Cheatography

Oracle 12c Json Support Cheat Sheet

by Jianmin Feng (taotao) via cheatography.com/79308/cs/19440/

Oracle Json support

before 12.1	pljson package
apex 5.0	apex_json
12.1	native support
12.2	simpler, more functions

Store JSON Documents

```
CREATE TABLE json_doc (
  id RAW(16) NOT NULL,
  data CLOB,
  CONSTRAINT json_doc_pk PRIMARY
KEY (id),
  CONSTRAINT json_doc_json_chk
CHECK
   (data IS JSON WITH UNIQUE
KEYS(STRICT))
);
--WITH UNIQUE KEYS: use care
(performance)
--(LAX): Lax syntax by default,
(STRICT) = (case insensitive ,"
equals ' etc)
--ORA- 02290: if constraint
violated when insert
INSERT INTO json_doc (id, data)
VALUES (SYS_GUID(),
'{
    "FirstName" : "John",
    "LastName" : "Doe",
     "Job" : "Clerk",
     "Address" : {
            "Street" : "99 My
Street",
             "City" : "My City",
             "Country" : "USA",
             "Postcode" : "123-
45"},
    "Phones" : [{"Home":"123-4-
56789"},
                {"Cell":"123-45-
6789"}],
     "DateOfBirth" : "01-JAN-19-
80",
     "Active" : true
}');
```

json stored in varchar2, clob; rarely in BLOB,NVARCHAR,NCLOB even supported

Load Json by external table1

```
CREATE TABLE json_doc_ext(data
clob)
ORGANI ZATION EXTERNAL
( TYPE ORACLE _LOADER
DEFAULT DIRECTORY ext_dir
ACCESS PARAMETERS
  ( records delimited by newline
    fields terminated by 0X'09'
    missing field values are
null
    badfile ext_dir:'json_l-
oad.bad'
    logfile ext_dir:'json_l-
oad.log'
    fields (data varchar2(5000)
)
   )
  LOC ATION (ext_dir: 'json_fil-
e.txt')
) REJECT LIMIT UNLIMITED;
--retrieve data
TRUNCATE TABLE json_doc;
INSERT /+ APPEND / INTO json_doc
  SELECT SYS_GUID(), json_d-
ocument
  FROM json_doc_ext
  WHERE data IS JSON;
COMMIT;
```

Load Json by external table2

```
CREATE TABLE json_docs_ext
  (id number, data clob)
ORGANI ZATION EXTERNAL
( TYPE ORACLE _LOADER
DEFAULT DIRECTORY ext_dir
ACCESS PARAMETERS
  ( records delimited by newline
    fields terminated by 0X'09'
    missing field values are
null
    ( id,fname )
      COLUMN TRANSFORMS (
      data FROM LOBFILE(f name)
FROM (ext_dir) )
   )
  LOC ATION (ext_dir: 'json_fil-
e_list.txt')
) REJECT LIMIT UNLIMITED;
--json_file_list.txt format
1 file1.json
2 file2.json
...
--retrieve data
TRUNCATE TABLE json_docs;
INSERT /+ APPEND / INTO json_doc
  SELECT SYS_GUID(), json_d-
ocument
  FROM json_docs_ext
  WHERE data IS JSON;
COMMIT;
```

Cheatography

Oracle 12c Json Support Cheat Sheet

by Jianmin Feng (taotao) via cheatography.com/79308/cs/19440/

Querying Json Data - dot notation

```
SELECT a.data.FirstName,
       a.data.LastName,
       a.data.phones.home AS
home_phone,
       a.data.phones.cell AS
cell_phone
FROM json_doc a
WHERE a.data.ContactDeta-
ils.Phone IS NULL;
--table must be aliased
--col must IS JSON
--key is intensive by default
(LAX)
```

Querying Json Data - JSON_VALUE

```
SELECT JSON_VALUE(
    a.data, '$.Phones'
    RETURNING VARCHAR2(250)
    ERROR ON ERROR
  ) AS contact_phones
FROM json_documents a
ORDER BY 1;
--NULL ON ERROR (default)
--ERROR ON ERROR
--JSON_VALUE returns scalar
value, return complex values (
array, nested records) as null
by default. $.Phones is non
scalar here
--Supproted type: varchar2,num-
ber,date,timestamp,timestamp
with time zone,sdo_geometry,
clob(18c),blob(18c)
--default varchar2(4000)
```

