PGDAY'17 RUSSIA

DataArt

DataArt

Technology Consulting & Solution Design

New York USA
London UK
Munich Germany
Zug Switzerland

We Help Clients Achieve Important Business Outcomes by

DataArt

- Building New Products and Services
- Modernizing and Re-engineering Legacy systems
- Consulting on New Technology Approaches
- Providing On-Demand IT
- Controlling Costs via Managed Support Services

JSON caveats

Oracle 12.1

Agenda

DataArt

1. JSON in relational storage
2. RDMS Configuration
3. Storage
4. Ingestion
5. Retrieval
6. Search
7. Maintenance
8. Fast search
9. Summary
10. Q&A session

JSON in RDBMS

DataArt

Why

- Consistency (integrity, transaction ACIDity) for storing JSON documents
- Denormalization of complex objects

What

- Logs with responses/requests received/sent during software interaction
- Configuration data, key/value user preferences
- Unstructured or semi-structured complex objects

and please forget about any analytics by JSON fields ☺

DB configuration

DataArt

- Install JSON fixes on regular base
- Retain scripts to check old issues – new fixes restores them often
- These patches MUST be installed
  1. Patch 20080249: JSON Patch Bundle 1
  2. Patch 20885778: JSON Patch Bundle 2
  3. Patch 24836374: JSON Patch Bundle 3

Oracle thinks JSON is stable now so no more dedicated JSON Bundle patches!

Fixes are inside ordinal database proactive bundle patches (Doc ID 1937782.1).

Table structure

DataArt

```
create   table json_storage_vc (
id number not null,
created_dtm timestamp not null,
json_data varchar2(4000)
--32767
-- please take into account only 3.5K are stored inline
);
```

```
create   table json_storage_cl (
id number not null,
created_dtm timestamp not null,
json_data clob
);
```

```
create   table json_storage_bl (
id number not null,
created_dtm timestamp not null,
json_data blob
);
```

```
{
    "glossary": {
        "title": "example glossary",
                "GlossDiv": {
            "title": "S",
                        "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
                                        "SortAs": "SGML",
                                        "GlossTerm": "Standard Generalized Markup Language",
                                        "Acronym": "SGML",
                                        "Abbrev": "ISO 8879:1986",
                                        "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
                                                "GlossSeeAlso": ["GML", "XML"]
                    },
                                        "GlossSee": "markup"
                }
            }
        }
    }
}
```

10000

```
SELECT  SEGMENT_NAME, ROUND(SUM(DS.BYTES) / (1024 * 1024)) AS MB
  FROM DBA_SEGMENTS DS
  WHERE SEGMENT_NAME IN ('JSON_STORAGE_VC', 'JSON_STORAGE_CL', 'JSON_STORAGE_BL')
 GROUP BY DS.TABLESPACE_NAME,
       SEGMENT_NAME;
```

ript Output | Query Result

SQL | All Rows Fetched: 3 in 0.077 seconds

	SEGMENT_NAME	MB
1	JSON STORAGE CL	16
2	JSON STORAGE VC	8
3	JSON STORAGE BL	8

UCS2

Table structure

DataArt

BLOB for JSON benefits:

- Twice less space consumption
- less I/O due less space
- No implicit character-set conversion if the database character set is not AL32UTF8

RFC 4627

3. Encoding

JSON text SHALL be encoded in Unicode. The default encoding is UTF-8.

Table structure

DataArt

```
insert into perso.json_storage
   values(
   1,
   systimestamp,
   utl_raw.cast_to_raw(
   '
{
    "widget": {
        "debug": "on",
        "window": {
            "title": "Sample Widget",
            "name": "main_window",
            "width": 500,
            "height": 500
        },
        "image": {
            "src": "Images/Sun.png",
            "name": "sun1",
            "hOffset": .250,
            "vOffset": .250,
            "alignment": "center"
        }
    }
}
   ')
   );
   1 row inserted;
   commit;
```

```
ALTER TABLE json_storage ADD CONSTRAINT ck_json CHECK (json_data IS JSON);
```

```
select *
from user_constraints
where lower(constraint_name) = 'ck_json'
```

ipt Output × | Query Result ×

SQL | All Rows Fetched: 1 in 0.481 seconds

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	SEARCH_CONDITION_VC
PERSO	CK_JSON	C	JSON_STORAGE	json_data IS JSON	json_data IS JSON

```
insert into perso.json_storage
   values(
   1,
   systimestamp,
   utl_raw.cast_to_raw(
   '
{
    "widget":
   ')
   );

Error report -
SQL Error: ORA-02290: check constraint (PERSO.CK_JSON) violated
02290. 00000 -  "check constraint (%s.%s) violated"
*Cause:    The values being inserted do not satisfy the named check

*Action:   do not insert values that violate the constraint
```

Constraint works fine but JAVA still fails ☹

Table structure

39.5.2.4 About Strict and Lax JSON Syntax

The Oracle default syntax for JSON is lax. In particular: it reflects the JavaScript syntax for object fields; the Boolean and null values are not case-sensitive; and it is more permissive with respect to numerals, whitespace, and escaping of Unicode characters.

```
ALTER TABLE json_storage ADD CONSTRAINT ck_json CHECK (json_data IS JSON strict);
```

Ingestion

DataArt

- Oracle treats JSON as string – no tailored object type
- Insert works fine
- No option to update a piece of JSON

```
update json_storage a
set a.JSON_DATA.menu.id = '25'
where id = 1
```

```
Error report -
SQL Error: ORA-40557: cannot update a JSON value
```

