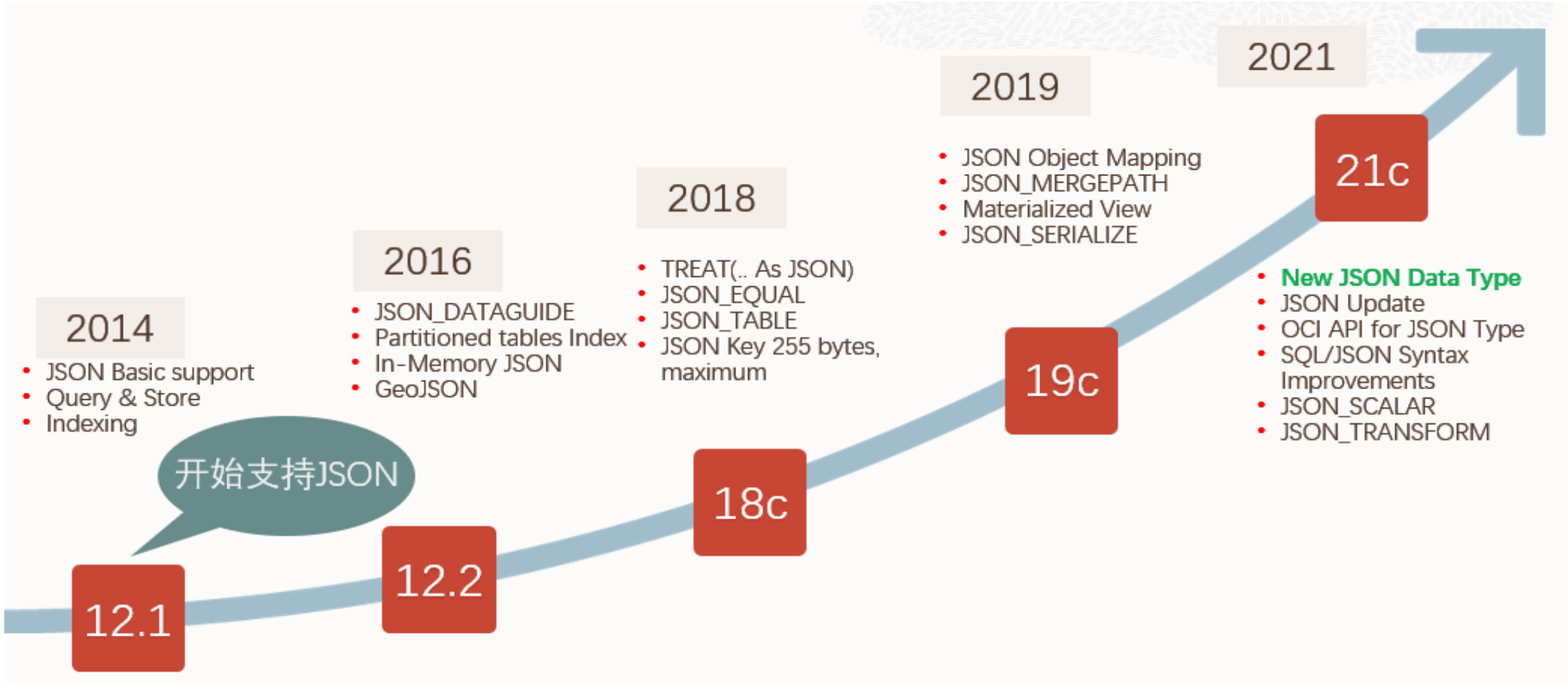
Oracle JSON的发展历程



Oracle 的 JSON 数据存储方式

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（ UCS2/AL16UTF16）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

ADW：19c (is json format **oson** (size limit 32m))

JSON (native json, 21c)

备注：除了JSON之外，其他数据类型都需要使用约束

- 创建表时，设置LOB (COLUMN_NAME) STORE AS (CACHE) 语句以便查询时提高性
- 使用SecureFiles LOBs 的 同时考虑中等压缩比，以达到空间与性能的良好平衡
- 创建JSON字段时，建议增加IS_JSON约束，以确保存储的数据为正确的JSON（**21c之前**）
- 21c前推荐用BLOB, 21c开始用JSON （ native data type ）

迁移到JSON数据类型

1.GoldenGate

2.在线重定义

```
BEGIN
DBMS_REDEFINITION.start_redef_table(
  ...
  col_mapping => 'JSON(text_jcol) json_type_col');
END;
```

Oracle 21c原生JSON数据类型

原生 JSON 二进制类型

- 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（独立格式）**
  - 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

与oracle平台深度集成

并行查询
RAC
Golden Gate
…
Leverage Advanced Security features
VPD, Encryption, ..

Manage JSON together with other data
Operational simplicity from converged database

Oracle 21c原生JSON数据测试

```
SQL> desc tjson
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 VJSON                                              VARCHAR2(4000)
 CJSON                                              CLOB
 BJSON                                              BLOB
 OJSON                                              JSON

SQL> insert into tjson values(1,'{"name":"xiaoming", "age":10}','{"name":"xiaoming", "age":10}','{"name":"xiaoming", "age":10}','{"name":"xiaoming", "age":10}');

1 row created.

SQL> insert into tjson values(2,'{"name":"xiaohong", "age":18}','{"name":"xiaohong", "age":18}','{"name":"xiaohong", "age":18}','{"name":"xiaohong", "age":18}');

1 row created.

SQL> insert into tjson values(3,'{"name":"小张", "age":21}','{"name":"小张", "age":21}','{"name":"小张", "age":21}','{"name":"小张", "age":21}');

1 row created.

SQL> commit;

Commit complete.

SQL> select dbms_rowid.rowid_relative_fno(rowid)file#,dbms_rowid.rowid_block_number(rowid) block# from tjson;

     FILE#     BLOCK#
---------- ----------
        16        867
        16        867
        16        867

SQL> alter system dump datafile 16 block 867;

System altered.

SQL>
```

