ORACLE

Oracle JSON Database for Developer

一场美丽的邂逅 : JSON

张弘弢

资深解决方案工程师

2021年12月17日

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。

初识

相知

契合

初识

相知

契合

初识JSON

- 诞生于2001年数据交换格式
  - 数据类型，stands for : JavaScript Object Nomination
  - 基于JavaScript，用于序列化对象，数组，数字，字符串，布尔值和空值的语法
  - 将JSON转换为JavaScript对象非常容易，反之亦然
    - 数据由名称/值对组成，其中值可以是对象，数组，数字，字符串等
    - 对象由一对用逗号分隔的名称/值对组成，并放在花括号内
    - 数组是方括号内的对象的逗号分隔列表，对象类型可以不同
- 和XML的关系
  - JSON是骨感版的XML，风头已经盖过XML
  - 更少的数据量，序列化优势明显
  - 更好的可读性和兼容性

XML

```
<empinfo>
   <employees>
      <employee>
         <name>James Kirk</name>
         <age>40</age>
      </employee>
      <employee>
         <name>Jean-Luc Picard</name>
         <age>45</age>
      </employee>
      <employee>
         <name>Wesley Crusher</name>
         <age>27</age>
      </employee>
   </employees>
</empinfo>
```

JSON

```
{  "empinfo" :
    {
        "employees" :  [
        {
            "name" : "James Kirk",
            "age" : 40,
        },
        {
            "name" : "Jean-Luc Picard",
            "age" : 45,
        },
        {
            "name" : "Wesley Crusher",
            "age" : 27,
        }
                    ]
    }
}
```

JSON特点

从XML汲取到的优势特点：

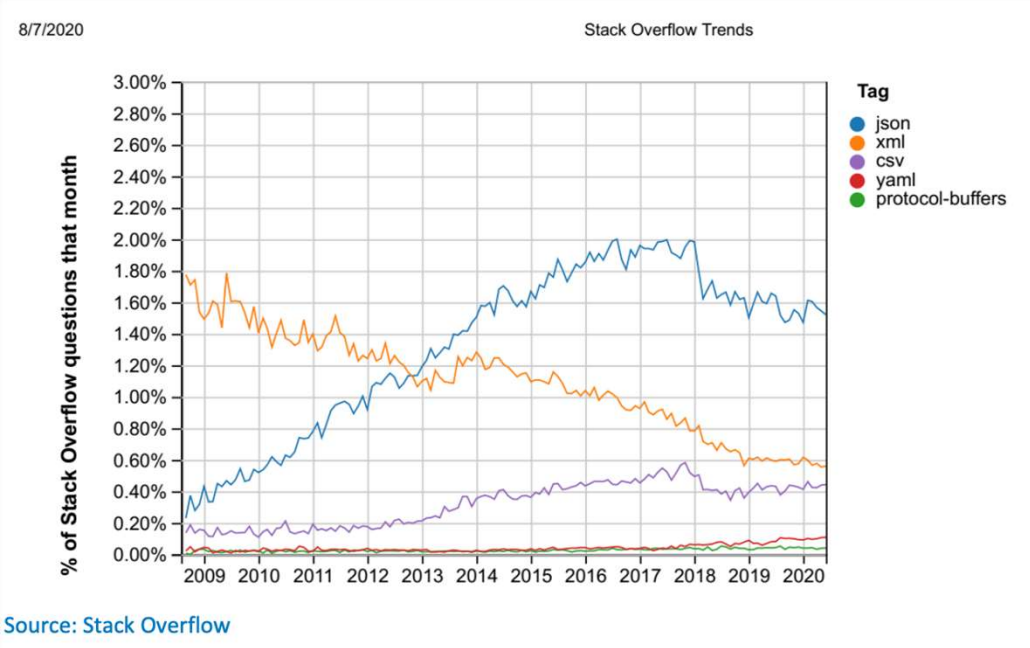
- 可读性好，结构清晰
- 分层存储(层次嵌套)
- 都可作为Ajax传输数据
- 都跨平台，可作为数据传输格式

相比XML额外的优势特点：

- 数据格式简单，易读易写，且数据都是压缩的，文件较小，便于传输
- JSON解析难度较低，而xml需要循环遍历DOM进行解析，效率较低
- 服务端和客户端可以直接使用JSON，便于维护。而不同客户端解析xml可能使用不同方法
- JSON 已成为当前服务器与 web 应用之间数据传输的公认标准

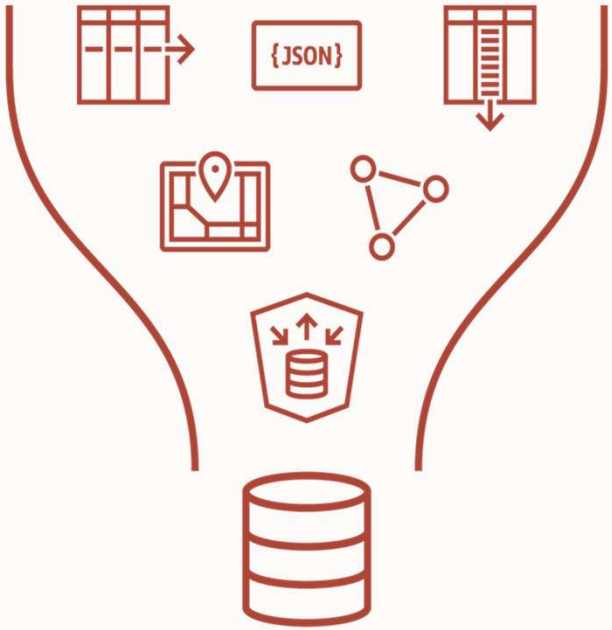
最受欢迎的数据交换格式

- 三个方面因素
  - JavaScript的普及
  - AJAX大行其道
  - Web 2.0的诞生
- 几乎所有的公共REST接口的支持
  - Facebook API, Google Geocoder, Twitter API , OCI API