Ingestion

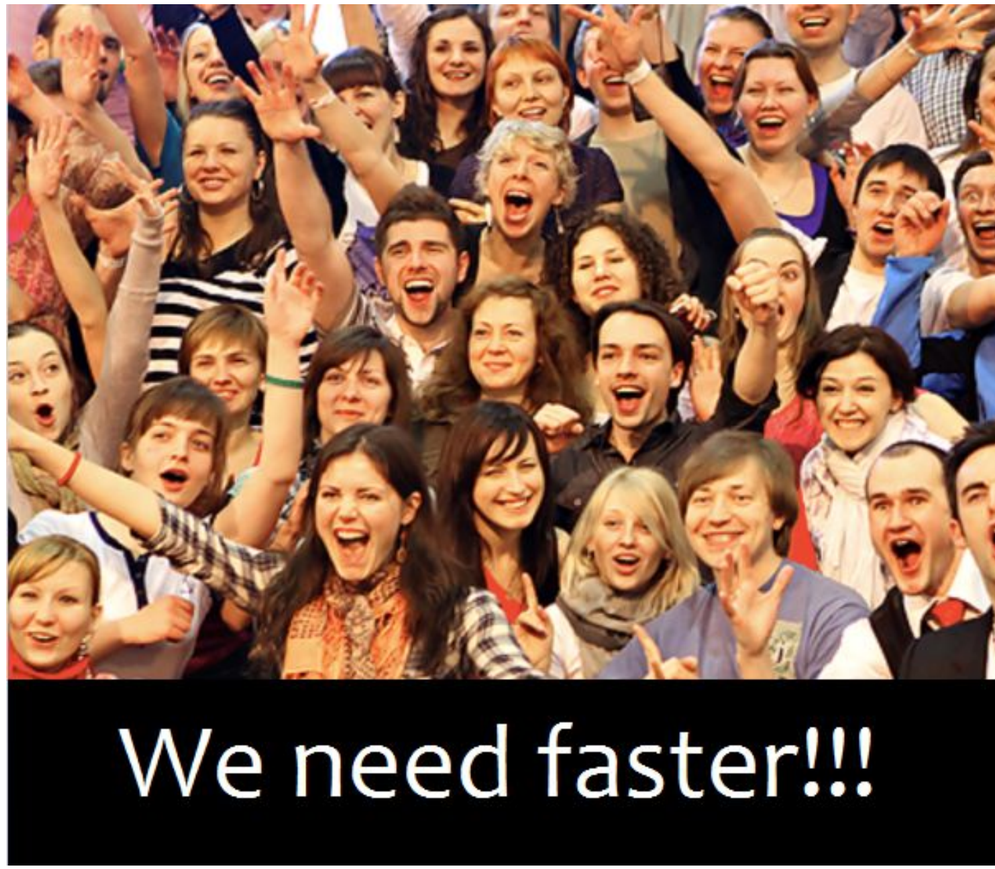
DataArt

```
declare
 lStart number;
begin
 lStart := DBMS_UTILITY.get_time;

 for lCounter in 1..10000 loop
   insert into perso.json_storage
   values(
   lCounter,
   systimestamp,
   utl_raw.cast_to_raw('
                         {"menu": {
                        "id": ' || lCounter  ||',
                        "value": "File",
                        "popup": {
                          "menuitem": [
                            {"value": "New", "onclick": "CreateNewDoc()"}
                            {"value": "Open", "onclick": "OpenDoc()"},
                            {"value": "Close", "onclick": "CloseDoc()"}
                          ]
                        }
                      }}
                       '
                    )
   );
 end loop;

 DBMS_OUTPUT.put_line('Elapsed: ' ||  (DBMS_UTILITY.get_time - lStart));
end;

ALTER TABLE json_storage ADD CONSTRAINT ck_json CHECK (json_data IS JSON strict);
aLTER TABLE json_storage drop CONSTRAINT ck_json;
ALTER TABLE json_storage ADD CONSTRAINT ck_json CHECK (json_data IS JSON LAX);
aLTER TABLE json_storage drop CONSTRAINT ck_json;
ALTER TABLE json_storage ADD CONSTRAINT ck_json CHECK (json_data IS JSON with unique keys);
aLTER TABLE json_storage drop CONSTRAINT ck_json;
```

Ingestion

DataArt

N, run	with json strict	with json lax	with json lax and unique names	without constraints
1	115	121	132	83
2	119	117	142	80
3	119	115	132	91
4	115	110	136	90
5	117	125	138	92
6	122	117	135	90
7	116	117	134	88
8	127	120	142	81
9	115	125	152	80
10	118	114	147	83
AVG	***118,3***	***118,1***	***139***	***85,8***



Ingestion

DataArt

```
select *
from user_lobs
where table_name = 'JSON_STORAGE'
```

Query Result

SQL | All Rows Fetched: 1 in 0.109 seconds

	TABLE_NAME	COLUMN_NAME	SEGMENT_NAME	TABLESPACE_NAME	INDEX_NAME	CHUNK	PCTVERSION	RETENTION	FREEPOOLS	CACHE
1	JSON_STORAGE	JSON_DATA	SYS_LOB0000099977C00003$$	USERS	SYS_IL0000099977C00003$$	8192	(null)	(null)	(null)	NO

`CACHE`	`VARCHAR2(10)`	Indicates whether and how the LOB data is to be cached in the buffer cache: • `YES` - LOB data is placed in the buffer cache • `NO` - LOB data either is not brought into the buffer cache or is brought into the buffer cache and placed at the least recently used end of the LRU list • `CACHEREADS` - LOB data is brought into the buffer cache only during read operations but not during write operations

```
select table_name, column_name
from user_lobs
where cache <> 'YES'
```

Ingestion

DataArt

N, run	with json strict	with json lax	with json lax and unique names	without constraints	with cache
1	115	121	132	83	78
2	119	117	142	80	78
3	119	115	132	91	84
4	115	110	136	90	75
5	117	125	138	92	78
6	122	117	135	90	75
7	116	117	134	88	75
8	127	120	142	81	78
9	115	125	152	80	77
10	118	114	147	83	77
AVG	***118,3***	***118,1***	***139***	***85,8***	***77,5***

Retrieval

DataArt

- Extract 1 row with raw JSON data and pass it to application server as is
- Issue SQL statement which extracts 1 row from table and parses it via Oracle JSON feature
- Create a view which encapsulates JSON treatment and extract a row from the view

Retrieval

DataArt

```
select json_data.menu.id from  json_storage
```

```
ORA-00904: "JSON_DATA"."MENU"."ID": invalid identifier
```

```
select a.json_data.menu.id from  json_storage a
```

```
ORA-00904: "A"."JSON_DATA"."MENU"."ID": invalid identifier
```

Nothing changes

```
ALTER TABLE json_storage ADD CONSTRAINT ck_json CHECK (json_data IS JSON LAX);
```

	MENU
1	25
2	26
3	27
4	28
5	29
6	30
7	31
8	32
9	33

Retrieval

DataArt

```
select json_value(json_data, '$.menu.id') id from  json_storage a
```

	ID
1	25
2	26
3	27
4	28

```
select json_value(json_data, '$.menu.id') id, json_value(json_data, '$.menu.value') val, json_value(json_data, '$.menu.popup.menuitem[0].value')val2
from  json_storage a
```

	ID	VAL	VAL2
1	25	File	New
2	26	File	New
3	27	File	New
4	28	File	New
5	29	File	New
6	30	File	New

Bad approach – each json_value function parses JSON again!