Oracle 21c原生JSON数据测试

CLOB

```
col  2: [88]
 00 70 00 01 02 0c 80 80 00 02 00 00 00 01 00 00 00 17 13 70 00 44 48 90 00
 3e 00 00 3a 01 00 7b 00 22 00 6e 00 61 00 6d 00 65 00 22 00 3a 00 22 00 78
 00 69 00 61 00 6f 00 68 00 6f 00 6e 00 67 00 22 00 2c 00 20 00 22 00 61 00
 67 00 65 00 22 00 3a 00 31 00 38 00 7d
LOB
Locator:
  Length:       112(88)
  Version:        1
  Byte Length:    2
  LobID: 00.00.00.01.00.00.00.17.13.70
  Flags[ 0x02 0x0c 0x80 0x80 ]:
    Type: CLOB
    Storage: SecureFile
    Characterset Format: IMPLICIT
    Options: VaringWidthReadWrite
  SecureFile Header:
    Length:   68
    Old Flag: 0x48 [ DataInRow SecureFile ]
    Flag 0:   0x90 [ INODE Valid ]
    Layers:
      Lengths Array: INODE:62
      INODE:
        FLG0: 0x00 [ INLINE LENNB: 1 ]
        FLG1: 0x00 [ VERNB: 1 ]
        LEN: 58
        VER: 1
        INLINE DATA (58):
          00 7b 00 22 00 6e 00 61 00 6d 00 65 00 22 00 3a
          00 22 00 78 00 69 00 61 00 6f 00 68 00 6f 00 6e
          00 67 00 22 00 2c 00 20 00 22 00 61 00 67 00 65
          00 22 00 3a 00 31 00 38 00 7d
```

BLOB

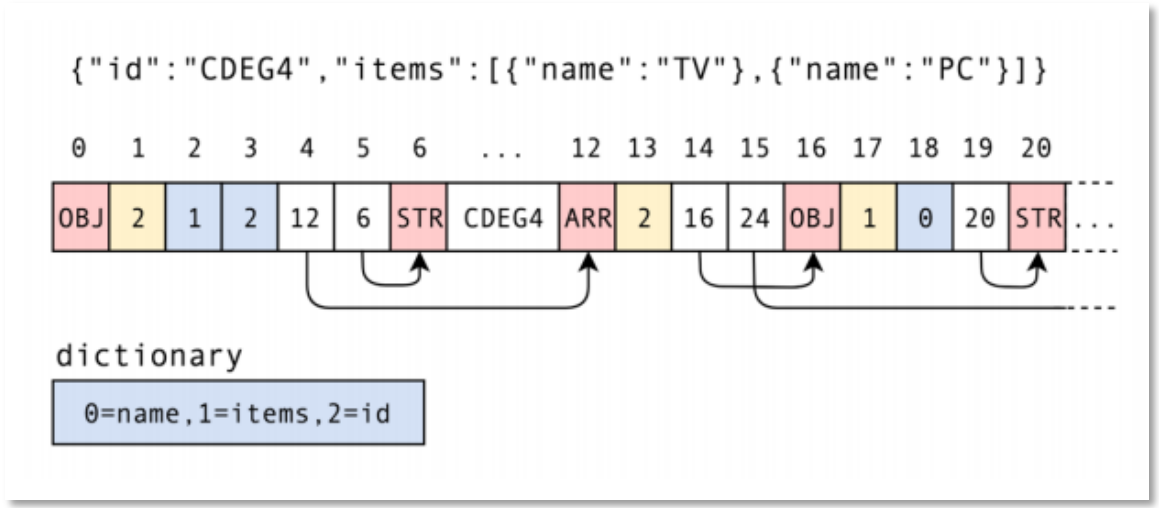
```
col  3: [59]
 00 70 00 01 01 0c 00 80 00 01 00 00 00 01 00 00 00 17 13 71 00 27 48 90 00
 21 00 00 1d 01 7b 22 6e 61 6d 65 22 3a 22 78 69 61 6f 68 6f 6e 67 22 2c 20
 22 61 67 65 22 3a 31 38 7d
LOB
Locator:
  Length:       112(59)
  Version:        1
  Byte Length:    1
  LobID: 00.00.00.01.00.00.00.17.13.71
  Flags[ 0x01 0x0c 0x00 0x80 ]:
    Type: BLOB
    Storage: SecureFile
    Characterset Format: IMPLICIT
    Options: ReadWrite
  SecureFile Header:
    Length:   39
    Old Flag: 0x48 [ DataInRow SecureFile ]
    Flag 0:   0x90 [ INODE Valid ]
    Layers:
      Lengths Array: INODE:33
      INODE:
        FLG0: 0x00 [ INLINE LENNB: 1 ]
        FLG1: 0x00 [ VERNB: 1 ]
        LEN: 29
        VER: 1
        INLINE DATA (29):
          7b 22 6e 61 6d 65 22 3a 22 78 69 61 6f 68 6f 6e
          67 22 2c 20 22 61 67 65 22 3a 31 38 7d
```

JSON

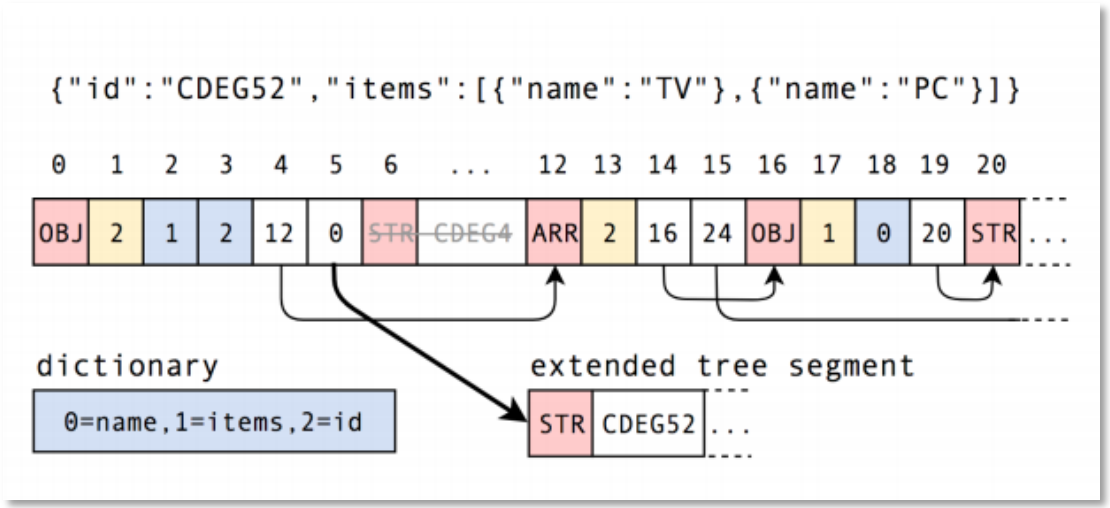
```
col  4: [78]
 00 70 00 01 01 0c 00 80 00 01 00 00 00 01 00 00 00 17 13 72 00 3a 48 90 00
 34 00 00 30 01 ff 4a 5a 01 21 06 02 00 09 00 14 00 00 9c e6 00 05 00 00 04
 6e 61 6d 65 03 61 67 65 84 02 02 01 00 08 00 11 08 78 69 61 6f 68 6f 6e 67
 21 c1 13
LOB
Locator:
  Length:       112(78)
  Version:        1
  Byte Length:    1
  LobID: 00.00.00.01.00.00.00.17.13.72
  Flags[ 0x01 0x0c 0x00 0x80 ]:
    Type: BLOB
    Storage: SecureFile
    Characterset Format: IMPLICIT
    Options: ReadWrite
  SecureFile Header:
    Length:   58
    Old Flag: 0x48 [ DataInRow SecureFile ]
    Flag 0:   0x90 [ INODE Valid ]
    Layers:
      Lengths Array: INODE:52
      INODE:
        FLG0: 0x00 [ INLINE LENNB: 1 ]
        FLG1: 0x00 [ VERNB: 1 ]
        LEN: 48
        VER: 1
        INLINE DATA (48):
          ff 4a 5a 01 21 06 02 00 09 00 14 00 00 9c e6 00
          05 00 00 04 6e 61 6d 65 03 61 67 65 84 02 02 01
          00 08 00 11 08 78 69 61 6f 68 6f 6e 67 21 c1 13
```