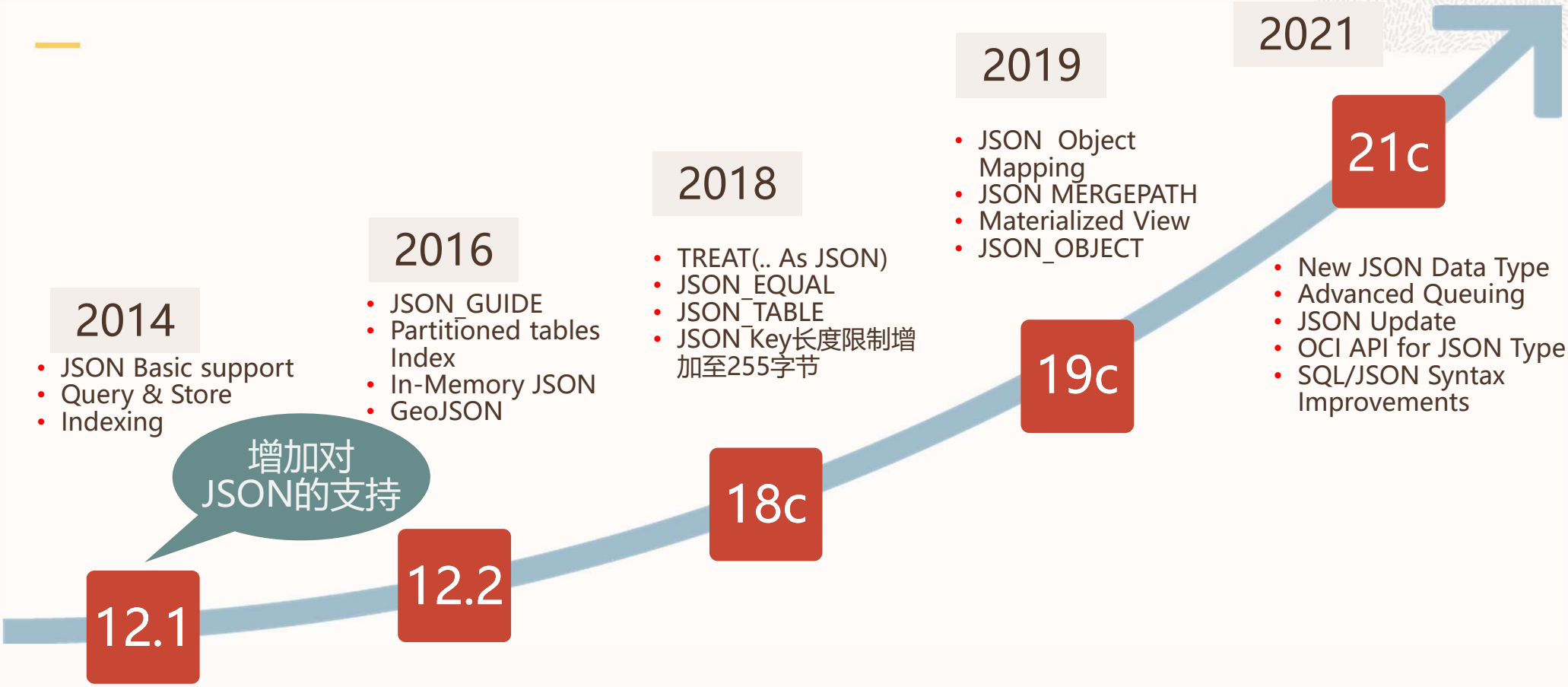
成熟的融合数据库

融合为一的多模型，多负载的数据平台



跨越模型的洞察帮助您轻松发现数据价值

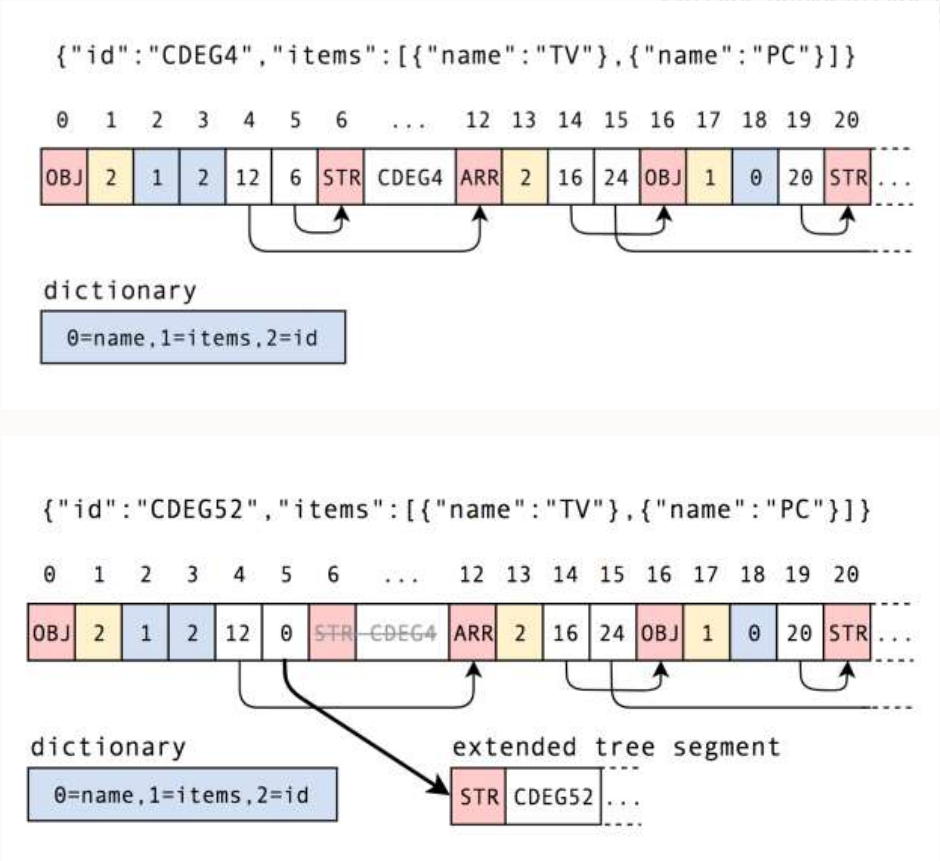
日趋完美的支持


2014
JSON Basic support
Query & Store
Indexing
增加对JSON的支持
12.1
2016
JSON_GUIDE
Partitioned tables Index
In-Memory JSON
GeoJSON
12.2
2018
TREAT(.. As JSON)
JSON_EQUAL
JSON_TABLE
JSON Key长度限制增加至255字节
18c
2019
JSON Object Mapping
JSON MERGEPATH
Materialized View
JSON_OBJECT
19c
2021
21c
New JSON Data Type
Advanced Queuing
JSON Update
OCI API for JSON Type
SQL/JSON Syntax Improvements