Querying Json Data - JSON_EXISTS

```
SELECT a.data.FirstName,
       a.data.LastName,
       a.data.Phones.cellAS
cell_hone
FROM json_documents a
WHERE JSON_EXISTS( a.data.Ph-
ones,
                   '$.cell' FALSE
ON ERROR)
--FALSE ON ERROR (default)
--TRUE ON ERROR
--ERROR ON ERROR
```

Querying Json Data - JSON_QUERY

```
SELECT
a.data.FirstName,a.data.LastName,
       JSON_QUERY(
          a.data, '$.phones'
          RETURNING VARCHAR2(-
1000)
          WITH WRAPPER
      ) AS contact_Phones
FROM json_documents a
ORDER BY a.data.FirstName,
a.data.Last_name;
--JSON_QUERY could return
multiple values
--return type: varchar2(4000),
clob(18c),blob(18c)
--with wrapper: [ {....} ]
```

Querying Json Data - JSON_TABLE

```
--incoporates
JSON_VALUE,JSON_EXISTS,JSON_QUERY
CREATE OR REPLACE VIEW
json_doc_v AS
SELECT row_number,
   jt.first_name,jt.last_name,
  ,jt.addr_city,jt.addr_country,
   TO_DATE(jt.dob, 'DD-MON-Y-
YYY') AS dob
FROM json_doc,
```

Querying Json Data - JSON_TABLE (cont)

```
  JSON_TABLE(data, '$' COLUMNS (
     row_num FOR ORDINA LITY,
     first_name varchar2(50)
PATH '$.FirstName',
     last_name varchar2(50) PATH
'$.LastName',
     addr_city varchar2(50 CHAR)
           PATH '$.Address.C-
ity',
     addr_country varchar2(50
CHAR)
          PATH '$.Address.Coun-
try',
     dob varchar2(11) PATH
'$.DateOfBirth'
  )) jt;
--NESTED (array) , Wrapper
(nested records)
SELECT jt.first_name,jt.last-
_name,
       jt.address.jt.phones
FROM json_documents,
   JSON_TABLE(data, '$' COLUMNS
(
     first_name varchar2(50)
PATH '$.FirstName',
     last_name varchar2(50) PATH
'$.LastName',
     addres varchar2(4000)
        FORMAT JSON WITH WRAPPER
        PATH '$.address',
     NESTED PATH '$.phones[*]'
COLUMNS (
         home_phone varchar2(12)
PATH '$.home',
         cell_phone varchar2(12)
PATH '$.cell',
     )
  )) jt;
```

12.2 rewite JSON_TABLE for fewer call to improve performance

Cheatography

Oracle 12c Json Support Cheat Sheet

by Jianmin Feng (taotao) via cheatography.com/79308/cs/19440/

JSON_TEXTCONTAINS

```
1. create context/full text
search index
--12.1
CREATE INDEX json_search_idx ON
json_doc(data)
INDEXTYPE IS CTXSYS.CONTEXT
PARAMETERS ('section group
  CTXSYS.JSON_SECTION_GROUP SYNC
(ON COMMIT)');
--12.2
CREATE INDEX json_search_idx ON
json_doc(data) FOR JSON;
2. collect stats
EXEC DBMS_STATS.gather_table_-
stats(user, 'JSON_DOC');
3. query
SELECT COUNT(*) FROM json_doc
WHERE JSON_TEXTCONTAINS( data,
        '$.phones.home', '123-4-
56789');
SELECT COUNT(*) FROM json_doc
WHERE JSON_EXISTS( data, '$.pho-
nes');
```

FORMAT JSON clause

BLOB	explicit
CLOB,VARCHAR2	implicit

Dot Notation query Transformation

```
ALTER SESSION SET EVENTS '10053
trace name context forever';
SELECT a.data.FirstName,
       a.data.LastName
FROM json_documents a;
ALTER SESSION SET EVENTS '10053
trace name context off';
```

