Test of Oracle JSON support in the view of CMS JSON data

Sartaj Singh Baveja

Supervisors : Katarzyna Maria Dziedziniewicz-Wojcik (CERN IT)
Valentin Kuznetsov (CMS)

Who am I?

- Alumnus of NSIT 

- Will join Lawrence Berkeley Labs as a Research scholar 

- Previously interned at MIT Media Lab, ESnet 

Contents

Importance of JSON

- Lightweight data-interchange format
- Used in web services responses
- Easy to parse which makes it easier to use

```
json = {
    "firstName": "John",
    "lastName": "Smith",
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        "212 555-1234",
        "646 555-4567"
    ]
}
```

CMS JSON data

- 200k-300k documents are generated everyday
- Size of each document : ~12kB i.e. 3GB/day or 1-2TB/year
- Each document
  - JSON data format
  - Deep Nested Structure
  - Represents CMS Framework Job Report

```
# CMS FWJR data structure
{"meta_data": {"agent_ver": "1.0.14.pre5",
               "fwjr_id": "1-0",
               "host": "a.b.com",
               "jobtype": "Processing",
               "jobstate": "success",
               "ts": 1456500229},
 "LFNArray": ["/store/file1.root",
              "/store/file2.root",
              "/lfn/fallbackfile.root", "/lfn/skipedfile.root"],
 "LFNArrayRef": ["fallbackFiles",
                 "outputLFNs",
                 "lfn",
                 "skippedFiles",
                 "inputLFNs"],
 "PFNArray": ["root://file1.root",
              "root://file2.root",
              ],
 "PFNArrayRef": ["inputPFNs", "outputPFNs", "pfn"],
 "steps": [{"name": "cmsRun1",
            "analysis": {},
            "cleanup": {},
            "logs": {},
            "errors": [
                  {
                      "details": "An exception",
                      "type": "Fatal Exception",
                      "exitCode": 8001
                  }
              ],
            "input": [{"catalog": "",
                       "events": 6893,
```

Current Solution | MongoDB

- Used by CMS as a buffer before putting docs to HDFS.
- Open-source NoSQL database
- Provides high performance, high availability, and automatic scaling
- Schema Less
- Dynamic Queries
- Support for full indexes, including inner objects

JSON in Oracle Database

- Recently introduced JSON support with relational database features such as
  - Transactions
  - Indexing
  - Declarative Querying (join JSON data with relational data, project data relationally)
  - Views
- Ensures data integrity
- ***is_json*** check constraints to ensure valid JSON instances

ORACLE

JSON in Oracle Database (contd.)

- Stored using SQL Data Types VARCHAR2, CLOB (**C**haracter **L**arge **Ob**ject) and BLOB (**B**inary **L**arge **Ob**ject)

- Dot notation to access fields of the embedded document

```
SELECT po.po_document.Requestor FROM j_purchaseorder po;
```

Oracle Release 12.1 vs 12.2

- Simplified dot notation doesn't work on arrays in 12.1 (fixed in 12.2)

```
SELECT test.doc.LFNArray[0] from test11 test;
```

```
ORA-00923: FROM keyword not found where expected
00923. 00000 - "FROM keyword not found where expected"
*Cause:
*Action:
```

- ORA 600 : No data to be read from socket appears in 12.1

- "Missing right parenthesis" error occurred. Queries such as "**SELECT * FROM table**" were not working. Due to this, the entire table had to be dropped

```
SELECT test.doc.LFNArray[0] from test8 test;
```

	LFNARRAY
1	/store/mc/Run660/file0.root
2	/store/mc/Run661/file0.root
3	/store/mc/Run662/file0.root
4	/store/mc/Run663/file0.root
5	/store/mc/Run664/file0.root

OBSERVATIONS

Machine Configuration :