原生的数据类型(21c)

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
  - 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



体系的支持才是最好的保障

- ✓ SQL:2016 开始包含JSON
- ✓ Oracle最新支持的标准是SQL:2016
- ✓ MAA架构支持
- ✓ MSA架构支持
- ✓ 云平台支持和自制数据库AJD

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

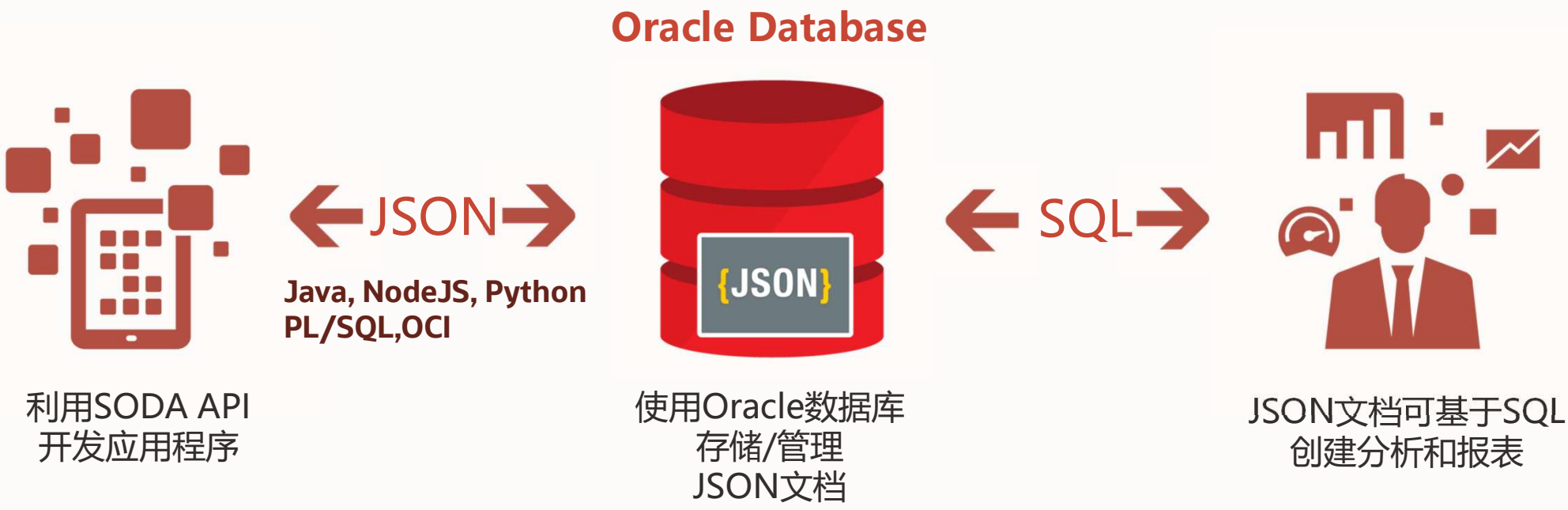
- 在本地安装的 Oracle 数据库
- Exadata
- 云上的DBaaS

初识

相知

契合

两种JSON数据访问方法



现有版本

- 数据库版本11.2.0.4
- 业务数据表
- 数据规模
- 字段类型
  - 常规类型
  - CLOB

```
SQL> info AIRPORTDELAYS
TABLE: AIRPORTDELAYS
        LAST ANALYZED:2021-08-26 09:55:49.0
        ROWS         :4408
        SAMPLE SIZE  :4408
        INMEMORY     :DISABLED
        COMMENTS     :

Columns
NAME          DATA TYPE             NULL  DEFAULT    COMMENTS
*ID           NUMBER                No
 AIRPORTCODE  VARCHAR2(4000 BYTE)   Yes
 NAME         VARCHAR2(4000 BYTE)   Yes
 TIME         CLOB                  Yes
 STATISTICS   CLOB                  Yes

Indexes
               INDEX_NAME    UNIQUENESS    STATUS    FUNCIDX_STATUS    COLUMNS
_________________________ _____________ _________ _________________ __________
KEVIN.AIRPORTDELAYS_PK    UNIQUE        VALID                       ID


SQL>
```

数据样例

工作表 查询构建器

```
1  SELECT
2      id,
3      airportcode,
4      name,
5      time,
6      statistics
7  FROM
8      airportdelays;
```

查询结果 ×

SQL | 已提取 50 行, 用时 0.47 秒

	ID	AI...	NAME	TIME	STATISTICS
1	10	FLL	Fort Lauder...	{"Label":"2003/06","Month":6...	{"# of Delays":{"Carrier":247,"Late Aircraft":256,"National Aviation Sys...
2	11	IAD	Washington,...	{"Label":"2003/06","Month":6...	{"# of Delays":{"Carrier":320,"Late Aircraft":295,"National Aviation Sys...
3	12	IAH	Houston, TX...	{"Label":"2003/06","Month":6...	{"# of Delays":{"Carrier":329,"Late Aircraft":730,"National Aviation Sys...
4	13	JFK	New York, N...	{"Label":"2003/06","Month":6...	{"# of Delays":{"Carrier":376,"Late Aircraft":226,"National Aviation Sys...
5	14	LAS	Las Vegas, ...	{"Label":"2003/06","Month":6...	{"# of Delays":{"Carrier":511,"Late Aircraft":678,"National Aviation Sys...
6	15	LAX	Los Angeles...	{"Label":"2003/06","Month":6...	{"# of Delays":{"Carrier":830,"Late Aircraft":765,"National Aviation Sys...
7	1	ATL	Atlanta, GA...	{"Label":"2003/06","Month":6...	{"# of Delays":{"Carrier":1009,"Late Aircraft":1275,"National Aviation S...
8	2	BOS	Boston, MA:...	{"Label":"2003/06","Month":6...	{"# of Delays":{"Carrier":374,"Late Aircraft":495,"National Aviation Sys...
9	3	BWI	Baltimore, ...	{"Label":"2003/06","Month":6...	{"# of Delays":{"Carrier":296,"Late Aircraft":477,"National Aviation Sys...
10	4	CLT	Charlotte, ...	{"Label":"2003/06","Month":6...	{"# of Delays":{"Carrier":300,"Late Aircraft":472,"National Aviation Sys...


```
"id": 1,
"AirportCode": "ATL",
"Name": "Atlanta, GA: Hartsfield-Jackson Atlanta International",
"Time": {
    "Label": "2003/06",
    "Month": 6,
    "Month Name": "June",
    "Year": 2003
},
"Statistics": {
    "# of Delays": {
        "Carrier": 1009,
        "Late Aircraft": 1275,
        "National Aviation System": 3217,
        "Security": 17,
        "Weather Codes":["SNW","RAIN","SUN","CLDY"],
        "Weather": 328
    },
    "Carriers": {
        "Aircraft Types": [{"make" : "Boeing", "models" : ["717","737","757","767","777","787"]},
                           {"make" : "Airbus", "models" : ["A320","A321","A330","A340","A350","A380"]}],
        "Names": "American Airlines Inc.,JetBlue Airways,Continental Air Lines Inc.,Delta Air Lines Inc.,
        "Total": 11
    },
    "Flights": {
        "Cancelled": 216,
        "Delayed": 5843,
        "Diverted": 27,
        "On Time": 23974,
        "Total": 30060
    },
    "Minutes Delayed": {
        "Carrier": 61606,
        "Late Aircraft": 68335,
        "National Aviation System": 118831,
```