Dot Notation query Transformation (cont)

```
-----
Final query after transformati-
ons: UNPARSED QUERY
SELECT
   JSON_QUERY("A"."DATA" FORMAT
JSON ,
     '$.FirstName' RETURNING
VARCHAR2(4000)
     ASIS WITHOUT ARRAY WRAPPER
     NULL ON ERROR) "FIRSTN-
AME",
   JSON_QUERY("A"."DATA" FORMAT
JSON ,
     '$.LastName' RETURNING
VARCHAR2(4000)
      ASIS WITHOUT ARRAY WRAPPER
      NULL ON ERROR) "LASTNAME"
FROM "TEST"."JSON_DOCUMENTS" "A"
```

Dot notation is automatically transformed to json_query or json_table for performance and index usage. It is good practice to avoid dot notion totally.

Identifying Columns Containing JSON

```
SELECT table_name,
       column_name,
       format,
       data_type
FROM user_json_columns;
--{USER|ALL|DBA}_JSON_COLUMNS
```

SQL/JSON Generator functions

```
JSON_OBJECT (KEY 'key_val' VALUE
t.col,....)
JSON_OBJECTAGG (KEY t.col1 VALUE
t.col2)
JSON_ARRAY(
     ROWNUM,
     JSON_OBJECT(KEY 'key_val'
VALUE t.col),
     ...
 )
JSON_ARRAYAGG(KEY t.col1 VALUE
t.col2)
-- NULL ON NULL ( default)
ABSENT ON NULL
--RETURNING VARCHAR2(4000) by
default
--RETURNING VARCHAR2(32767)
--CLOB(AGG function 12c, all for
18c), BLOB (18c)
--FORMAT JSON required for BLOB
only
--TO_CHAR(e.empno) to use number
key
SELECT JSON_OBJECT (
  KEY 'departments' VALUE (
     SELECT JSON_ARRAYAGG(
        JSON_OBJECT (
          KEY 'department_name'
VALUE d.dname,
          KEY 'department_no'
VALUE d.deptno,
          KEY 'employees' VALUE
(
            SELECT JSON_ARRAYAGG
(
               JSON_OBJECT(
                  KEY 'emp_no'
VALUE e.empno,
                  KEY 'emp_name'
VALUE e.ename))
            FROM emp e
            WHERE e.deptno =
d.deptno)
         ))FROM dept d
)) AS departments
FROM dual;
```

SQL/JSON Generator functions (cont)

Cheatography

Oracle 12c Json Support Cheat Sheet

by Jianmin Feng (taotao) via cheatography.com/79308/cs/19440/

Oracle 18c JSON support

JSON_VALUE return	clob
JSON_QUERY return	clob, blob
SQL/JSON type	clob, blob
JSON_EQUAL Condition	new function
JSON_TABLE	simpler syntax
SODA for PL/SQL	new function

https://oracle-base.com/articles/18c/json--support-in-oracle-database-18c

JSON_EQUAL

```
--compare independent of order and format
--18c
CREATE TABLE json_equal_tab (
  id NUMBER NOT NULL,
  data1 VARCHAR2(50),
  data2 VARCHAR2(50),
  CONSTRAINT json_equal_tab_pk PRIMARY KEY (id),
  CONSTRAINT json_equal_tab_json1_chk
       CHECK (data1 IS JSON),
  CONSTRAINT json_equal_tab_json2_chk
       CHECK (data2 IS JSON)
);
-- Matching members, order and format.
INSERT INTO json_equal_tab
VALUES (2, '{"name1":"value1","name2":"value2"}', '{"name1":"value1","name2":"value2"}');
-- Matching members/order, but differing format.
INSERT INTO json_equal_tab
VALUES (3, '{"name1":"value1","name2":"value2"}',
'{ "name1":"value1", "name2":"value2" }');
-- Matching members, but differing order.
```

JSON_EQUAL (cont)

```
INSERT INTO json_equal_tab
VALUES (4, '{"name1":"value1","name2":"value2"}', '{"name2":"value2","name1":"value1"}');
--return 1 row
select *
from json_equal_tab
where data1=data2;
 --return 3 rows
select *
from json_equal_tab
where JSON_EQUAL(data1, data2);
```

Current JSON_EQUAL only valid in where or case when clause, no native PL/SQL support yet.