OS MacOSX
Processor 2.6GHz, Intel Core i5
Dual Core
Memory 8GB RAM
Storage 256GB

Generating randomized JSON Documents

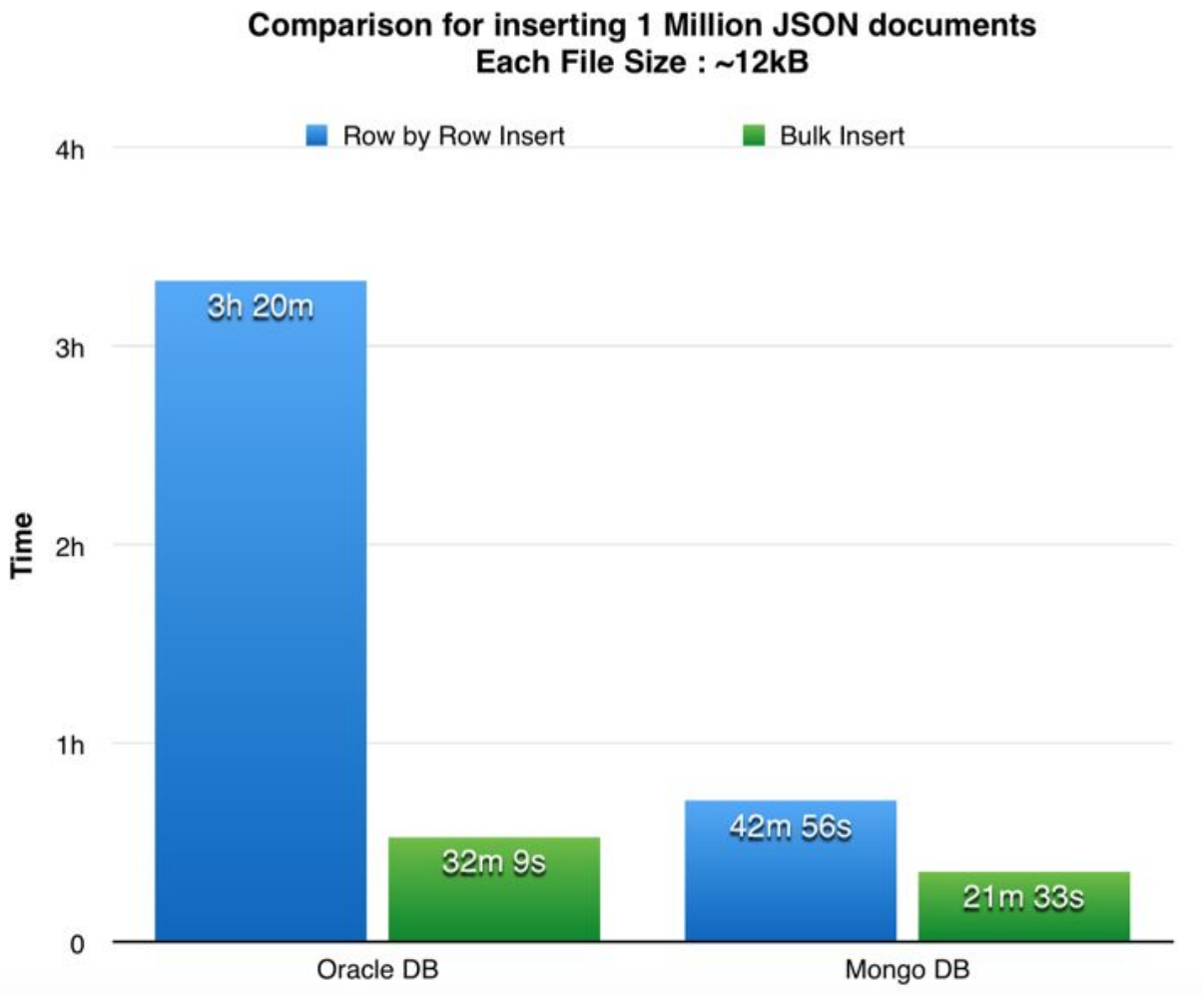
- Certain fields randomized according to the CMS Framework Job Report format

- Focus on functional tests

```
50  def generateJSON(doc, index, idx):
51      x = 250000
52
53      newdoc = copy.deepcopy(doc)
54      newdoc['wmaid'] = get_random_string(32)
55
56      for i in range(len(newdoc['steps'])):
57          storage_key = newdoc['steps'][i]['performance']['storage']
58
59          storage_key['writeTotalMB'] = round(random.uniform(200,400), 2)
60          storage_key["readPercentageOps"] = random.uniform(1, 2)
61          storage_key["readMBSec"] = random.uniform(0,1)
62
63          steps_key = newdoc['steps'][i]
64
65          steps_key['site'] = 'T' + str(random.randint(1,5)) + '_US_FNAL_Disk'
66          output_length = len(steps_key['output'])
```