JSON数据的查询和处理

11g

```
SQL> select BANNER from v$version;

                                                                        BANNER
--------------------------------------------------------------------------------
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
```

```
SQL> select a.statistics
  2  from airportdelays a
  3  where a.airportcode = 'SFO'
  4* and a.time like '%June%2010%';

                                                                    STATISTICS
--------------------------------------------------------------------------------
{"# of Delays":{"Carrier":593,"Late Aircraft":1048,"National Aviation System":16

SQL>
```

19c

```
SQL> select BANNER from v$version;

                                                                        BANNER
--------------------------------------------------------------------------------
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
```

```
SQL> select a.statistics
  2    from airportdelays a
  3   where a.airportcode = 'SFO'
  4     and a.time."Month Name" = 'June'
  5*    and a.time.Year = '2010';

                                                                    STATISTICS
--------------------------------------------------------------------------------
{"# of Delays":{"Carrier":593,"Late Aircraft":1048,"National Aviation System":16

SQL>
```

JSON数据的查询和处理 – SQL函数

- 多种类型支持
- IS JSON
- 丰富的函数
- Dot Notation

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

处理JSON的5个常用函数

- JSON_VALUE
  - 路径表达式
  - 错误处理

```
COLUMN FirstName FORMAT A15
COLUMN LastName FORMAT A15

SELECT JSON_VALUE(a.data, '$.FirstName') AS first_name,
       JSON_VALUE(a.data, '$.LastName') AS last_name
FROM   json_documents a
ORDER BY 1, 2;

FIRST_NAME      LAST_NAME
--------------- ---------------
Jayne           Doe
John            Doe

2 rows selected.

SQL>
```

- NULL ON ERROR
- ERROR ON ERRO
- DEFAULT on ERROR

- *ORA-40462: JSON_VALUE evaluated to no value*
- *ORA-40597: JSON path expression syntax error*

处理JSON的5个常用函数

- JSON_QUERY
  - 一个或多个JSON片段
  - VARCHAR2/CLOB/BLOB
  - JSON_TABLE的特例

```
SELECT a.data.FirstName,
       a.data.LastName,
       JSON_QUERY(a.data, '$.ContactDetails' WITH WRAPPER) AS contact_details
FROM   json_documents a
ORDER BY a.data.FirstName,
         a.data.Last_name;

FIRSTNAME       LASTNAME        CONTACT_DETAILS
--------------- --------------- ----------------------------------------
Jayne           Doe             [{"Email":"jayne.doe@example.com","Phone
                                ":""}]

John            Doe             [{"Email":"john.doe@example.com","Phone"
                                :"44 123 123456","Twitter":"@johndoe"}]


2 rows selected.

SQL>
```

处理JSON的5个常用函数

- JSON_TABLE
  - 更多功能
  - 更好性能
  - JSON_VALUE/JSON_QUERY/ JSON_EXISTS
  - 略显复杂

```
SELECT jt.first_name,
       jt.last_name,
       jt.contact_details
FROM   json_documents,
       JSON_TABLE(data, '$'
         COLUMNS (first_name    VARCHAR2(50 CHAR) PATH '$.FirstName',
                  last_name     VARCHAR2(50 CHAR) PATH '$.LastName',
                  contact_details  VARCHAR2(4000 CHAR)
                    FORMAT JSON WITH WRAPPER PATH '$.ContactDetails')) jt;

FIRST_NAME      LAST_NAME       CONTACT_DETAILS
--------------- --------------- ----------------------------------------
John            Doe             [{"Email":"john.doe@example.com","Phone"
                                :"44 123 123456","Twitter":"@johndoe"}]

Jayne           Doe             [{"Email":"jayne.doe@example.com","Phone
                                ":""}]


2 rows selected.

SQL>
```

处理JSON的5个常用函数

- JSON_MERGEPATCH
  - 查询和更新
  - 部分更新
  - 支持PRETTY

```
SELECT data FROM json_documents;

DATA
--------------------------------------------------------------------------------
{"id":1,"first_name":"Iron","last_name":"Man"}
{"id":2,"first_name":"Wonder","last_name":"Woman"}
{"id":3,"first_name":"The","last_name":"Hulk"}

SQL>
```

```
SELECT JSON_MERGEPATCH(data, '{"last_name":"banana"}') AS data
FROM   json_documents;

DATA
--------------------------------------------------------------------------------
{"id":1,"first_name":"Iron","last_name":"banana"}
{"id":2,"first_name":"Wonder","last_name":"banana"}
{"id":3,"first_name":"The","last_name":"banana"}

SQL>
```

```
UPDATE json_documents a
SET    a.data = JSON_MERGEPATCH(a.data, '{"last_name":"banana","new_element":"surprise"}')
WHERE  a.data.first_name = 'Iron';
```

```
DATA
--------------------------------------------------------------------------------
{"id":1,"first_name":"Iron","last_name":"banana","new_element":"surprise"}
```