The same for .notation!

```
select a.json_data.menu.popup.menuitem[0].value from  json_storage a    ORA-00923: FROM keyword not found where expected
```

Retrieval

DataArt

Storage logic could be encapsulated inside virtual columns

```
create or replace function get_2nd_item(pJson json_storage.json_data%type) return varchar2  deterministic is
 lResult varchar2(1000);
begin
 select json_value(pJson, '$.menu.popup.menuitem[1].value')
 into lResult
 from dual;  --json_value isn't permitted in plsql

 return lResult;
end;
```

```
ALTER TABLE json_storage ADD (j_id VARCHAR2(100)
    GENERATED ALWAYS AS (JSON_VALUE(json_data, '$.menu.id' RETURNING VARCHAR2(100))))

ALTER TABLE json_storage ADD (j_item VARCHAR2(100)
    GENERATED ALWAYS AS (get_2nd_item(json_data) ))
```

```
select * from json_storage
```

	ID	CREATED_DTM	JSON_DATA	J_ITEM	J_ID
1	25	27-JUN-17 03.32.51.568262000 AM	(BLOB)	Open	25

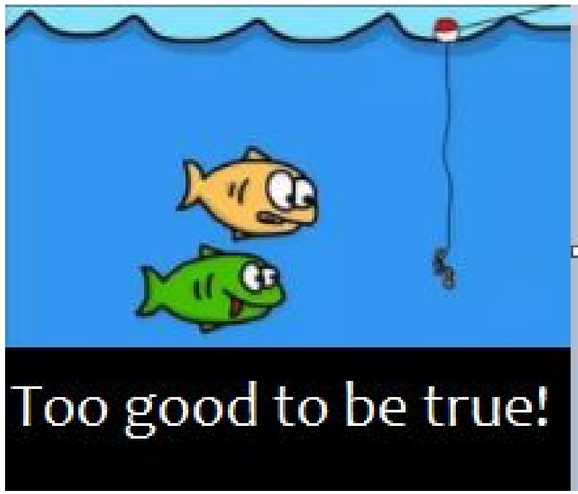
Retrieval

DataArt

```
select s.id,
       jt.*
from json_storage s,
      JSON_TABLE(json_data, '$.menu'
         COLUMNS (code    VARCHAR2(50 CHAR) PATH '$.id',
                  val     VARCHAR2(50 CHAR) PATH '$.value'
                 )
                ) jt
```

SQL | Fetched

	ID	CODE	VAL
1	25	25	File
2	26	26	File
3	27	27	File
4	28	28	File
5	29	29	File




```
SELECT  *
FROM  JSON_TABLE(
                 '
                  {
                   "data":[[26,"Alex",2],[17,"Tom",1],[31,"Jane",1],[18,"Dim",1]]
                  }
                 ',
                 '$.data[*]'
                 columns
                   id VARCHAR2(20) PATH '$[0]',
                   name VARCHAR2(20) PATH '$[1]',
                   type VARCHAR2(20) PATH '$[2]'
                )
```

	ID	NAME	TYPE
1	26	Alex	2
2	17	Tom	1
3	31	Jane	1
4	18	Dim	1

Retrieval

DataArt

```
select s.id,
       jt.*,
       jt2.*
from json_storage s,
      JSON_TABLE(json_data, '$.menu'
         COLUMNS (code    VARCHAR2(50 CHAR) PATH '$.id',
                  val     VARCHAR2(50 CHAR) PATH '$.value',
                  items  varchar2(4000) format json path '$.popup.menuitem'
                 )
                ) jt,
      json_table(jt.items, '$.menu.popup.menuitem[*]'
                 columns item VARCHAR2(50 CHAR)  path '$.value'
                ) jt2
```

```
ORA-40556: unsupported chaining of JSON_TABLE
```

```
with tmp as (
  select    s.id,
            jt.*
  from json_storage s,
        JSON_TABLE(json_data, '$.menu'
           COLUMNS (code    VARCHAR2(50 CHAR) PATH '$.id',
                    val     VARCHAR2(50 CHAR) PATH '$.value',
                    items  varchar2(4000) format json path '$.popup.menuitem'
                   )
                  ) jt
    )
   select /*+ no_merge(t)*/
          t.*
        , jt2.item
   from tmp t
      , json_table(t.items,
          '$[*]'
          columns item varchar2(50) path '$.value'
        ) jt2
```

	ID	CODE	VAL	ATTR1
1	25	25	File	New
2	25	25	File	Open
3	25	25	File	Close
4	26	26	File	New
5	26	26	File	Open
6	26	26	File	Close

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		667G	1348T	2217M (1)	24:03:34
1	NESTED LOOPS		667G	1348T	2217M (1)	24:03:34
2	VIEW		81M	168G	271K (1)	00:00:11
3	NESTED LOOPS		81M	46G	271K (1)	00:00:11
4	TABLE ACCESS FULL	JSON_STORAGE	10004	5969K	238 (0)	00:00:01
5	JSONTABLE EVALUATION					
6	JSONTABLE EVALUATION					

Retrieval

DataArt

```
select s.id,
        jt.*,
        jt2.*
 from json_storage s,
       JSON_TABLE(json_data, '$.menu'
          COLUMNS (code    VARCHAR2(50 CHAR) PATH '$.id',
                   val     VARCHAR2(50 CHAR) PATH '$.value'
                  )
                 ) jt,
       json_table(json_data, '$.menu.popup.menuitem[*]'
                  columns attr1 VARCHAR2(50 CHAR)  path '$.value'
                 ) jt2
```

	ID	CODE	VAL	ATTR1
1	25	25	File	New
2	25	25	File	Open
3	25	25	File	Close
4	26	26	File	New
5	26	26	File	Open
6	26	26	File	Close

```
-------------------------------------------------------------------------------------------
| Id  | Operation                | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |              |   667G|   374T|  2217M  (1)| 24:03:34 |
|   1 |  NESTED LOOPS            |              |   667G|   374T|  2217M  (1)| 24:03:34 |
|   2 |   NESTED LOOPS           |              |    81M|    46G|   271K  (1)| 00:00:11 |
|   3 |    TABLE ACCESS FULL     | JSON_STORAGE | 10004 |  5969K|   238   (0)| 00:00:01 |
|   4 |    JSONTABLE EVALUATION  |              |       |       |            |          |
|   5 |   JSONTABLE EVALUATION   |              |       |       |            |          |
-------------------------------------------------------------------------------------------
```

Retrieval

DataArt

```
select s.id,
       jt.*
from json_storage s,
      JSON_TABLE(json_data, '$.menu'
         COLUMNS (code    VARCHAR2(50 CHAR) PATH '$.id',
                  val     VARCHAR2(50 CHAR) PATH '$.value',
                   nested    path '$.popup.menuitem[*]'  columns (
                                                          val_item  varchar2(100) path '$.value'
                                                         )
                 )
                ) jt
```

ID	CODE	VAL	ATTR1
1	25	25	File	New
2	25	25	File	Open
3	25	25	File	Close
4	26	26	File	New
5	26	26	File	Open
6	26	26	File	Close

```
-------------------------------------------------------------------------------------
| Id  | Operation              | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |              |    81M|    46G|   271K  (1)| 00:00:11 |
|   1 |  NESTED LOOPS          |              |    81M|    46G|   271K  (1)| 00:00:11 |
|   2 |   TABLE ACCESS FULL    | JSON_STORAGE | 10004 |  5969K|   238   (0)| 00:00:01 |
|   3 |   JSONTABLE EVALUATION |              |       |       |            |          |
-------------------------------------------------------------------------------------
```