Row Inserts vs Bulk Inserts

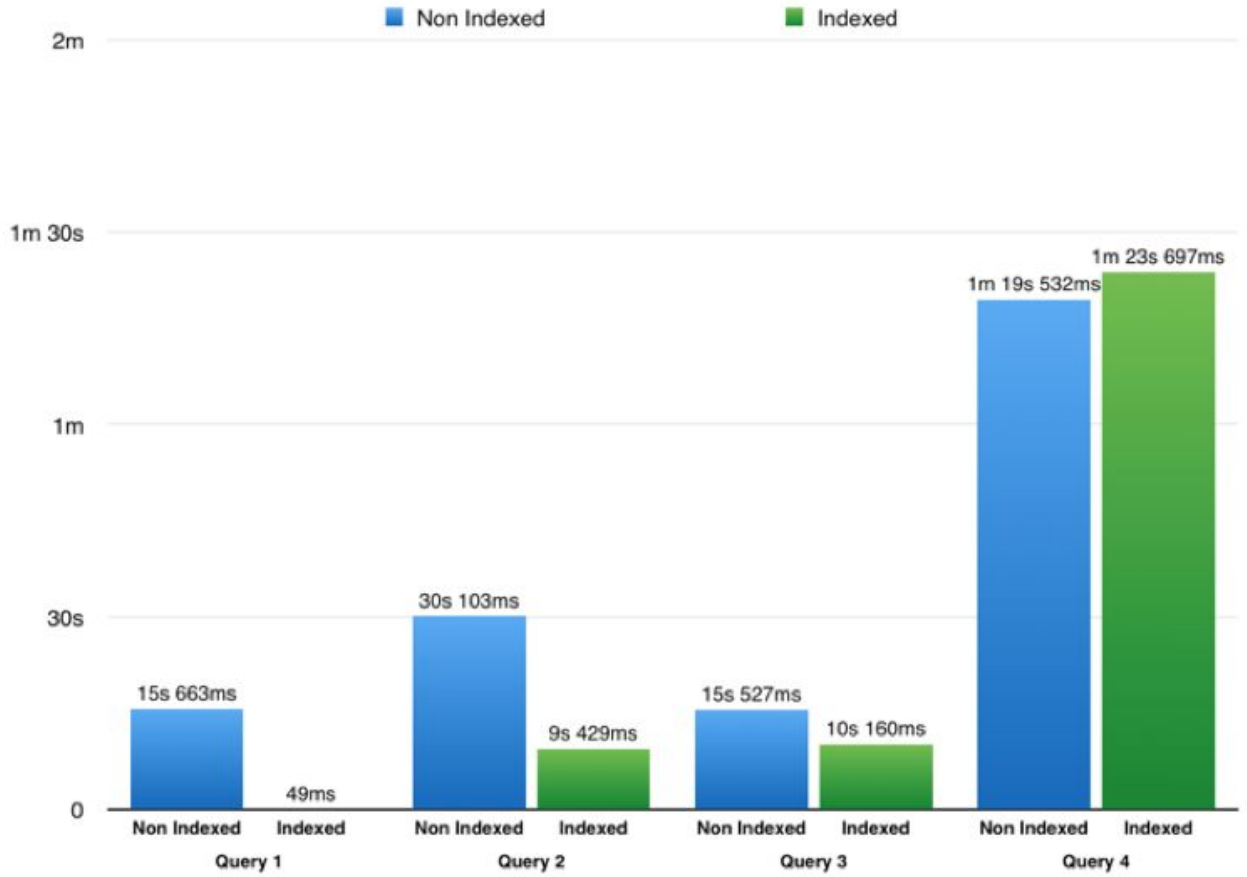
Injection Rate



Indexed vs Non Indexed Queries

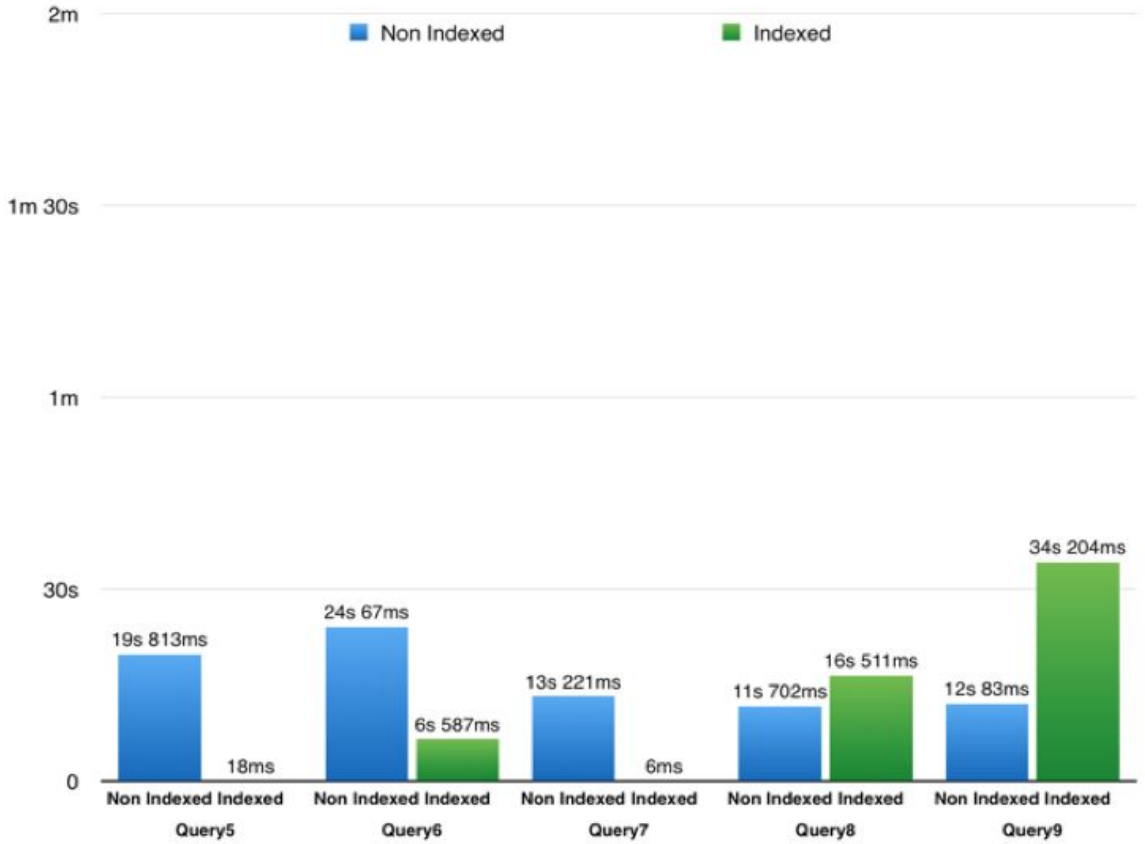
MongoDB Indexed vs Non Indexed Queries

Query 1,2,3 : Find specific string from document
Query 4 : Aggregate data



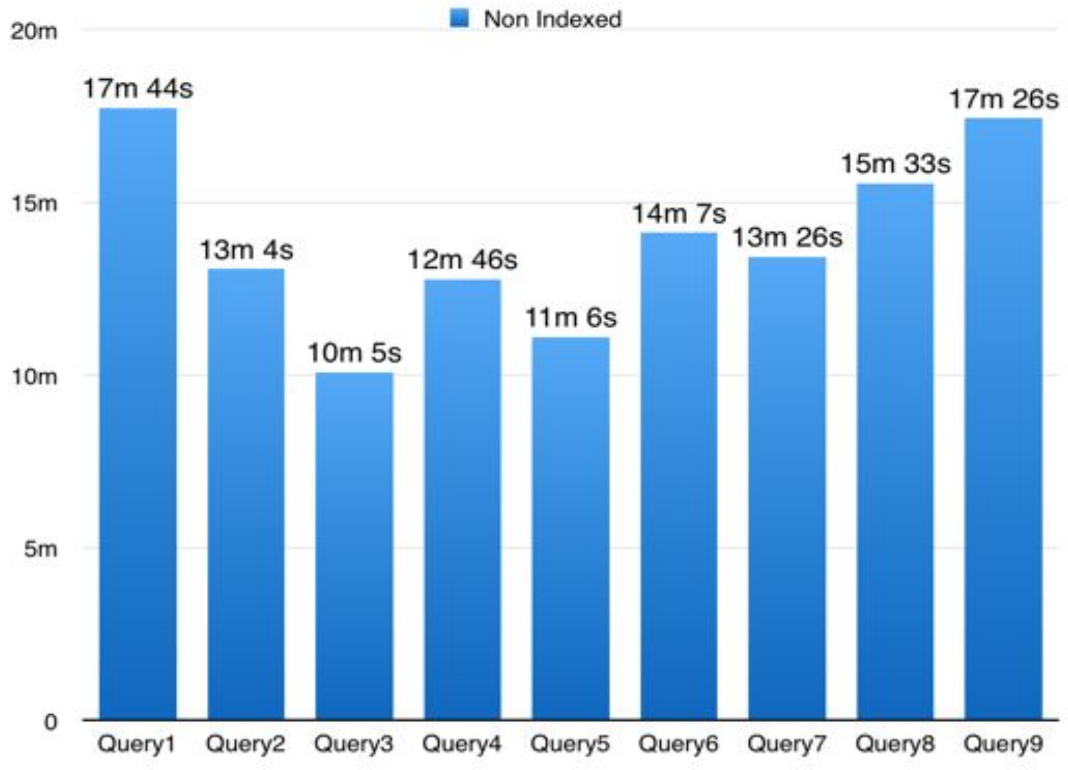
MongoDB Indexed vs Non Indexed Queries

Query 5: Find a string
Query 6/7: Find records based on provided pattern
Query 8,9: Logical / Comparison Query Operators



Oracle Non Indexed Queries

Query 1/2/3: Search for specific string
Query 4: Aggregate data
Query 5: Find a string
Query 6/7: Find records based on pattern
Query 8,9: Logical / Comparison Operators



Oracle Indexed Queries

```
create index ind_lfn on test11 json_query(doc, '$.LFNArray');

explain plan for SELECT M.*
  FROM test11 p,
       json_table(
        p.doc,
        '$'
        columns (
          wmaid varchar2(2000 char) path '$.wmaid',
          meta_data varchar2(2000 char) format json with wrapper path '$.meta_data',
          nested path '$.LFNArray[*]'
          columns (
            lfn varchar2(2000 char) path '$'
          )
        )
      ) M
WHERE lfn = '/store/mc/Run3212/file0.root';
```

```
select * from table(dbms_xplan.display);
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 15 in 0.02 seconds

```
PLAN_TABLE_OUTPUT
Plan hash value: 2620315792

-------------------------------------------------------------------------------------------
| Id  | Operation                | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |         |    81M|  7244M|    27M  (2)| 00:17:54 |
|   1 |  NESTED LOOPS            |         |    81M|  7244M|    27M  (2)| 00:17:54 |
|   2 |   TABLE ACCESS FULL      | TEST11  |  1000K|    82M|  1688   (1)| 00:00:01 |
|*  3 |   JSONTABLE EVALUATION   |         |       |       |            |          |
-------------------------------------------------------------------------------------------