Oracle 21c原生数据类型存储结构

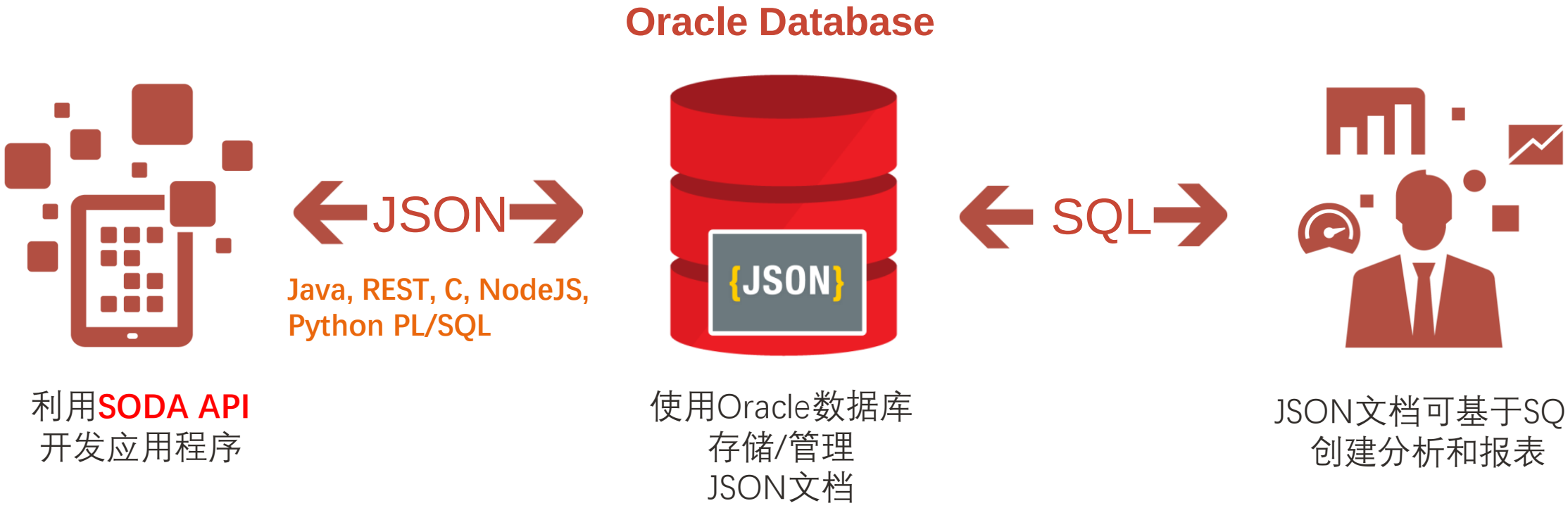
内部存储结构



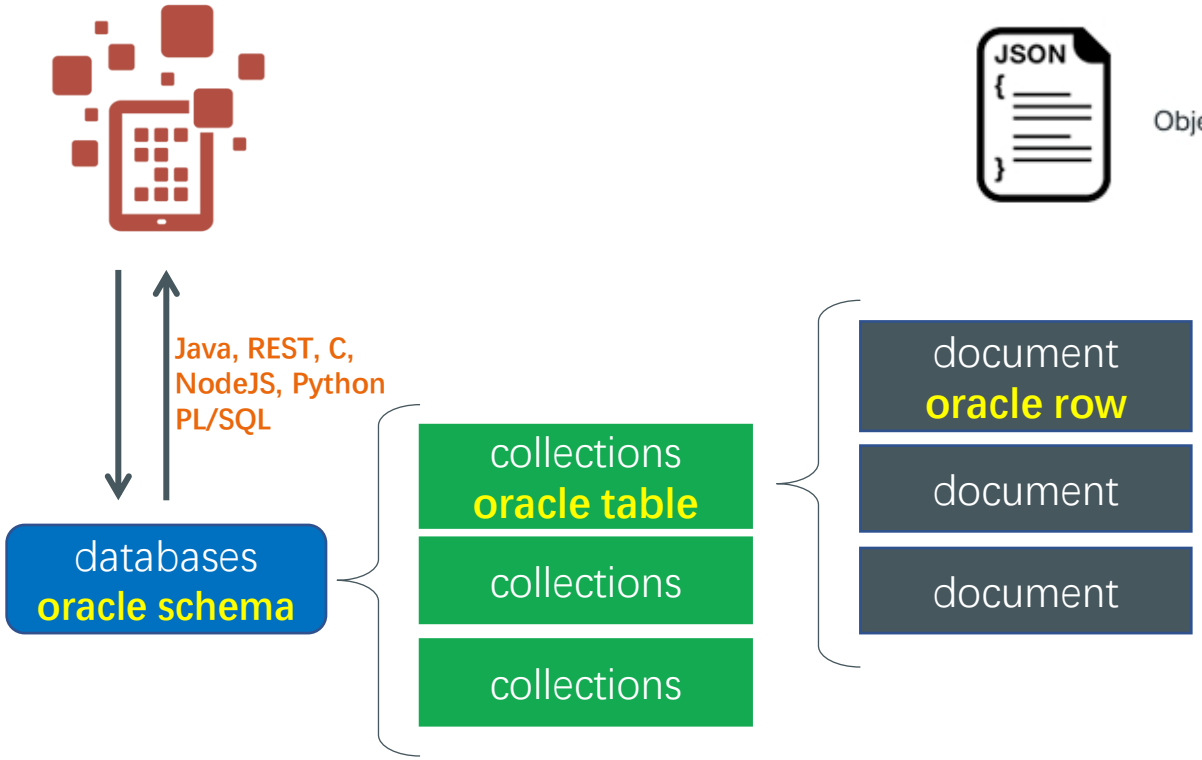
更新时的结果

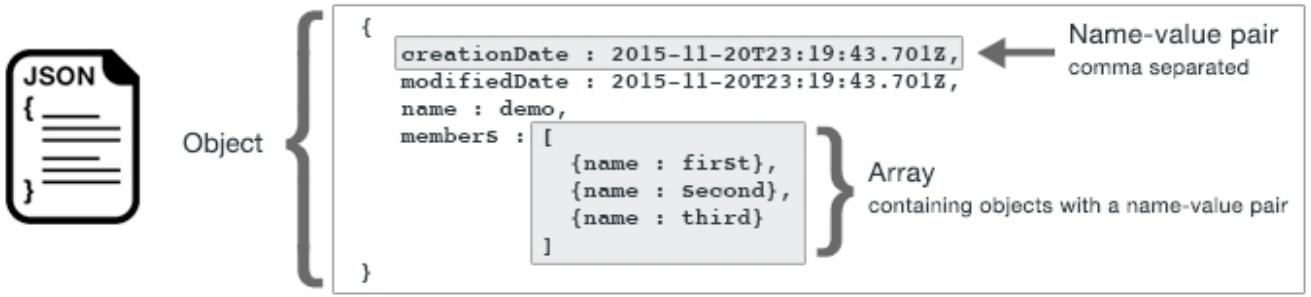


Oracle 的 JSON 数据访问方式



SODA (Simple Oracle Document Access)





A **document** is a JSON value

- Document is identified by a **Key**
  - Key drives CRUD operations: get, update, delete
  - Beyond CRUD: support of filter queries

Store documents inside a **collection**

  - Collection stores similar documents
  - Collection is identified by name
- One or more collections reside in **database**
  - Application connects to database

https://www.oracle.com/database/technologies/appdev/soda.html
grant SODA_APP to ohsdba;

SODA (Simple Oracle Document Access)

简单Oracle文档访问 (Simple Oracle Document Access, SODA) 是一组 NoSQL 样式 API， 让您无需了解结构化查询语言 (Structured Query Language, SQL) 或者这些文档在数据库中的 存储方式，就能在 Oracle 数据库中创建和存储文档的集合（在特定 JSON 中），检索这些文档 以及查询这些文档。

SQLcl

适用Oracle数据库的更易用的SQL命令行开发界面

- 内联编辑，语句完成，命令调用…
- SODA 命令

```
[oracle@xd09mdb03 bin]$ ./sql /nolog


SQLcl: Release 22.3 Production on Thu Oct 13 23:35:46 2022

Copyright (c) 1982, 2022, Oracle.  All rights reserved.

SQL> help
For help on a topic type help <topic>
List of Help topics available:
/               @               @@              ACCEPT          ALIAS
APEX            APPEND          AQ              ARBORI          ARCHIVE_LOG
BREAK           BRIDGE          BTITLE          CD              CHANGE
CLEAR           CLOUDSTORAGE    CODESCAN        COLUMN          COMPUTE
CONNECT         CONNECTIONS     COPY            CS              CTAS
DATAPUMP        DBCCRED         DDL             DEFINE          DEL
DESCRIBE        DG              DISCONNECT      EDIT            EXECUTE
EXIT            FIND            FORMAT          GET             HISTORY
HOST            INFORMATION     INPUT           LIQUIBASE       LIST
LOAD            MIGRATEADVISOR  MKSTORE         MODELER         NET
OCI             OERR            ORAPKI          PASSWORD        PAUSE
```

```
SQL> help soda
SODA Help
------

Simple Oracle Document Access (SODA) is a set of NoSQL-style APIs that let you create
and store collections of documents (in particular JSON) in Oracle Database, retrieve
them, and query them, without needing to know Structured Query Language (SQL) or how
the documents are stored in the database. See:
https://docs.oracle.com/en/database/oracle/simple-oracle-document-access/
            soda create <collection_name>
                      Create a new collection.
                      /* Example: create a collection named fruit */
                      soda create fruit

            soda insert <collection_name> <json_str | filename>
                      Insert a JSON document into a collection.

                      /* Example: insert a JSON document into a collection named fruit */
                      soda insert fruit {"name" : "orange", "quantity" : 20}

            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>
                     /* Example: Get all documents from the collection named fruit. */
                     soda get fruit

                     /* Example: Get the documents in the collection named fruit where the