Retrieval

DataArt

Views

1. Often become non mergable so performance degrades
2. If 2 or more *json_table* are used exception doesn't occur but results could be wrong in aggregate functions. *no_merge* hint helps sometimes.
3. *ORA-600* and *ORA-7445 No Data to be read from socket* arise in arbitrary places
4. Count(distinct <field>) fails with *ORA-7445 No Data to be read from socket.* Could be fixed by removing *group by*, adding *row_number() over (partition by <group by fields>) rn* and filtering out records where *rn <> 1*

Search

DataArt

```
select * from json_storage where json_value(json_data, '$.menu.id') = 3500

-------------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |              |     1 |   622 |   241   (1)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| JSON_STORAGE |     1 |   622 |   241   (1)| 00:00:01 |
-------------------------------------------------------------------------------------

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

   1 - filter(TO_NUMBER(JSON_VALUE("JSON_DATA" FORMAT JSON , '$.menu.id'
              RETURNING VARCHAR2(4000) NULL ON ERROR))=3500)
```

```
create index json_storage_idx on json_storage (json_value(json_data, '$.menu.id' returning number ERROR on ERROR) ) ;
```

Search

DataArt

```
create  index json_storage_idx_2 on json_storage (to_number(json_value(json_data, '$.menu.id' returning varchar2(4000) null on error) )) ;

select * from json_storage where json_value(json_data, '$.menu.id') = 3500
```

```
-----------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                    |   600 |   364K|    22   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| JSON_STORAGE       |   600 |   364K|    22   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                  | JSON_STORAGE_IDX_2 |   240 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

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

   2 - access(TO_NUMBER(JSON_VALUE("JSON_DATA" FORMAT JSON , '$.menu.id' RETURNING VARCHAR2(4000)
              NULL ON ERROR))=3500)
```

```
create  index json_storage_idx_3 on json_storage (json_value(json_data, '$.menu.id' returning varchar2(4000) null on error) ) ;

select * from json_storage where json_value(json_data, '$.menu.id') = '3500'
```

```
-----------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                    |   600 |   364K|    61   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| JSON_STORAGE       |   600 |   364K|    61   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                  | JSON_STORAGE_IDX_3 |   240 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

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

   2 - access(JSON_VALUE("JSON_DATA" FORMAT JSON , '$.menu.id' RETURNING VARCHAR2(4000) NULL ON
              ERROR)='3500')
```

Search

DataArt

```
select * from json_storage a where a.json_data.menu.id = 3500

-------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |              |     1 |   622 |   241   (1)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| JSON_STORAGE |     1 |   622 |   241   (1)| 00:00:01 |
-------------------------------------------------------------------------------

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

   1 - filter(TO_NUMBER(JSON_QUERY("A"."JSON_DATA" FORMAT JSON ,
              '$.menu.id' RETURNING VARCHAR2(4000) ASIS  WITHOUT ARRAY WRAPPER NULL ON
              ERROR))=3500)

select * from json_storage a where a.json_data.menu.id = '3500'
-------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |              |     1 |   622 |   241   (1)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| JSON_STORAGE |     1 |   622 |   241   (1)| 00:00:01 |
-------------------------------------------------------------------------------

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

   1 - filter(JSON_QUERY("A"."JSON_DATA" FORMAT JSON , '$.menu.id'
              RETURNING VARCHAR2(4000) ASIS  WITHOUT ARRAY WRAPPER NULL ON
              ERROR)='3500')
```

Separate indexes should be created for .notation?

Search

DataArt

```
create index json_storage_idx on json_storage (json_data.menu.id)      ORA-15179: missing or invalid alias name

create index json_storage_idx_2 on json_storage j (j.json_data.menu.id) ;

select * from json_storage a where a.json_data.menu.id = '3500'
```

```
-------------------------------------------------------------------------------------------------------
| Id  | Operation                             | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                      |                   |   600 |   364K|     61   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| JSON_STORAGE      |   600 |   364K|     61   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                    | JSON_STORAGE_IDX |   240 |       |      1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------

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

   2 - access(JSON_QUERY("JSON_DATA" FORMAT JSON , '$.menu.id' RETURNING VARCHAR2(4000) ASIS
              WITHOUT ARRAY WRAPPER NULL ON ERROR)='3500')
```

Search

DataArt

```
create  index json_storage_idx_4 on json_storage (json_value(json_data, '$.menu.id' returning varchar2(4000) error on error) ) ;
```

```
select * from json_storage a where a.json_data.menu.id = '3500'
```

```
-------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                    |   100 | 63200 |    12   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| JSON_STORAGE       |   100 | 63200 |    12   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                  | JSON_STORAGE_IDX_4 |    40 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------

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

   2 - access(JSON_VALUE("JSON_DATA" FORMAT JSON , '$.menu.id' RETURNING VARCHAR2(4000) ERROR ON
              ERROR)='3500')
```

Query rewrite

```
2 - access(JSON_QUERY("JSON_DATA" FORMAT JSON , '$.menu.id' RETURNING VARCHAR2(4000) ASIS
           WITHOUT ARRAY WRAPPER NULL ON ERROR)='3500')
```

Search

DataArt

Use index side effect – create JSON validation

```
create index json_storage_idx_4 on json_storage (json_value(json_data, '$.menu.id' returning number ERROR on ERROR) ) ;

insert into perso.json_storage
   values(
   -2,
   systimestamp,
   utl_raw.cast_to_raw('
                         {"menu": {
                        "id": "' || '1a'  ||'",
                        "value": "File",
                        "popup": {
                          "menuitem": [
                            {"value": "New", "onclick": "CreateNewDoc()"},
                            {"value": "Open", "onclick": "OpenDoc()"},
                            {"value": "Close", "onclick": "CloseDoc()"}
                          ]
                        }
                      }}
                       '
                    )
   );
```

Validator ☺

ORA-01722: invalid number

Search

DataArt

Multiple columns indexing

```
create index json_storage_idx
on json_storage (json_value(json_data, '$.menu.id' returning varchar2(100) null on error),
                 json_value(json_data, '$.menu.value'  returning varchar2(100)  null on error)
                );

select *
from json_storage
where json_value(json_data, '$.menu.id' returning varchar2(100) ) = '2000'
      and json_value(json_data, '$.menu.value' returning varchar2(100) ) = 'File'
```


```
-------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name             | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                  |    36 | 22392 |    10   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| JSON_STORAGE     |    36 | 22392 |    10   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                  | JSON_STORAGE_IDX |    36 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------

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

   2 - access(JSON_VALUE("JSON_DATA" FORMAT JSON , '$.menu.id' RETURNING VARCHAR2(100) NULL ON
              ERROR)='2000' AND JSON_VALUE("JSON_DATA" FORMAT JSON , '$.menu.value' RETURNING VARCHAR2(100)
              NULL ON ERROR)='File')
```