处理JSON的5个常用函数

- JSON_TRANSFORM
  - 21c推出，ADB可用
  - 更新更容易
  - 多种更新操作

```
select json_transform(json_data,
                      set '$.quantity' = 20
                      returning clob pretty) as data
from   t1
where  id = 1;

DATA
--------------------------------------------------------------------------------
{
  "fruit" : "apple",
  "quantity" : 20
}

SQL>
```

```
select json_transform(json_data,
                      insert '$.updated_date' = systimestamp
                      returning clob pretty) as data
from   t1
where  id = 1;

DATA
--------------------------------------------------------------------
{
  "fruit" : "apple",
  "quantity" : 10,
  "updated_date" : "2021-01-05T08:44:58.406618Z"
}

SQL>
```

SODA简介

SODA APIs & SQLcl

- NoSQL风格 APIs
  - Java, JavaScript/Node.js, Python, REST, PL/SQL, C...
- 用于管理JSON数据
  - 创建collection
  - 在collection中存储文档
  - 获取文档
  - 查询文档
  - ***不需要了解NoSQL！***
  - ***非关系型模型***

```
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>
```

SODA对象和表的关系

row

- A **document** is a JSON value
- Document is identified by a **Key**
  - Key drives CRUD operations: get, update, delete
  - Beyond CRUD: support of filter qu
- 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

table

schema

SODA的多语言支持

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;
```

SODA的REST支持

SODA via REST

- Create
- Insert
- Query
- Update

*需要ORDS支持

```
KevinsdeMBP:JSON-Home kevzhang_cn$ curl -u "KEVIN:$MYPWD" -i -X POST \
> "$mysoda_url/soda/latest/$mycollection_name?action=insert" \
> -H "Content-Type: application/json" \
> -d @airportDelays.json
HTTP/1.1 100 Continue

HTTP/1.1 200 OK
Date: Thu, 26 Aug 2021 12:45:11 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Cache-Control: private,must-revalidate,max-age=0
Content-Length: 877253
```

```
SQL> soda count airportdelayscollection

 已选择 4,408 行。

SQL>
```

```
KevinsdeMBP:JSON-Home kevzhang_cn$ curl -X POST -u "KEVIN:$MYPWD" "$mysoda_url/soda/latest/$mycollection_name?action=query"\
>  -H "Content-Type: application/json" \
>    --data-binary \
> '{"AirportCode": "SFO","Time.Month Name": "June","Time.Year": 2010}'
{"items":[{"id":"9AB71A9DC6E9403FBFD464CA209ACFAD","etag":"330A7212B09DE8833D60BF9C3E910D58CE76188FB812744B3E2D8FE48AE7CD09","lastMod
ted":"2021-08-26T12:45:11.540749000Z","links":[{"rel":"self","href":"http://152.67.223.97:38888/ords/kevin/soda/latest/airportdelaysc
,"value":{"id":2463,"AirportCode":"SFO","Name":"San Francisco, CA: San Francisco International","Time":{"Label":"2010/06","Month":6,"
{"# of Delays":{"Carrier":593,"Late Aircraft":1048,"National Aviation System":1648,"Security":1,"Weather Codes":["SNW","RAIN","SUN","
pes":[{"make":"Boeing","models":["717","737","757","767","777","787"]},{"make":"Airbus","models":["A320","A321","A330","A340","A350",
ska Airlines Inc.,JetBlue Airways,Continental Air Lines Inc.,Delta Air Lines Inc.,Frontier Airlines Inc.,AirTran Airways Corporation,
United Air Lines Inc.,US Airways Inc.,Southwest Airlines Co.,Mesa Airlines Inc.","Total":13},"Flights":{"Cancelled":228,"Delayed":337
,"Minutes Delayed":{"Carrier":37339,"Late Aircraft":76800,"National Aviation System":113992,"Security":42,"Total":234902,"Weather Cod
9}}}}],"hasMore":false,"count":1}KevinsdeMBP:JSON-Home kevzhang_cn$
```

```
KevinsdeMBP:JSON-Home kevzhang_cn$ curl -u "KEVIN:$MYPWD" -i -X PUT \
> $mysoda_url/soda/latest/$mycollection_name
HTTP/1.1 201 Created
Date: Thu, 26 Aug 2021 12:43:26 GMT
X-Frame-Options: SAMEORIGIN
Cache-Control: private,must-revalidate,max-age=0
Location: http://152.67.223.97:38888/ords/kevin/soda/latest/airportdelayscollection/
Content-Length: 0

KevinsdeMBP:JSON-Home kevzhang_cn$
```

```
SQL> soda list
集合的列表：

        airportdelayscollection

SQL>
```

```
KevinsdeMBP:JSON-Home kevzhang_cn$ curl -X POST -u "KEVIN:$MYPWD" "$mysoda_url/soda/latest/$mycollection_name?action=update" \
>  -H "Content-Type: application/json" \
>    --data-binary \
> '{
>     "$query": {"id": 1},
>     "$patch": [
>         {
>             "op": "test",
>             "path": "/Statistics/# of Delays/Carrier",
>             "value": 1009
>         },
>         {
>             "op": "replace",
>             "path": "/Statistics/# of Delays/Carrier",
>             "value": 1019
>         }
>     ]
> }'
{"count":1,"itemsUpdated":1}KevinsdeMBP:JSON-Home kevzhang_cn$
KevinsdeMBP:JSON-Home kevzhang_cn$
```

SODA和SQL的对比

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)
);
```

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)

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	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)

初识

相知

契合

JSON数据类型

21c中的JSON类型

- Collection 的默认类型
- 表字段类型

```
SQL> info airportdelayscollection
TABLE: AIRPORTDELAYSCOLLECTION
        LAST ANALYZED:
        ROWS          :
        SAMPLE SIZE   :
        INMEMORY      :DISABLED
        COMMENTS      :

Columns
NAME            DATA TYPE           NULL  DEFAULT    COMMENTS
*ID             VARCHAR2(255 BYTE)  No
 CREATED_ON     TIMESTAMP(6)        No    sys_extract_utc(SYSTIMESTAMP)
 LAST_MODIFIED  TIMESTAMP(6)        No    sys_extract_utc(SYSTIMESTAMP)
 VERSION        VARCHAR2(255 BYTE)  No
 JSON_DOCUMENT  JSON                Yes

Indexes
          INDEX_NAME     UNIQUENESS     STATUS     FUNCIDX_STATUS     COLUMNS
____________________ ____________ _________ _________________ __________
KEVIN.SYS_C008460    UNIQUE         VALID                       ID
```

```
SQL>
  2  CREATE TABLE AIRPORTDELAYS
  3      (
  4       ID          NUMBER ,
  5       AIRPORTCODE VARCHAR2(4000) ,
  6       NAME        VARCHAR2(4000) ,
  7       TIME        JSON ,
  8       STATISTICS  JSON
  9*     ) ;

Table AIRPORTDELAYS 已创建。

SQL>
```