                        "name" attribute is equal to  "orange" */
                     soda get fruit -f {"name" : "orange"}

                     /* Example: Get the documents in the collection named fruit where the
                        "name" attribute is equal to  "orange" */
                     soda get fruit -f {"quantity" : {"$lt" : 50}}
```

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();
```

Python

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

Java

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

PL/SQL (and Oracle Application Express)

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

```
create table json_documents1 (
  id    raw(16) not null,
  data  blob,
  constraint json_documents_pk primary key (id),
  constraint json_documents_json_chk check   (data is json)
);
```

```
SQL> create table json_documents1 (
  2    id    raw(16) not null,
  3    data  blob,
  4    constraint json_documents_pk primary key (id),
  5    constraint json_documents_json_chk check   (data is json)
  6* );

Table JSON_DOCUMENTS1 created.
```

```
OracleClient client = new OracleRDBMSClient();
db = client.getDatabase(jdbcConn);
OracleCollection col = db.admin.createCollection("json_documents2");
```

```
SQL> soda create json_documents2

Successfully created collection: json_documents2
```

```
SQL> desc json_documents1

Name    Null?       Type
_______ ___________ _____________
ID      NOT NULL    RAW(16 BYTE)
DATA                BLOB
```

```
SQL> desc json_documents2

Name              Null?       Type
________________  __________  _____________________
ID                NOT NULL    VARCHAR2(255)
CREATED_ON        NOT NULL    TIMESTAMP(6)
LAST_MODIFIED     NOT NULL    TIMESTAMP(6)
VERSION           NOT NULL    VARCHAR2(255)
JSON_DOCUMENT                 JSON
```

JSON使用可定制的Collection Metadata

```
{
   "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

```
SQL> desc json_documents2

Name                 Null?       Type
-------------------- ----------- --------------------
ID                   NOT NULL    VARCHAR2(255)
CREATED_ON           NOT NULL    TIMESTAMP(6)
LAST_MODIFIED        NOT NULL    TIMESTAMP(6)
VERSION              NOT NULL    VARCHAR2(255)
JSON_DOCUMENT                    JSON
SQL>
```

通过SQL访问JSON数据

21c JSON Developer's Guide

1.SQL/JSON query functions:
    1.JSON_VALUE
    2.JSON_QUERY
    3.JSON_TABLE
2.SQL conditions:
    1.JSON_TEXTCONTAINS
    2.JSON_EQUAL
3.SQL/JSON conditions:
    1.JSON_EXISTS
    2.IS JSON
    3.IS NOT JSON
4.Simple dot notation acts similar to a combination of query functions JSON_VALUE and JSON_QUERY.
    1.A dot-notation syntax, is essentially a table alias, followed by a JSON column name, followed by one or more field names — all separated by periods(.).
5.SQL/JSON generation functions:
    1.JSON_OBJECT
    2.JSON_ARRAY
    3.JSON_OBJECTAGG
    4.JSON_ARRAYAGG
6.SQL functions:
    1.JSON_SERIALIZE
    2.JSON_SCALAR [21c]
7.SQL aggregate function:
    1.JSON_DATAGUIDE
8.JSON Update functions:
    1.JSON_TRANSFORM [21c]
    2.JSON_MERGEPATCH

```
select po.po_document.ponumber from j_purchaseorder po;
```