Search

DataArt

```
ALTER TABLE json_storage ADD (j_id VARCHAR2(100)
  GENERATED ALWAYS AS (JSON_VALUE(json_data, '$.menu.id' RETURNING VARCHAR2(100))));

ALTER TABLE json_storage ADD (j_value VARCHAR2(100)
  GENERATED ALWAYS AS (JSON_VALUE(json_data, '$.menu.value' RETURNING VARCHAR2(100))));


CREATE INDEX json_storage_c_idx ON json_storage (j_id, j_value);


select * from json_storage  where j_id = '2000' and j_value = 'File'
```

```
-----------------------------------------------------------------------------------------------------
| Id  | Operation                            | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                     |                    |    36 | 22392 |    10   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| JSON_STORAGE       |    36 | 22392 |    10   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                   | JSON_STORAGE_C_IDX |    36 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

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

   2 - access("J_ID"='2000' AND "J_VALUE"='File')
```

```
select column_name, c.data_type from dba_tab_cols c where lower(table_name) = 'json_storage' and virtual_column = 'YES'
```

	COLUMN_NAME	DATA_TYPE
1	SYS_NC00006$	VARCHAR2
2	SYS_NC00005$	VARCHAR2
3	SYS_NC00004$	NUMBER

Search

DataArt

```
select s.*,
       jt.*
from json_storage s,
      JSON_TABLE(json_data, '$.menu'
         COLUMNS (code    VARCHAR2(50 CHAR) PATH '$.id',
                  val     VARCHAR2(50 CHAR) PATH '$.value'
                 )
                ) jt
where jt.code = '3500'
```

```
-------------------------------------------------------------------------------------
| Id  | Operation              | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |              |  4901K|  2926M|  1632K  (1)| 00:01:04 |
|   1 |  NESTED LOOPS          |              |  4901K|  2926M|  1632K  (1)| 00:01:04 |
|   2 |   TABLE ACCESS FULL    | JSON_STORAGE | 60005 |    35M|   238   (0)| 00:00:01 |
|*  3 |   JSONTABLE EVALUATION |              |       |       |            |          |
-------------------------------------------------------------------------------------

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

   3 - filter("P"."CODE"='3500')
```

Search

DataArt

```
CREATE  INDEX json_fts_idx  ON json_storage (json_data)  INDEXTYPE IS CTXSYS.CONTEXT    PARAMETERS ('section group CTXSYS.JSON_SECTION_GROUP')
```

```
select JSON_VALUE(json_data, '$.menu.id')  from json_storage  where json_textcontains(json_data, '$.menu.id', '10')
```

```
-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |              |     1 |   619 |    20   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| JSON_STORAGE |     1 |   619 |    20   (0)| 00:00:01 |
|*  2 |   DOMAIN INDEX               | JSON_FTS_IDX |       |       |    20   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

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

   2 - access("CTXSYS"."CONTAINS"("JSON_STORAGE"."JSON_DATA",'10
              INPATH(/menu/id)')>0)
```

```
select *  from json_storage  where JSON_value(json_data, '$.menu.id') = '10'
```

```
-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |              |     1 |   632 |    20   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| JSON_STORAGE |     1 |   632 |    20   (0)| 00:00:01 |
|*  2 |   DOMAIN INDEX               | JSON_FTS_IDX |       |       |    20   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

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

   1 - filter(JSON_VALUE("JSON_DATA" FORMAT JSON , '$.menu.id' RETURNING
              VARCHAR2(4000) NULL ON ERROR)='10')
   2 - access("CTXSYS"."CONTAINS"("JSON_STORAGE"."JSON_DATA",'{10}
              INPATH(/menu/id)')>0)
```

```
select *  from json_storage  where JSON_exists(json_data, '$.menu.id')
```

```
-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |              |  1399 |  863K|   413   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| JSON_STORAGE |  1399 |  863K|   413   (0)| 00:00:01 |
|*  2 |   DOMAIN INDEX               | JSON_FTS_IDX |       |       |   100   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

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

   2 - access("CTXSYS"."CONTAINS"("JSON_STORAGE"."JSON_DATA",'HASPATH(/menu/id)')>0)
```

Search

DataArt

```
select s.*,
       jt.*
 from json_storage s,
      JSON_TABLE(json_data, '$.menu'
         COLUMNS (code    VARCHAR2(50 CHAR) PATH '$.id',
                  val     VARCHAR2(50 CHAR) PATH '$.value'
                 )
                ) jt
 where jt.code = '10'
```

```
-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |              |    11 |  6996 |    49   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                 |              |    11 |  6996 |    49   (0)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| JSON_STORAGE |     1 |   632 |    20   (0)| 00:00:01 |
|*  3 |    DOMAIN INDEX               | JSON_FTS_IDX |       |       |    20   (0)| 00:00:01 |
|*  4 |   JSONTABLE EVALUATION        |              |       |       |            |          |
-------------------------------------------------------------------------------------------

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

   3 - access("CTXSYS"."CONTAINS"("S"."JSON_DATA",'{10} INPATH(/menu/id)')>0)
   4 - filter("P"."CODE"='10')
```

```
select * from json_storage where JSON_value(json_data, '$.menu.id') = '1' and JSON_value(json_data, '$.menu.value') = 'File'
```

```
-----------------------------------------------------------------------------------------
| Id  | Operation                     | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |              |     1 |   632 |    75   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| JSON_STORAGE |     1 |   632 |    75   (0)| 00:00:01 |
|*  2 |   DOMAIN INDEX                | JSON_FTS_IDX |       |       |    75   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

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

   1 - filter(JSON_VALUE("JSON_DATA" FORMAT JSON , '$.menu.id' RETURNING
              VARCHAR2(4000) NULL ON ERROR)='1' AND JSON_VALUE("JSON_DATA" FORMAT JSON ,
              '$.menu.value' RETURNING VARCHAR2(4000) NULL ON ERROR)='File')
   2 - access("CTXSYS"."CONTAINS"("JSON_STORAGE"."JSON_DATA",'({1}
              INPATH(/menu/id)) and ({File} INPATH(/menu/value))')>0)
```

Search

DataArt

```
select *  from json_storage j where j.json_data.menu.id = '10'

-------------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |              |     1 |   632 |   239   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| JSON_STORAGE |     1 |   632 |   239   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

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

   1 - filter(JSON_QUERY("J"."JSON_DATA" FORMAT JSON , '$.menu.id'
              RETURNING VARCHAR2(4000) ASIS  WITHOUT ARRAY WRAPPER NULL ON ERROR)='10')
```

. notation is not supported again!