JSON数据索引

19c中的JSON类型索引

- 单值索引

```
SQL> create index AIRPORTDELAYS_INDX01
  2*   on AIRPORTDELAYS t (t.Time.Year.number(),t.Time.Month.number());

Index AIRPORTDELAYS_INDX01 已创建。

SQL>
```

```
SQL> explain plan
  2  for select count(*) from AIRPORTDELAYS t
  3* where t.Time.Year.number() = :1;

已解释。

                                                                                PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------
Plan hash value: 679434008

-----------------------------------------------------------------------------------------
| Id  | Operation         | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                     |     1 |    13 |     2   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE   |                     |     1 |    13 |            |          |
|*  2 |   INDEX RANGE SCAN| AIRPORTDELAYS_INDX01 |    33 |   429 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access(JSON_VALUE("TIME" /*+ LOB_BY_VALUE */  FORMAT OSON ,
              '$.Year.number()' RETURNING NUMBER NULL ON ERROR)=TO_NUMBER(:1))

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

已选择 19 行。

SQL>
```

JSON数据索引

21c中的JSON类型索引

- 多值索引

```
SQL> create multivalue index AIRPORTDELAYS_INDX02
  2* on AIRPORTDELAYS t (t.Statistics.Carriers."Aircraft Types".models.string());

Multivalue INDEX 已创建。

SQL>
```

```
SQL> explain plan
  2  for select count(*) from AIRPORTDELAYS t
  3* where t.Statistics.Carriers."Aircraft Types".models.string() = :3;

                                                                                                                    PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------------------------------------------------------------
Plan hash value: 1188854045

-----------------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                     |     1 |  6104 |    45   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE                     |                     |     1 |  6104 |            |          |
|*  2 |   TABLE ACCESS BY INDEX ROWID BATCHED| AIRPORTDELAYS       |     1 |  6104 |    45   (0)| 00:00:01 |
|   3 |    HASH UNIQUE                      |                     |     1 |  6104 |            |          |
|*  4 |     INDEX RANGE SCAN (MULTI VALUE)  | AIRPORTDELAYS_INDX02 |    13 |       |     9   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(JSON_VALUE("T"."STATISTICS" /*+ LOB_BY_VALUE */  FORMAT OSON , '$.Carriers."Aircraft
              Types".models.string()' RETURNING VARCHAR2(4000) NULL ON ERROR)=:3)
   4 - access(JSON_QUERY("STATISTICS" /*+ LOB_BY_VALUE */  FORMAT OSON , '$.Carriers."Aircraft
              Types".models.string()' RETURNING VARCHAR2(4000) ASIS  WITHOUT ARRAY WRAPPER ERROR ON ERROR NULL ON
              EMPTY NULL ON MISMATCH MULTIVALUE)=:3)

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)

已选择 24 行。
```

JSON数据性能

JSON类型的性能提升

- 数据行总量1M
- 11g vs. 21c
- CLOB vs. JSON
- 7.963 vs. 0.460

```
SQL> info airportdelays;
TABLE: AIRPORTDELAYS
        LAST ANALYZED:2021-08-27 06:02:00.0
        ROWS          :1128448
        SAMPLE SIZE   :1128448
        INMEMORY      :DISABLED
        COMMENTS      :
```

```
SQL> select count(*) from airportdelays;

  COUNT(*)
----------
   1128448
```

11g

```
SQL> select count(*)
  2  from airportdelays a
  3* where a.time like '%"Month":6%"Year":2010%';

  COUNT(*)
----------
      7424
```

```
Plan hash value: 2584671063

-----------------------------------------------------
| Id  | Operation          | Name          | E-Rows |
-----------------------------------------------------
|   0 | SELECT STATEMENT   |               |        |
|   1 |  SORT AGGREGATE    |               |      1 |
|*  2 |   TABLE ACCESS FULL| AIRPORTDELAYS |    220 |
-----------------------------------------------------
```

```
Statistics
----------------------------------------------------------
            738  CPU used by this session
            738  CPU used when call started
            744  DB time
              3  Requests to/from client
              1  enqueue releases
              1  enqueue requests
          23481  non-idle wait count
             58  non-idle wait time
              2  opened cursors cumulative
              1  opened cursors current
          23478  physical read total IO requests
          23250  physical read total multi block requests
              1  pinned cursors current
              1  recursive calls
         374721  session logical reads
             58  user I/O wait time
              4  user calls
用时：  00:00:07.963
SQL>
```

21c

```
SQL> select  count(*)
  2    from airportdelays a
  3   where a.Time.Month.number() = 6
  4*  and a.Time.Year.number() = 2010;

  COUNT(*)
----------
      7424
```

```
Plan hash value: 679434008

-----------------------------------------------------------------
| Id  | Operation         | Name                 | E-Rows |
-----------------------------------------------------------------
|   0 | SELECT STATEMENT  |                      |        |
|   1 |  SORT AGGREGATE   |                      |      1 |
|*  2 |   INDEX RANGE SCAN| AIRPORTDELAYS_INDX01 |   7619 |
-----------------------------------------------------------------
```

```
Statistics
----------------------------------------------------------
              1  DB time
              3  Requests to/from client
             39  non-idle wait count
              2  opened cursors cumulative
              1  opened cursors current
             36  physical read total IO requests
              1  pinned cursors current
             36  session logical reads
              4  user calls
用时：  00:00:00.460
SQL>
```

JSON数据性能

JSON类型的连接 – 单值索引

- 数据行总量 1M & 10K
- 全表扫描 vs. 单值索引
- 10.056 vs. 3.474

全表扫描

```
SQL> explain plan
  2  for
  3  select  count(*)
  4    from airportdelays a,airportdelays_2_merge b
  5   where a.Time.Month.number() = b.Time.Month.number()
  6*  and a.Time.Year.number() = b.Time.Year.number();

已解释。

SQL>
```

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------
Plan hash value: 724543300