JSON基于SQL和SODA创建索引

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;
```

Search Index

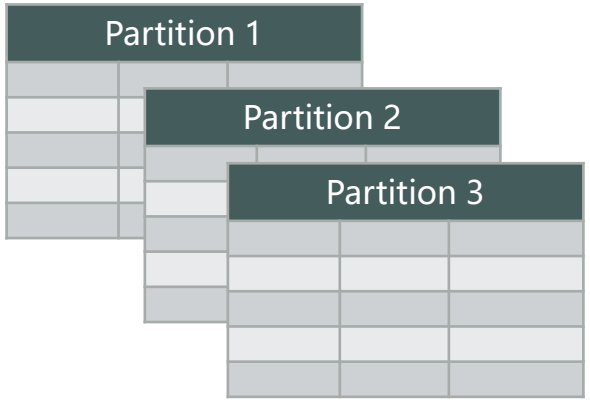
```
OracleDatabase db = new
OracleRDBMSClient().getDatabase(Conn);
OracleCollection col = =
db.openCollection(collection);
doc = db.createDocumentFromString(indexSpec);
col.admin().createIndex(doc);

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

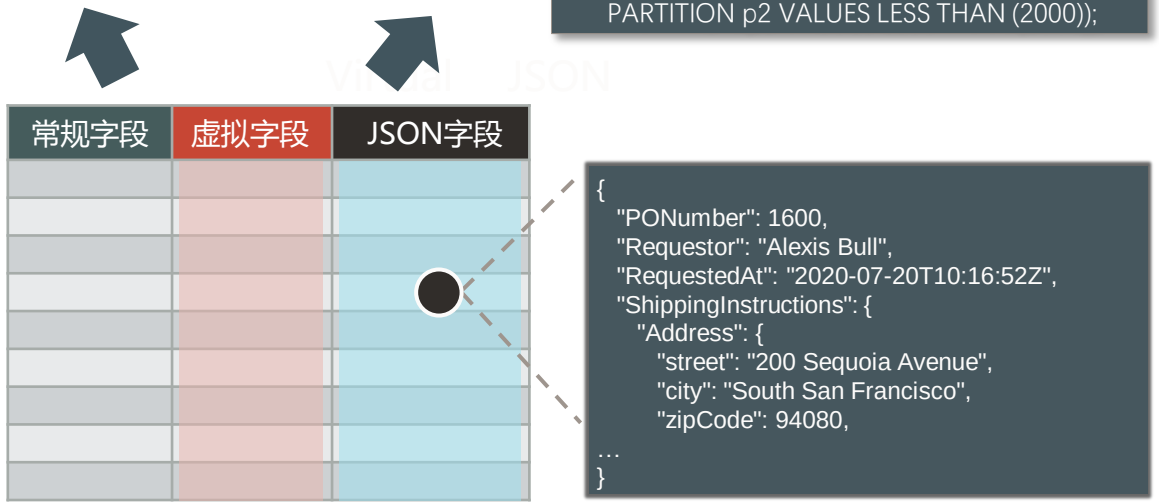
Function-Based

```
OracleDatabase db = new
OracleRDBMSClient().getDatabase(Conn);
OracleCollection col =
db.openCollection(collection);
col.admin().createJsonSearchIndex(indexSpec);
indexSpec
{ "name" : "SEARCH_AND_DATA_GUIDE_IDX" }
```

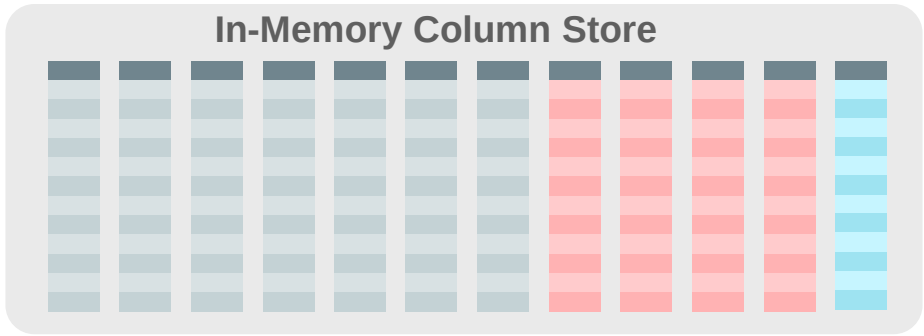
JSON和分区表及In-Memory

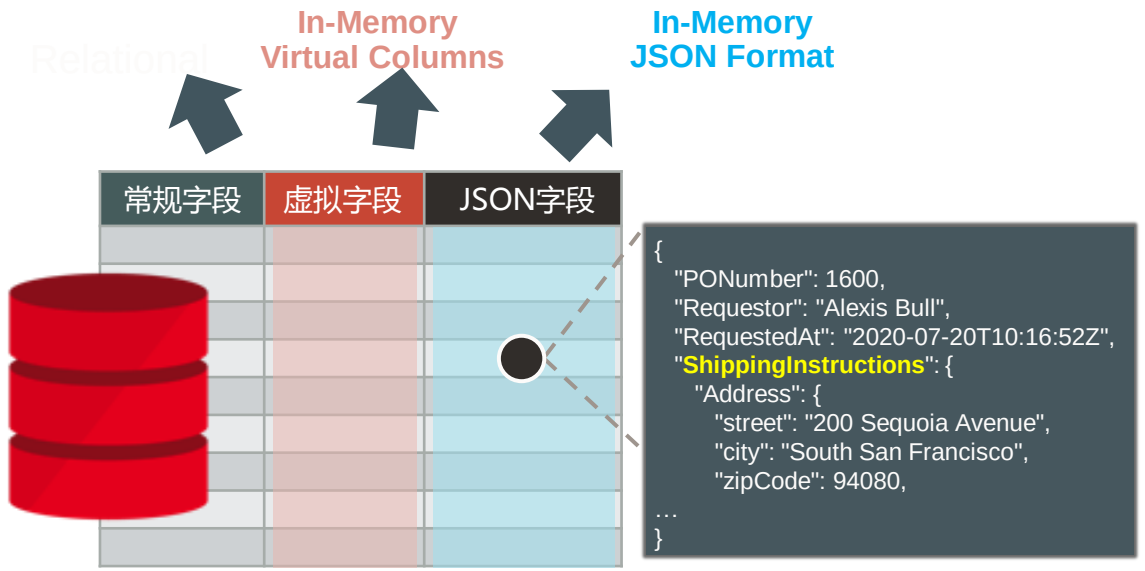



```
CREATE TABLE j_purchaseorder_partitioned
  (id VARCHAR2 (32) NOT NULL PRIMARY KEY,
   date_loaded TIMESTAMP (6) WITH TIME ZONE,
   po_document JSON,
   po_num_vc NUMBER GENERATED ALWAYS AS
    (json_value (po_document, '$.PONumber'
     RETURNING NUMBER))
  )
  PARTITION BY RANGE (po_num_vc)
   (PARTITION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