Search

DataArt

```
insert into json_storage(id, created_dtm, json_data) values(1, systimestamp, utl_raw.cast_to_raw('{"values":"RS_485"}'))

select *  from json_storage  where JSON_TEXTCONTAINS(json_data, '$.values', 'RS_485')
```

= 0

```
select * from json_storage  where JSON_TEXTCONTAINS(json_data, '$.values', 'RS') and JSON_TEXTCONTAINS(json_data, '$.values', '485')

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |              |     1 |   632 |    39   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| JSON_STORAGE |     1 |   632 |    39   (0)| 00:00:01 |
|*  2 |   DOMAIN INDEX               | JSON_FTS_IDX |       |       |    39   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

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

   2 - access("CTXSYS"."CONTAINS"("JSON_STORAGE"."JSON_DATA",'(RS INPATH(/values))
              and (485 INPATH(/values))')>0)
```

2 different tokens

= 1

```
select *  from json_storage where JSON_TEXTCONTAINS(json_data, '$.values', '{RS_485}')
```

All Oracle text reserved words - https://docs.oracle.com/database/121/CCREF/cqspcl.htm#CCREF2091

Search

DataArt


```
{
 "class":[
          {
           "class_type":"ownership",
           "values":[{"nm":"id","value":"1"}]
          },
          {
           "class_type":"country",
           "values":[{"nm":"id","value":"640"}]
          },
          ,
          {
           "class_type":"features",
           "values":[{"nm":"id","value":"15"},{"nm":"id","value":"20"}]
          }
         ]
}
```

```
{
 "class":[
          {
           "class_type":"ownership",
           "values":[{"nm":"id","value":"18"}]
          },
          {
           "class_type":"country",
           "values":[{"nm":"id","value":"11"}]
          },
          ,
          {
           "class_type":"features",
           "values":[{"nm":"id","value":"7"},{"nm":"id","value":"640"}]
          }
         ]
}
```

Search

DataArt

```
select *
from json_storage
where JSON_TEXTCONTAINS(json_data, '$.class.values.value', '640')
      and JSON_TEXTCONTAINS(json_data, '$.class.class_type', 'country')

select *
from perso.json_storage
where ctxsys.contains(json_data, '(640 INPATH(/class/values/value)) and (country inpath (/class/class_type))')>0
```

	ID	JS
1	-100	(BLOB)
2	-200	(BLOB)

```
select s.id,
        jt.*
 from json_storage s,
       JSON_TABLE(json_data, '$.class[*]'
          COLUMNS (class_type    VARCHAR2(50 CHAR) PATH '$.class_type',
                   nested path '$.values.value' columns(val     VARCHAR2(50 CHAR) PATH '$')
                  )
                 ) jt
 where s.id in (-100, -200)
       and class_type = 'country'
       and val = '640'
```

ID	CLASS_TYPE	VAL
-100	country	640

```
-----------------------------------------------------------------------------------------------
| Id  | Operation                     | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |              |     1 |   627 |   126   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                 |              |     1 |   627 |   126   (0)| 00:00:01 |
|*  2 |   TABLE ACCESS BY INDEX ROWID| JSON_STORAGE |     1 |   623 |    97   (0)| 00:00:01 |
|*  3 |    DOMAIN INDEX               | JSON_FTS_IDX |       |       |    97   (0)| 00:00:01 |
|*  4 |   JSONTABLE EVALUATION        |              |       |       |            |          |
-----------------------------------------------------------------------------------------------

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

   2 - filter("S"."ID"=(-200) OR "S"."ID"=(-100))
   3 - access("CTXSYS"."CONTAINS"("S"."JSON_DATA",'({country}
              INPATH(/class/class_type)) and ({640} INPATH(/class/values/value))')>0)
   4 - filter("P"."CLASS_TYPE"='country' AND "P"."VAL"='640')
```

Search

DataArt

```
select ID, json_data
from json_storage
where JSON_EXISTS(json_data,'$.class?(@.values.value == $VALUE && @.class_type == $TYPE)' passing '640' as "VALUE", 'features' as "TYPE")
```

filter

Anchor Inside one object

	ID	JS
1	-200	(BLOB)

```
-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |              |     1 |   623 |    97   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| JSON_STORAGE |     1 |   623 |    97   (0)| 00:00:01 |
|*  2 |   DOMAIN INDEX                | JSON_FTS_IDX |       |       |    97   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

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

   1 - filter(JSON_EXISTS2("JSON_DATA" FORMAT JSON , '$.class?(@.values.value ==
              $VALUE && @.class_type == $TYPE)' FALSE ON ERROR)=1)
   2 - access("CTXSYS"."CONTAINS"("JSON_STORAGE"."JSON_DATA",'{640} INPATH
              (/class/values/value) and {features} INPATH (/class/class_type)')>0)
```

Ingestion

DataArt

```
insert into json_storage(id, created_dtm, json_data)
   values(
   -1000,
   systimestamp,
   utl_raw.cast_to_raw('
                         {"menu": {
                        "id": "' || '888888888'  ||'",
                        "value": "File",
                        "popup": {
                          "menuitem": [
                            {"value": "New", "onclick": "CreateNewDoc()"},
                            {"value": "Open", "onclick": "OpenDoc()"},
                            {"value": "Close", "onclick": "CloseDoc()"}
                          ]
                        }
                      }}
                       '
                    )
   );
```

```
select count(*)
from json_storage
where json_textcontains(json_data, '$.menu.id', '888888888')
```

= 0

```
begin
 ctx_ddl.SYNC_INDEX('PERSO.json_fts_idx'); --schema inside :)
end;
```

```
select count(*)
from json_storage
where json_textcontains(json_data, '$.menu.id', '888888888')
```

= 1

Ingestion

DataArt

```
select i.idx_sync_interval, idx_sync_type, i.*
from ctxsys.CTX_INDEXES i
where idx_table = 'JSON_STORAGE'
```

	IDX_SYNC_INTERVAL	IDX_SYNC_TYPE	IDX_ID	IDX_OWNER	IDX_NAME	IDX_TABLE_OWNER	IDX_TABLE
1	(null)	(null)	1441	PERSO	JSON FTS IDX	PERSO	JSON STORAGE

```
begin
 CTX_DDL.REPLACE_INDEX_METADATA('PERSO.json_fts_idx', 'REPLACE METADATA SYNC (ON COMMIT)'); --wtf so many metadata????
end;
```