-------------------------------------------------------------------------------------------
| Id  | Operation           | Name                 | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                      |     1 |   242 |       | 58563   (1)| 00:00:03 |
|   1 |  SORT AGGREGATE     |                      |     1 |   242 |       |            |          |
|*  2 |   HASH JOIN         |                      |  1128K|   260M|  1304K| 58563   (1)| 00:00:03 |
|   3 |    TABLE ACCESS FULL| AIRPORTDELAYS_2_MERGE |  9999 |  1181K|       |   453   (1)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| AIRPORTDELAYS        |  1128K|   130M|       | 50938   (1)| 00:00:02 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access(JSON_VALUE("A"."TIME" /*+ LOB_BY_VALUE */  FORMAT OSON , '$.Month.number()'
              RETURNING NUMBER NULL ON ERROR)=JSON_VALUE("B"."TIME" /*+ LOB_BY_VALUE */  FORMAT OSON ,
              '$.Month.number()' RETURNING NUMBER NULL ON ERROR) AND JSON_VALUE("A"."TIME" /*+
              LOB_BY_VALUE */  FORMAT OSON , '$.Year.number()' RETURNING NUMBER NULL ON
              ERROR)=JSON_VALUE("B"."TIME" /*+ LOB_BY_VALUE */  FORMAT OSON , '$.Year.number()' RETURNING
              NUMBER NULL ON ERROR))

已选择 21 行。

SQL>
```

```
SQL> select  count(*)
  2    from airportdelays a,airportdelays_2_merge b
  3   where a.Time.Month.number() = b.Time.Month.number()
  4*  and a.Time.Year.number() = b.Time.Year.number();

  COUNT(*)
----------
  74232576

用时：00:00:10.056
SQL>
```

单值索引

```
SQL> explain plan
  2  for
  3  select  count(*)
  4    from airportdelays a,airportdelays_2_merge b
  5   where a.Time.Month.number() = b.Time.Month.number()
  6*  and a.Time.Year.number() = b.Time.Year.number();

已解释。

SQL>
```

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------
Plan hash value: 4133734091

-------------------------------------------------------------------------------------------
| Id  | Operation             | Name                       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                            |     1 |    14 |  1197  (35)| 00:00:01 |
|   1 |  SORT AGGREGATE       |                            |     1 |    14 |            |          |
|*  2 |   HASH JOIN           |                            |    74M|   991M|  1197  (35)| 00:00:01 |
|   3 |    INDEX FAST FULL SCAN| AIRPORTDELAYS_2_MERGE_INDX01 |  9999 | 69993 |     9   (0)| 00:00:01 |
|   4 |    INDEX FAST FULL SCAN| AIRPORTDELAYS_INDX01       |  1128K|  7714K|   779   (2)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access(JSON_VALUE("TIME" /*+ LOB_BY_VALUE */  FORMAT OSON , '$.Month.number()'
              RETURNING NUMBER NULL ON ERROR)=JSON_VALUE("TIME" /*+ LOB_BY_VALUE */  FORMAT OSON ,
              '$.Month.number()' RETURNING NUMBER NULL ON ERROR) AND JSON_VALUE("TIME" /*+ LOB_BY_VALUE */
              FORMAT OSON , '$.Year.number()' RETURNING NUMBER NULL ON ERROR)=JSON_VALUE("TIME" /*+
              LOB_BY_VALUE */  FORMAT OSON , '$.Year.number()' RETURNING NUMBER NULL ON ERROR))

已选择 20 行。

SQL>
```

```
SQL> select  count(*)
  2    from airportdelays a,airportdelays_2_merge b
  3   where a.Time.Month.number() = b.Time.Month.number()
  4*  and a.Time.Year.number() = b.Time.Year.number();

  COUNT(*)
----------
  74232576

用时：00:00:03.474
SQL>
```

JSON数据性能

JSON类型的连接 – 多值索引

- 数据行总量 1M & 10K
- 全表扫描 vs. 多值索引
- 03:14.171 vs. 00:11.214

全表扫描

```
SQL> explain plan
  2  for
  3  with kkk_result as (
  4  select JSON_QUERY(t1.Statistics, '$."Minutes Delayed"."Weather Codes"[0]') ttt1,
  5  JSON_QUERY(t2.Statistics, '$."Minutes Delayed"."Weather Codes"[0]') ttt2
  6  from AIRPORTDELAYS t1, AIRPORTDELAYS_2_MERGE t2
  7  where JSON_EXISTS(t1.Statistics,'$?(@."Minutes Delayed"."Weather Codes"[*] == "CLDY")')
  8  and JSON_EXISTS(t2.Statistics,'$?(@."Minutes Delayed"."Weather Codes"[*] == "RAIN")'))
  9  select count(*)
 10  from kkk_result
 11* where rownum < 100000000;

已解释。

SQL>
```

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
Plan hash value: 2899806858

-------------------------------------------------------------------------------------------------------------
| Id  | Operation              | Name                  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                       |     1 |  1800 |  5105K  (1)| 00:03:20 |
|   1 |  SORT AGGREGATE        |                       |     1 |  1800 |            |          |
|*  2 |   COUNT STOPKEY        |                       |       |       |            |          |
|   3 |    MERGE JOIN CARTESIAN|                       |  1128K|  1936M|  5105K  (1)| 00:03:20 |
|*  4 |     TABLE ACCESS FULL  | AIRPORTDELAYS_2_MERGE |   100 | 90200 |   487   (1)| 00:00:01 |
|   5 |     BUFFER SORT        |                       | 11284 |  9895K|  5105K  (1)| 00:03:20 |
|*  6 |      TABLE ACCESS FULL | AIRPORTDELAYS         | 11284 |  9895K| 51050   (1)| 00:00:02 |
-------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(ROWNUM<100000000)
   4 - filter(JSON_EXISTS2("T2"."STATISTICS" /*+ LOB_BY_VALUE */  FORMAT OSON ,
              '$?(@."Minutes Delayed"."Weather Codes"[*] == "RAIN")' FALSE ON ERROR)=1)
   6 - filter(JSON_EXISTS2("T1"."STATISTICS" /*+ LOB_BY_VALUE */  FORMAT OSON ,
              '$?(@."Minutes Delayed"."Weather Codes"[*] == "CLDY")' FALSE ON ERROR)=1)