```
{
  "PONumber": 1600,
  "Requestor": "Alexis Bull",
  "RequestedAt": "2020-07-20T10:16:52Z",
  "ShippingInstructions": {
    "Address": {
      "street": "200 Sequoia Avenue",
      "city": "South San Francisco",
      "zipCode": 94080,
…
}
```





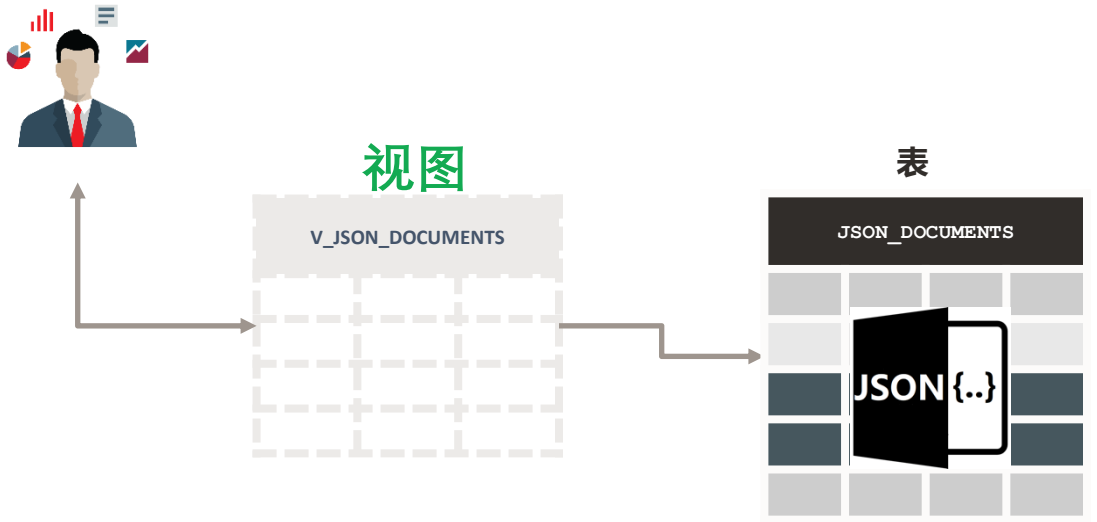
可以将完整JSON Document装载到in-memory

- 可以利用添加虚拟列的方式将单独的property装载到in-memory

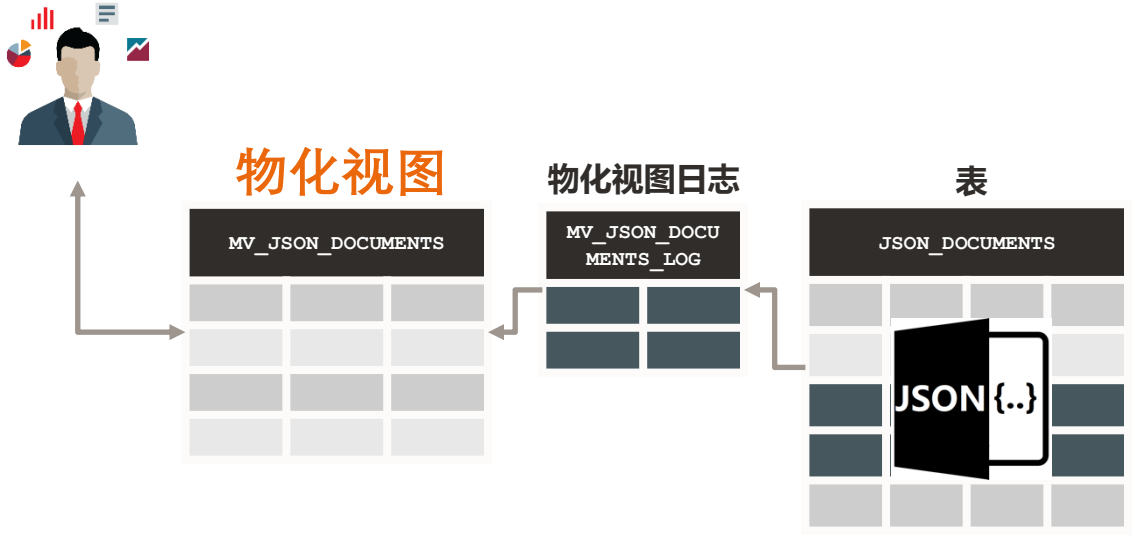
在查询JSON内容时，优化器将自动定位到in-memory中的数据

- 例如. 满足如下条件的ShippingInstructions.Address.street contains “Avenue”的数据

JSON和物化视图