	IDX_SYNC_INTERVAL	IDX_SYNC_TYPE	IDX_ID	IDX_OWNER	IDX_NAME	IDX_TABLE_OWNER	IDX_TABLE
1	(null)	ON COMMIT	1441	PERSO	JSON FTS IDX	PERSO	JSON STORAGE

```
select count(*)
from json_storage
where json_textcontains(json_data, '$.menu.id', '888888888')
```

= 0

```
commit;
```

```
select count(*)
from json_storage
where json_textcontains(json_data, '$.menu.id', '888888888')
```

= 1

Ingestion

DataArt

```
declare
 lStart number;
begin
 lStart := DBMS_UTILITY.get_time;

 for lCounter in 1..10000 loop
   insert into perso.json_storage
   values(
   lCounter,
   systimestamp,
   utl_raw.cast_to_raw('
                         {"menu": {
                        "id": ' || lCounter ||',
                        "value": "File",
                        "popup": {
                          "menuitem": [
                            {"value": "New", "onclick": "CreateNewDoc()"},
                            {"value": "Open", "onclick": "OpenDoc()"},
                            {"value": "Close", "onclick": "CloseDoc()"}
                          ]
                        }
                      }}
                       '
                    )
   );
   commit;
 end loop;

 DBMS_OUTPUT.put_line('Elapsed: ' ||   (DBMS_UTILITY.get_time - lStart));
 commit;
end;
```

Execution time: ~125 seconds

```
begin ctxsys.drvdml.com_sync_index(:idxname, :idxmem, :partname);
```

Ingestion

DataArt

```
begin
 CTX_DDL.REPLACE_INDEX_METADATA('PERSO.json_fts_idx', 'REPLACE METADATA SYNC (EVERY "SYSDATE+1/24/60")');
end;

select i.idx_sync_interval, idx_sync_type, idx_sync_jobname, i.*
from ctxsys.CTX_INDEXES i
where idx_table = 'JSON_STORAGE'
```

	IDX_SYNC_INTERVAL	IDX_SYNC_TYPE	IDX_SYNC_JOBNAME	IDX_ID	IDX_OWNER	IDX_NAME	IDX_TABLE_OWNER	IDX_TABLE
1	SYSDATE+1/24/60	AUTOMATIC	DR$JSON FTS IDX$J	1441	PERSO	JSON FTS IDX	PERSO	JSON STORAGE

```
select job_name, job_type, job_action, repeat_interval from dba_scheduler_jobs where owner = 'PERSO'
```

JOB_NAME	JOB_TYPE	JOB_ACTION	REPEAT_INTERVAL
DR$JSON FTS IDX$J	PLSQL BLOCK	ctxsys.drvdml.auto sync index('JSON FTS IDX', 67108864, NULL, NULL, NULL, 0);	SYSDATE+1/24/60

```
declare
 lStart number;
begin
 lStart := DBMS_UTILITY.get_time;

 for lCounter in 1..10000 loop
   insert into perso.json_storage
   values(
   lCounter,
   systimestamp,
   utl_raw.cast_to_raw('
                        {"menu": {
                       "id": ' || lCounter ||',
                       "value": "File",
                       "popup": {
                         "menuitem": [
                           {"value": "New", "onclick": "CreateNewDoc()"},
                           {"value": "Open", "onclick": "OpenDoc()"},
                           {"value": "Close", "onclick": "CloseDoc()"}
                         ]
                       }
                     }}
                      '
                    )
   );
   commit;
 end loop;

 DBMS_OUTPUT.put_line('Elapsed: ' ||  (DBMS_UTILITY.get_time - lStart));
 commit;
end;
```

Execution time: ~7 seconds

Refresh job execution time: ~4 seconds

Ingestion

DataArt

```
drop INDEX json_fts_idx force;

CREATE drop INDEX json_fts_idx force  ON json_storage (json_data)  INDEXTYPE IS CTXSYS.CONTEXT
PARAMETERS ('section group CTXSYS.JSON_SECTION_GROUP SYNC (EVERY "SYSDATE+1/24") TRANSACTIONAL') ;

declare
 lStart number;
begin
 lStart := DBMS_UTILITY.get_time;

 for lCounter in 1..10000 loop
   insert into perso.json_storage
   values(
   lCounter,
   systimestamp,
   utl_raw.cast_to_raw('
                         {"menu": {
                        "id": ' || lCounter ||',
                        "value": "File",
                        "popup": {
                          "menuitem": [
                            {"value": "New", "onclick": "CreateNewDoc()"},
                            {"value": "Open", "onclick": "OpenDoc()"},
                            {"value": "Close", "onclick": "CloseDoc()"}
                          ]
                        }
                      }}
                       '
                    )
   );
   --commit;
 end loop;

 DBMS_OUTPUT.put_line('Elapsed: ' ||  (DBMS_UTILITY.get_time - lStart));

end;
```

Execution time: ~8 seconds

Ingestion

DataArt

```
select count(*)  from json_storage  where json_textcontains(json_data, '$.menu.id', '1') ;
```
= 0

```
select count(*)  from json_storage   where contains(json_data, '1 INPATH(/menu/id)') >0;
```
= 0

```
commit;
```

```
select count(*)  from json_storage  where json_textcontains(json_data, '$.menu.id', '1') ;
```
= 0

```
select count(*)  from json_storage   where contains(json_data, '1 INPATH(/menu/id)') >0;
```
= 0


```
select count(*) from json_storage where contains(json_data, '1') >0;
```
= 1

TRANSACTIONAL doesn't work for sections! Only for tokens!

Ingestion

DataArt

```
begin
 ctx_ddl.create_preference('PERSO_STORE', 'BASIC_STORAGE');
 ctx_ddl.set_attribute('PERSO_STORE', 'STAGE_ITAB', 'YES');
end;
```

```
create index  json_fts_idx on json_storage (json_data)
indextype is ctxsys.context PARAMETERS ('section group CTXSYS.JSON_SECTION_GROUP storage PERSO_STORE sync (on commit)')
```

```
begin
 ctx_ddl.add_auto_optimize('json_fts_idx'); --to move data async
end;
```

```
select owner, job_name, program_name,last_start_date,state from dba_scheduler_jobs where owner='CTXSYS';
```

OWNER	JOB_NAME	PROGRAM_NAME	LAST_START_DATE	STATE
CTXSYS	DR$BGOPTJOB	DR$BGOPTPRG	25-JUN-17 11.04.15.128537000 AM AMERICA/NEW YORK	SUCCEEDED

```
select table_name from dba_tables where table_name like '%$G%' and owner = 'PERSO'
```