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

   3 - filter("P"."LFN"='/store/mc/Run3212/file0.root')
```

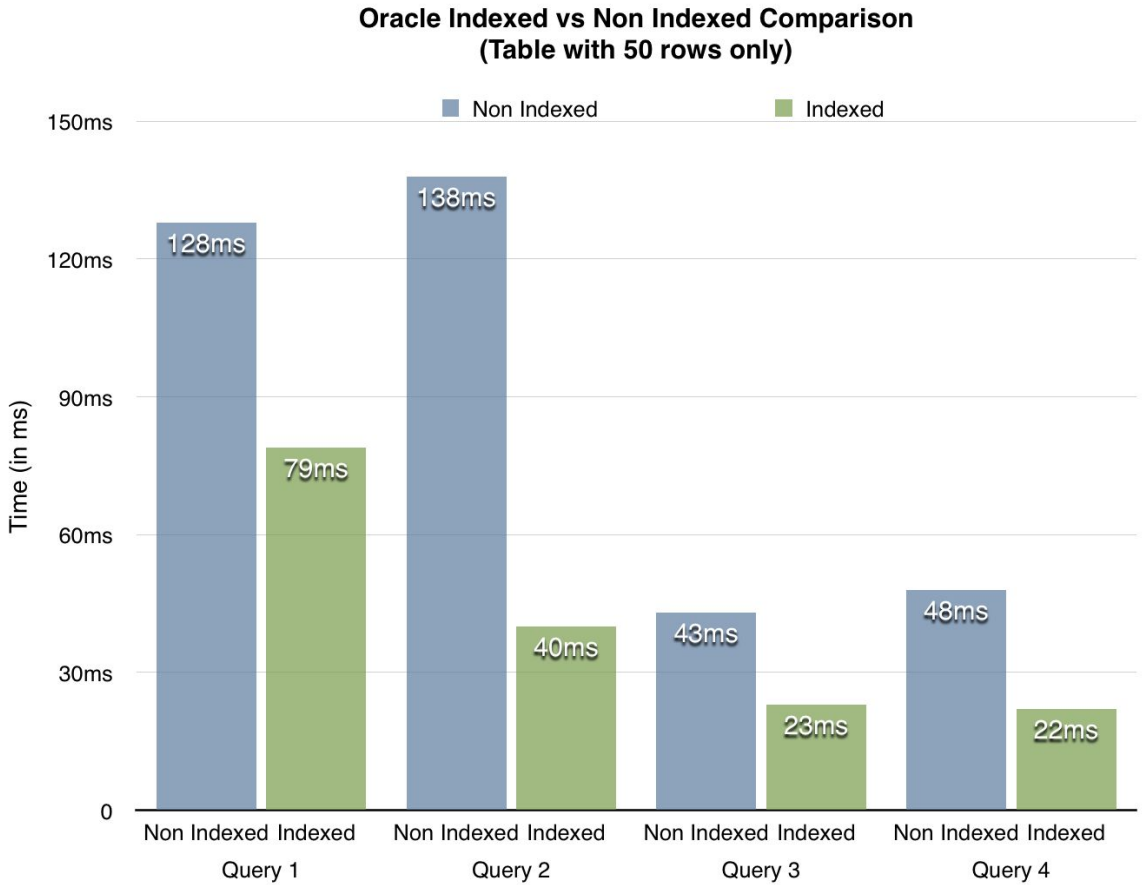
Oracle Indexed Queries

```
alter session set events '10053 trace name context forever, level 1';

SELECT VALUE FROM V$DIAG_INFO WHERE NAME = 'Default Trace File';