```
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和数据库

可以直接使用SQL/PLSQL 进行交互

- 从关系型表获取JSON数据
- 从JSON中导出平坦数据
- PL/SQL可直接操作JSON数据
- 支持SQL/NoSQL风格

数据存储

- 基于BLOB，可以存储超大型数据

灵活的数据模型

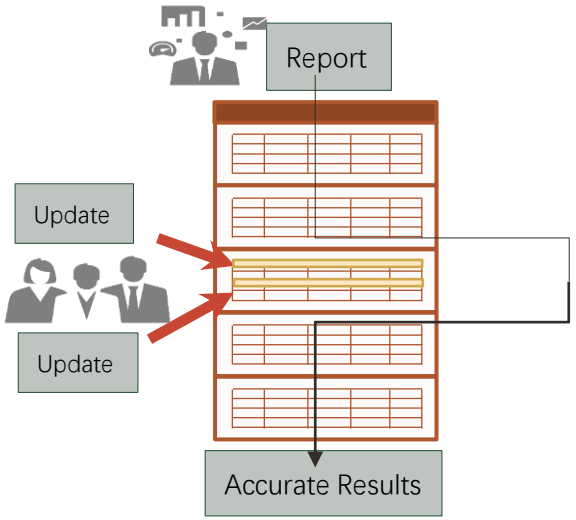
- 一对多
- 多对多
- 完备的JOIN支持

默认支持事务的一致性

- 可以跨多个collection和多个documents
- 不需要在程序端中额外处理

默认保证读一致性

- 多版本读取一致性模型可确保读取查询和更新查询同时运行，无阻塞且结果一致



Oracle 平台对JSON数据的广泛支持

- Sharding
- RAC
- Golden Gate Replication
- Flashback

关键数据的管理保障

- Active Data Guard
- ZDLRA

企业级安全

- Transparent Data Encryption
- VPD
- Data Masking

Oracle工具集

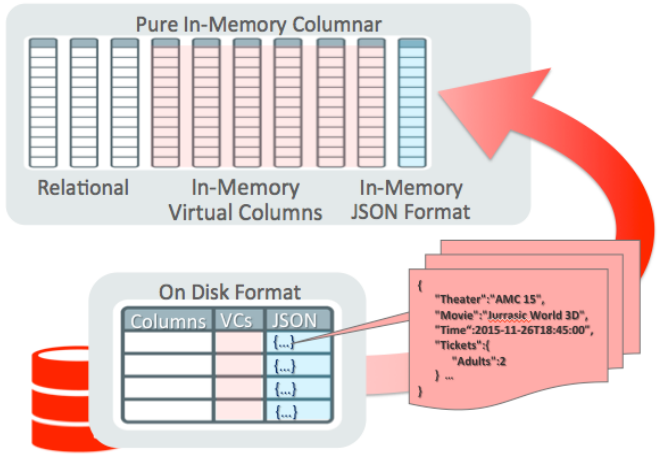
- Datapump
- SQL*Loader
- Apex
- SQLDeveloper

Oracle Enterprise Manager

Oracle JSON和数据库

In Memory Columnar Store for JSON

- 利用虚拟列将JSON documents的内容加载到 In-Memory
- 查询是优化器将自动的直接查询以加载到in-memory 中的数据



Exadata Smart Scans

- Exadata Smart Scans 在存储单元上执行部分SQL查询
- JSON查询操作“下推”到Exadata存储单元

  Massively parallel processing of JSON documents

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

Oracle JSON和MongoDB驱动

select ords.installed_version from dual;

```
SQL> select ords.installed_version from dual;

INSTALLED_VERSION
------------------------------------------------------------------------
22.3.3.r3111929
```

[oracle@xd09mdb04 ~]$ ords config set mongo.enabled true

```
[oracle@xd09mdb04 ~]$ ords config set mongo.enabled true
Picked up _JAVA_OPTIONS: -Xms1200M -Xmx1200M

ORDS: Release 22.3 Production on Wed Nov 16 08:39:30 2022

Copyright (c) 2010, 2022, Oracle.

Configuration:
  /u01/ords-standalone/config/

The global setting named: mongo.enabled was set to: true
[oracle@xd09mdb04 ~]$ ords config list
Picked up _JAVA_OPTIONS: -Xms1200M -Xmx1200M

ORDS: Release 22.3 Production on Wed Nov 16 08:39:42 2022

Copyright (c) 2010, 2022, Oracle.

Configuration:
  /u01/ords-standalone/config/

Database pool: default

Setting                             Value                                          Source
----------------------------------  ---------------------------------------------  -----------
database.api.enabled                true                                           Global
db.connectionType                   basic                                          Pool
db.hostname                         xd09mdb04                                      Pool
db.password                         ******                                         Pool Wallet
db.port                             1521                                           Pool
db.servicename                      orcl                                           Pool
db.username                         ORDS_PUBLIC_USER                               Pool
feature.sdw                         true                                           Pool
mongo.enabled                       true                                           Global
restEnabledSql.active               true                                           Pool
security.requestValidationFunction  ords_util.authorize_plsql_gateway              Pool
standalone.context.path             /ords                                          Global
standalone.doc.root                 /u01/ords-standalone/config/global/doc_root    Global
standalone.http.port                8888                                           Global

[oracle@xd09mdb04 ~]$
```

[oracle@xd09mdb04 ~]$ ords serve &

```
[oracle@xd09mdb04 ~]$ ords serve &
[1] 18411
[oracle@xd09mdb04 ~]$ Picked up _JAVA_OPTIONS: -Xms1200M -Xmx1200M

[oracle@xd09mdb04 ~]$
ORDS: Release 22.3 Production on Wed Nov 16 08:48:43 2022

Copyright (c) 2010, 2022, Oracle.

Configuration:
  /u01/ords-standalone/config/

2022-11-16T08:48:45.458Z INFO        HTTP and HTTP/2 cleartext listening on host: 0.0.0.0 port: 8888
2022-11-16T08:48:45.497Z INFO        Disabling document root because the specified folder does not exist: /u01/ords-standalone/config/global/doc_root
2022-11-16T08:48:45.989Z INFO        Oracle API for MongoDB listening on port: 27017
2022-11-16T08:48:45.992Z INFO        The Oracle API for MongoDB connection string is:
        mongodb://[{user}:{password}@]localhost:27017/{user}?authMechanism=PLAIN&authSource=$external&ssl=true&retryWrites=false&loadBalanced=true
2022-11-16T08:48:52.756Z INFO        Configuration properties for: |default|lo|
```

sqlplus ohsdba/*******@xd09mdb04/orcl

exec ords.enable_schema(true);

```
[oracle@xd09mdb04 ~]$ sqlplus ohsdba/Oracle12#@xd09mdb04/orcl

SQL*Plus: Release 21.0.0.0.0 - Production on Wed Nov 16 16:41:46 2022
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.

Last Successful login time: Mon Oct 17 2022 17:25:40 +08:00

Connected to:
Oracle Database 21c Enterprise Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> exec ords.enable_schema(true);

PL/SQL procedure successfully completed.
```

Oracle JSON和MongoDB驱动

可以通过MongoDB 驱动连接到ORACLE数据库

- 开发人员可以沿用已掌握的技能， MongoDB 相关工具, drivers 驱动等
- 从MongoDB 移植到Oracle将变得简单
  - 减少代码的修改量（除了 aggregation pipeline）
- 支持 SQL 操作 JSON collections
  - 更加高效的分析能力, 基于机器学习
  - 可以与其他的不同数据模型进行关联
  - 将数据以JSON形式返回

Oracle Database API for MongoDB 支持与 MongoDB 兼容的客户端和驱动程序直接连接到Oracle数据库。
使用以下其中一项作为 MongoDB 连接字符串 URL。
mongodb://[user:password@]host:27017/[user]?authMechanism=PLAIN&authSource=$external&ssl=true&retryWrites=false&loadBalanced=true

```
[oracle@xd09mdb04 ~]$ mongosh --tlsAllowInvalidCertificates 'mongodb://ohsdba:Oracle12%23@192.168.10.99:27017/ohsdba?authMechanism=PLAIN&authSource=$external&ssl=tru
e&retryWrites=false&loadBalanced=true'
Current Mongosh Log ID: 6374833857dd05bd12365082
Connecting to:          mongodb://<credentials>@192.168.10.99:27017/ohsdba?authMechanism=PLAIN&authSource=%24external&ssl=true&retryWrites=false&loadBalanced=true&tl
sAllowInvalidCertificates=true&appName=mongosh+1.6.0
Using MongoDB:          4.2.14
Using Mongosh:          1.6.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

ohsdba>

ohsdba> show databases;
ohsdba  517.66 KiB
admin          0 B
ohsdba> show collections;
employees
mustories
ohsdba>
```