已选择 22 行。
```

```
SQL> with kkk_result as (
  2  select JSON_QUERY(t1.Statistics, '$."Minutes Delayed"."Weather Codes"[0]') ttt1,
  3  JSON_QUERY(t2.Statistics, '$."Minutes Delayed"."Weather Codes"[0]') ttt2
  4  from AIRPORTDELAYS t1, AIRPORTDELAYS_2_MERGE t2
  5  where JSON_EXISTS(t1.Statistics,'$?(@."Minutes Delayed"."Weather Codes"[*] == "CLDY")')
  6  and JSON_EXISTS(t2.Statistics,'$?(@."Minutes Delayed"."Weather Codes"[*] == "RAIN")'))
  7  select count(*)
  8  from kkk_result
  9* where rownum < 100000000;

  COUNT(*)
----------
  99999999

用时：00:03:14.171
SQL>
```

多值索引

```
SQL> explain plan
  2  for
  3  with kkk_result as (
  4  select JSON_QUERY(t1.Statistics, '$."Minutes Delayed"."Weather Codes"[0]') ttt1,
  5  JSON_QUERY(t2.Statistics, '$."Minutes Delayed"."Weather Codes"[0]') ttt2
  6  from AIRPORTDELAYS t1, AIRPORTDELAYS_2_MERGE t2
  7  where JSON_EXISTS(t1.Statistics,'$?(@."Minutes Delayed"."Weather Codes"[*] == "CLDY")')
  8  and JSON_EXISTS(t2.Statistics,'$?(@."Minutes Delayed"."Weather Codes"[*] == "RAIN")'))
  9  select count(*)
 10  from kkk_result
 11  where rownum < 100000000;
 12*

已解释。

SQL>
```

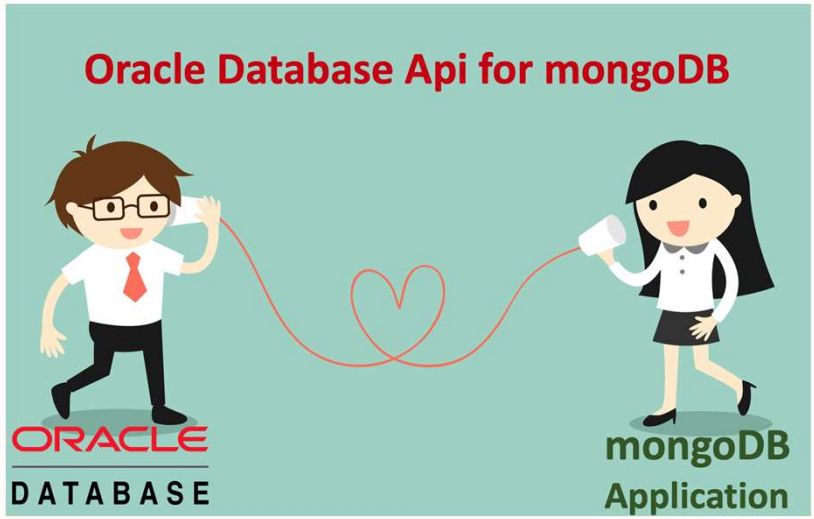
```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
Plan hash value: 918283156

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name                  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                       |     1 |  2151 |     2   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE                |                       |     1 |  2151 |            |          |
|*  2 |   COUNT STOPKEY                |                       |       |       |            |          |
|   3 |    MERGE JOIN CARTESIAN        |                       |     1 |  2151 |     2   (0)| 00:00:01 |
|   4 |     HASH UNIQUE                |                       |     1 |  2151 |            |          |
|*  5 |      INDEX RANGE SCAN (MULTI VALUE) | AIRPORTDELAYS_2_MERGE_INDX02 |  1 |  1077 |     1   (0)| 00:00:01 |
|   6 |     BUFFER SORT                |                       |     1 |  1074 |     1   (0)| 00:00:01 |
|   7 |      HASH UNIQUE               |                       |     1 |  2151 |            |          |
|*  8 |       INDEX RANGE SCAN (MULTI VALUE)| AIRPORTDELAYS_INDX02 |     1 |  1074 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(ROWNUM<100000000)
   5 - access(JSON_QUERY("STATISTICS" /*+ LOB_BY_VALUE */  FORMAT OSON , '$."Minutes Delayed"."Weather
              Codes"[*].string()' RETURNING VARCHAR2(4000) ASIS  WITHOUT ARRAY WRAPPER ERROR ON ERROR NULL ON EMPTY NULL
              ON MISMATCH MULTIVALUE)='RAIN')
   8 - access(JSON_QUERY("STATISTICS" /*+ LOB_BY_VALUE */  FORMAT OSON , '$."Minutes Delayed"."Weather
              Codes"[*].string()' RETURNING VARCHAR2(4000) ASIS  WITHOUT ARRAY WRAPPER ERROR ON ERROR NULL ON EMPTY NULL
              ON MISMATCH MULTIVALUE)='CLDY')
```

```
SQL> with kkk_result as (
  2  select JSON_QUERY(t1.Statistics, '$."Minutes Delayed"."Weather Codes"[0]') ttt1,
  3  JSON_QUERY(t2.Statistics, '$."Minutes Delayed"."Weather Codes"[0]') ttt2
  4  from AIRPORTDELAYS t1, AIRPORTDELAYS_2_MERGE t2
  5  where JSON_EXISTS(t1.Statistics,'$?(@."Minutes Delayed"."Weather Codes"[*] == "CLDY")')
  6  and JSON_EXISTS(t2.Statistics,'$?(@."Minutes Delayed"."Weather Codes"[*] == "RAIN")'))
  7  select count(*)
  8  from kkk_result
  9* where rownum < 100000000;

  COUNT(*)
----------
  99999999

用时：00:00:11.214
SQL>
```

自治数据库 Mongo API

3W

- What?
  - " 像使用MongoDB一样来使用Oracle融合数据库" (限制*)
    - 双向读写，真实连接 – 非复制的机制
    - * Oracle DB的认证/权限机制
    - * SQL来做分析/聚合, 不支持Mongo的AF等 …
- Why?
  - 利用现有的mongoDB开发技能/驱动/工具/框架等 … … 同时您将从Oracle融合数据库获益颇丰
  - 简化迁移到云的操作
- When?
  - 目前有限可用
  - 很快就可以GA!



功能演示 – API for MongoDB

Demo using Autonomous JSON database

Q&A

ORACLE