exit;
```

```
explain plan for SELECT /*+ index(test11 ind_lfn) */ M.*
  FROM test11 p,
       json_table(
        p.doc,
        '$'
        columns (
          wmaid varchar2(2000 char) path '$.wmaid',
          meta_data varchar2(2000 char) format json with wrapper path '$.meta_data',
          nested path '$.LFNArray[*]'
          columns (
            lfn varchar2(2000 char) path '$'
          )
        )
      ) M
WHERE lfn = '/store/mc/Run3212/file0.root';
```

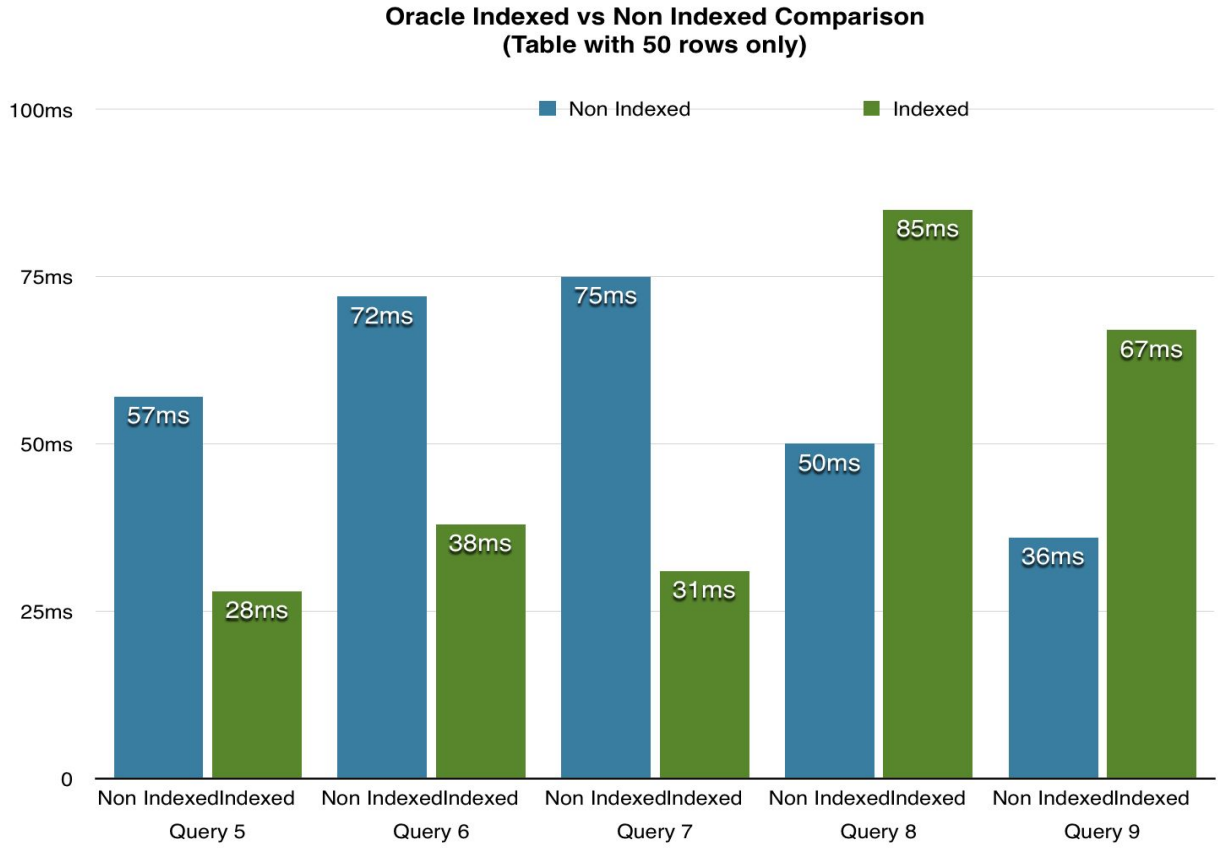
Oracle Indexed vs Non Indexed Queries (Smaller Table)

Query 1,2,3 : Find specific string from document
Query 4 : Aggregate data



Oracle Indexed vs Non Indexed Queries (Smaller Table)

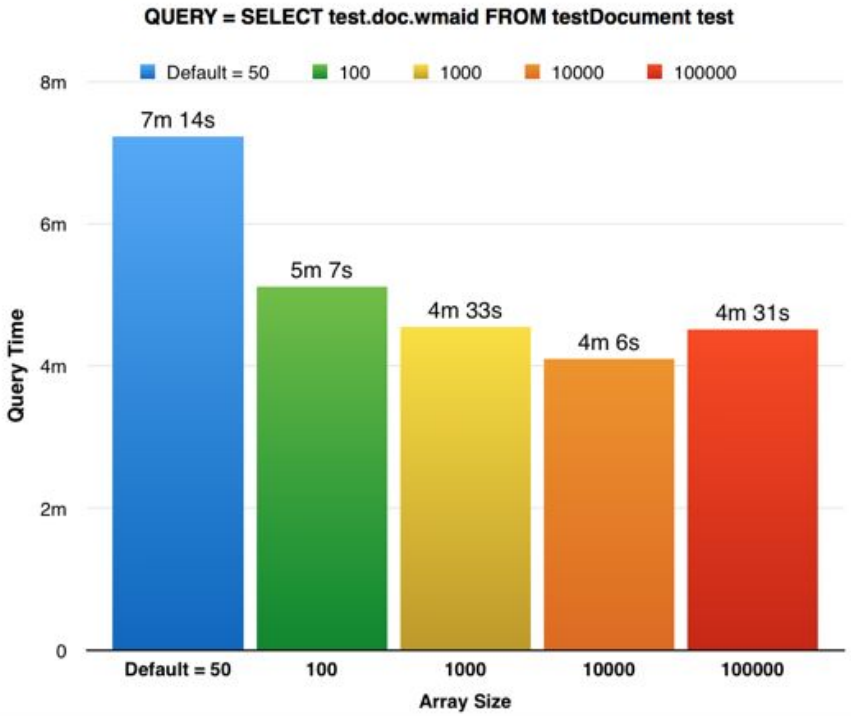
Query 5: Find a string
Query 6/7: Find records based on provided pattern
Query 8,9: Logical / Comparison Query Operators



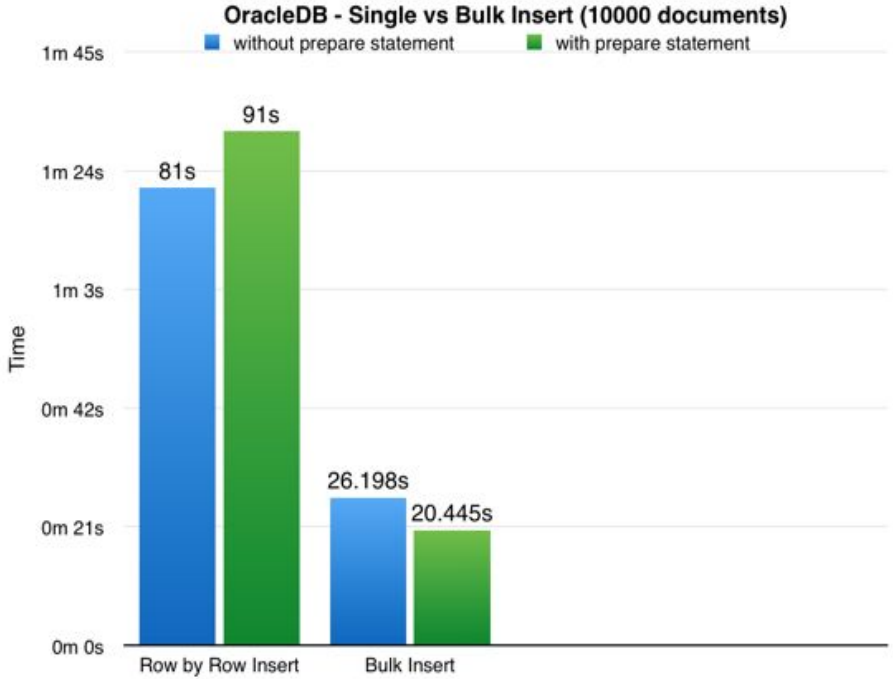