Oracle 23c JSON Schema和JSON Relational Duality

CREATE TABLE jsonschema (jcol JSON **VALIDATE '{"type" : "object"}'**);

```
Connected to:
Oracle Database 21c Enterprise Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> CREATE TABLE jsonschema (jcol JSON VALIDATE '{"type" : "object"}');
CREATE TABLE jsonschema (jcol JSON VALIDATE '{"type" : "object"}')
                                            *
ERROR at line 1:
ORA-00907: missing right parenthesis


SQL>
```

```
Connected to:
Oracle Database 23c Enterprise Edition Release 23.0.0.0.0 - Beta
Version 23.1.0.0.0

SQL> CREATE TABLE jsonschema (jcol JSON VALIDATE '{"type" : "object"}');

Table created.

SQL>
SQL> desc jsonschema
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 JCOL                                               JSON
```

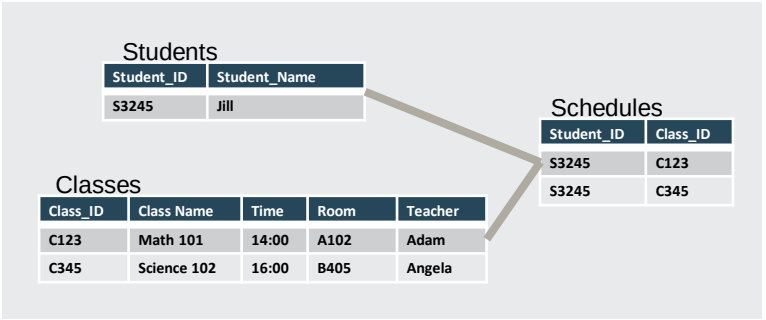
JSON Model of Students

```
{
  "student"    : "Jill",    "studentid": S3245,
  "schedule": [
   {
     "class"   : "Math 101",
                          "classID": "C123",
     "time"    : "14:00",
     "room"    : "A102",
     "teacher" : "Adam",      "teachId": "T543"
   },
   {
     "class"   : "Science 102","classID": "C345",
     "time"    : "16:00",
     "room"    : "B405",
     "teacher" : "Angela",    "teachId": "T789"
   }]
}
```

JSON Access Methods (HTTP)

PUT, POST, GET, DELETE

Relational Model of Students



Relational Access Method (SQL)

```
Select *
From Students
natural join schedule
natural join Classes
```

```
SQL> CREATE OR REPLACE JSON RELATIONAL DUALITY VIEW team_dv AS
  2    SELECT
  3    JSON {'teamId' : t.team_id,
  4          'name'   : t.name,
  5          'points' : t.points,
  6          'driver' :[
  7          SELECT
  8          JSON {'driverId' : d.driver_id,
  9                'name'     : d.name,
 10                'points'   : d.points WITH NOCHECK}
 11          FROM driver d WITH INSERT UPDATE
 12          WHERE d.team_id = t.team_id]}
 13    FROM team t WITH INSERT UPDATE DELETE;

View created.
```

将数据存储为关系格式的行，通过文档API进行访问，开发人员可以使用 Mongo API、SQL JSON 或两者来构建应用程序

Oracle JSON Demo

```
[demo@xd09mdb03 JdbcExamples]$ mvn -q exec:java -Dexec.args='jdbc:oracle:thin:ohsdba/Oracle12#@192.168.10.205/orclpdb1' -Dexec.mainClass="emp.RunAll"

Running CreateTable
-----------------------
Created table emp

Running Insert
-----------------------
Inserted three employees into the emp table

Running JSONP
-----------------------
Inserted employee Clark
Retrieved Smith from the database
{"name":"Smith","job":"Programmer","salary":40000,"created":"2022-10-16T02:40:02.773000Z"}
"2022-10-16T02:40:02.773000Z" is of type STRING
"2022-10-16T02:40:02.773000Z" is of type TIMESTAMPTZ
true

Running JSONB
-----------------------
Emp object inserted successfully!
Emp object retrieved from database.
King, king@oracle.com
  MOBILE 555-333-2222
  WORK 555-333-1111

Running Jackson
-----------------------
Jack inserted into successfully
```

```
[demo@xd09mdb03 SodaExamples]$ mvn -q exec:java -Dexec.args='jdbc:oracle:thin:ohsdba/Oracle12#@192.168.10.205/orclpdb1' -Dexec.mainClass="emp.RunAll"

Running CreateCollection
-----------------------
Created collection

Running Insert
-----------------------
Inserted three employees into the employee collection

Running JSONP
-----------------------
Inserted employee Clark
Retrieved Smith from the database
{"name":"Smith","job":"Programmer","salary":40000,"created":"2022-10-16T02:22:57.793000Z"}
"2022-10-16T02:22:57.793000Z" is of type STRING
"2022-10-16T02:22:57.793000Z" is of type TIMESTAMPTZ
true

Running JSONB
-----------------------
Employee object inserted successfully!
Employee object retrieved from database.
King, king@oracle.com

Running Jackson
-----------------------
Jack inserted into successfully
```

Jdk :17+
Database : 21.3
yum install maven git -y
git clone https://github.com/oracle/json-in-db/

```
SQL> desc emp
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DATA                                               JSON

SQL> desc employees
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL VARCHAR2(255)
 CREATED_ON                                NOT NULL TIMESTAMP(6)
 LAST_MODIFIED                             NOT NULL TIMESTAMP(6)
 VERSION                                   NOT NULL VARCHAR2(255)
 JSON_DOCUMENT                                      JSON
```

mvn package
mvn -q exec:java -Dexec.args='jdbc:oracle:thin:ohsdba/Oracle12#@192.168.10.205/orclpdb1' -Dexec.mainClass="emp.RunAll"

Blockchain
JSON
Tables

JSON Data Guide

Oracle Database 23c

In-Memory
Document Analytics

Full SQL
Support

New In 23c

2X Faster Operational
Workloads than
MongoDB

New In 23c

JSON Schema

MongoDB API

New In 23c

New In 23c

JSON Relational Duality

Native JSON Datatype

Best in Class Security

New In 23c

JS Stored
Procedures

New In 23c

Graph Analytics

New In 23c

10X faster
Analytics