	TABLE_NAME
1	DR$JSON_FTS_IDX$G

Ingestion

DataArt

```
declare
 lStart number;
begin
 lStart := DBMS_UTILITY.get_time;

 for lCounter in 1..10000 loop
  insert into perso.json_storage
  values(
  lCounter,
  systimestamp,
  utl_raw.cast_to_raw('
                        {"menu": {
                       "id": ' || lCounter ||',
                       "value": "File",
                       "popup": {
                         "menuitem": [
                           {"value": "New", "onclick": "CreateNewDoc()"},
                           {"value": "Open", "onclick": "OpenDoc()"},
                           {"value": "Close", "onclick": "CloseDoc()"}
                         ]
                       }
                     }}
                      '
                   )
  );
  --commit;
 end loop;

 DBMS_OUTPUT.put_line('Elapsed: ' ||  (DBMS_UTILITY.get_time - lStart));
 commit;
end;
```

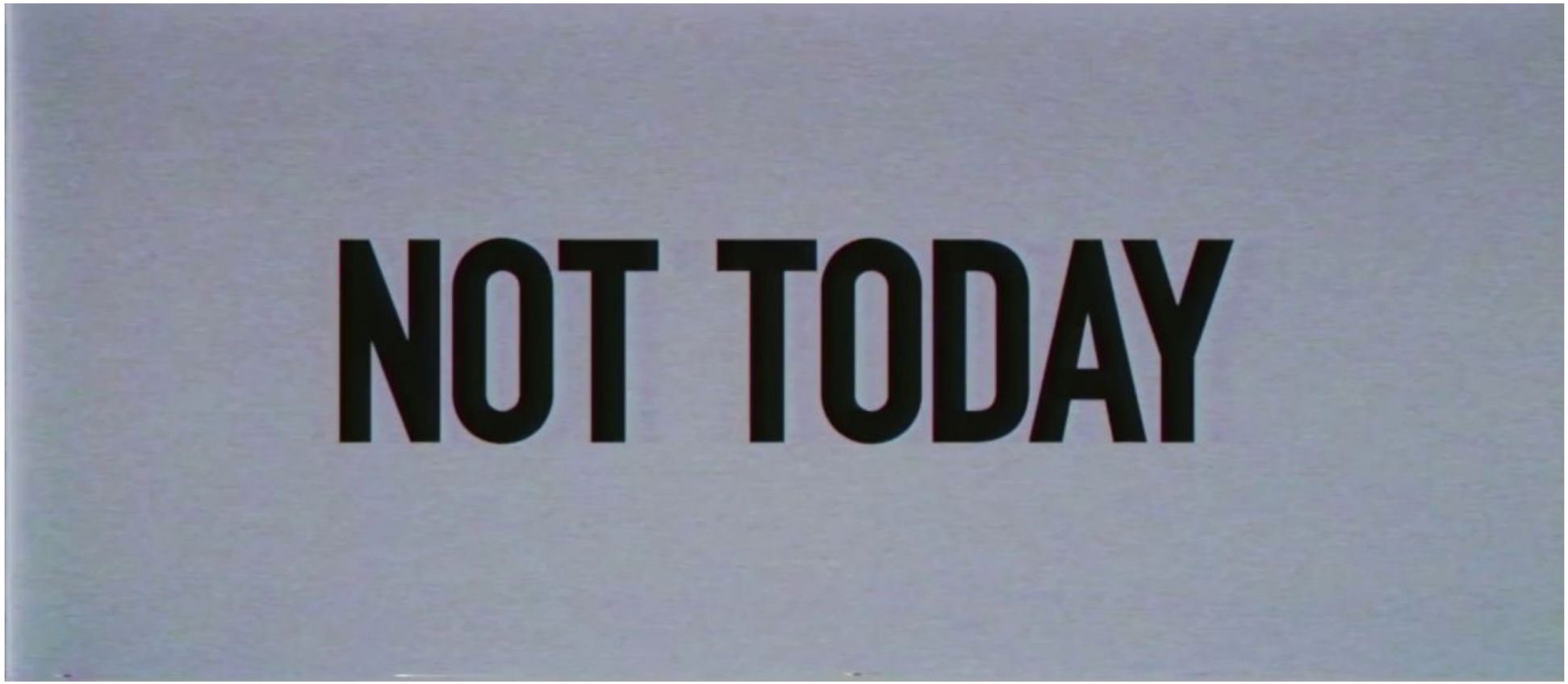
Execution time: ~6 seconds

```
select job_name, log_date, run_duration
from dba_scheduler_job_run_details r
where owner='CTXSYS'
order by log_date desc;
```

	JOB_NAME	LOG_DATE	RUN_DURATION
1	DR$BGOPTJOB	26-JUN-17 08.45.39.069395000 AM -04:00	+00 00:00:01.000000
2	DR$BGOPTJOB	26-JUN-17 08.43.19.174367000 AM -04:00	+00 00:00:00.000000

Fast search

DataArt



Follow DataArt ITTalks on https://dataart.ru/events

Maintenance

DataArt

Pos/prefix columns with JSON data via _JSON like *INVOICE_JSON* before.

- Create daily checks
  1. If you need control JSON format (strict/lax) use *dba_tab_columns* and *all_json_columns* views to check JSON constrains
  2. If you need insert performance check *dba_lobs* to check *cache* attribute
- Check CONTEXT indexes are in proper state

Maintenance

DataArt

Provide regular indexes optimization

1. Collect fragmented indexes (*estimated row fragmentation*)
2. Collect indexes with many deleted rows (*estimated garbage size*)
3. Run *ctx_ddl.optimize_index* in FULL mode (SERIAL or PARALLEL)

```
declare
 lXML clob;
 lOffset number := 0;
begin
   ctx_report.index_stats('json_fts_idx', lXML, frag_stats => true, report_format => 'XML');
end;
```

```
<STAT_FRAG_STATS>
    <STAT_STATISTIC NAME="total size of $I data">3,810,329 (3.63 MB)</STAT_STATISTIC>
    <STAT STATISTIC NAME="$I rows">31,069</STAT STATISTIC>
    <STAT_STATISTIC NAME="estimated row fragmentation">65 %</STAT_STATISTIC>
    <STAT_STATISTIC NAME="garbage docids">0</STAT_STATISTIC>
    <STAT_STATISTIC NAME="estimated garbage size">0</STAT_STATISTIC>
```

```
begin
 ctx_ddl.optimize_index('json_fts_idx', CTX_DDL.OPTLEVEL_FULL);
end;
```

Conclusion

DataArt

- JSON is always tradeoff between performance/data treatment convenience/integrity
- Indexing strategy should be checked very careful you use 2 notations especially
- JSON treatment is acceptable in row-per-row scenario
- JSON features are still non-stable
- Oracle fails with JSON more 2 Mb very often
- Current implementation doesn’t look like “document-stored” DB
- Tailored search solutions bring better performance

and we are waiting Oracle 12.2 ☺

Q&A

DataArt

THANK YOU.
WE ARE HIRING!

Alexander Tokarev
Senior Developer
DataArt
Alexander.Tokarev@dataart.com


DataArt