Oracle : Performance Improvements

Performance Improvements in OracleDB

1. Cursor.arraysize

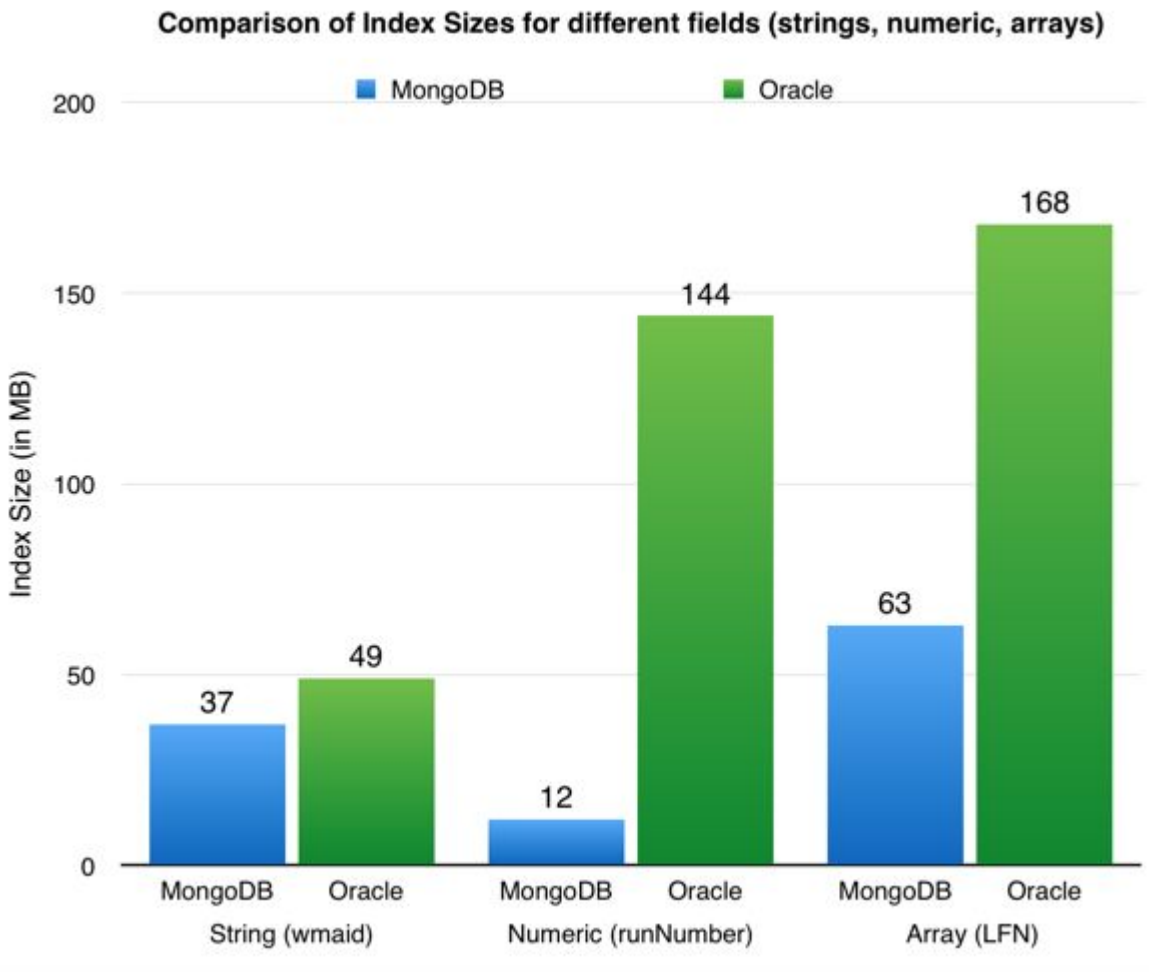


2. Bind Variables



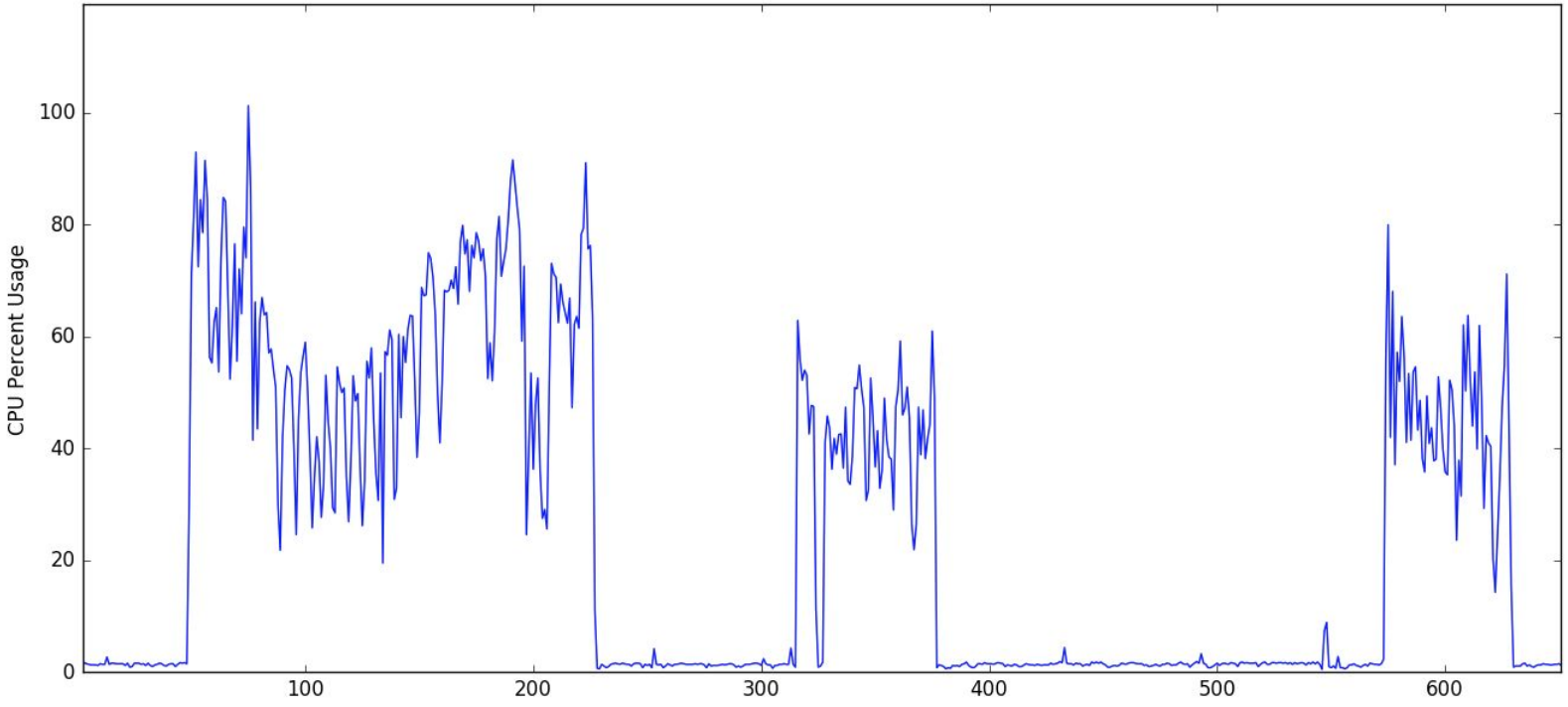
Index Size

Comparison of Index Sizes

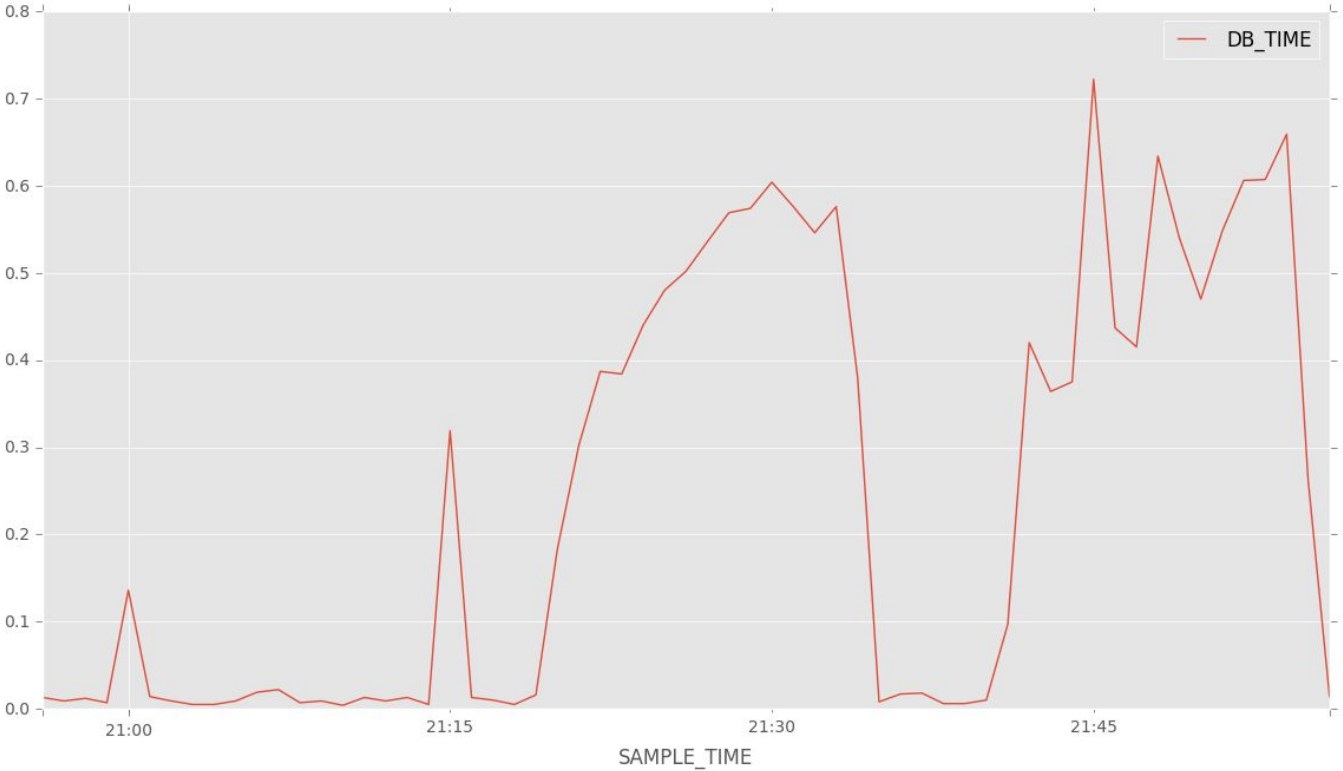


CPU Usage Metrics

MongoDB CPU Usage



Oracle CPU Usage



COMPARISON

MongoDB	Oracle
Higher Data Size 1.3 GB for 1M documents	**Lower** Data Size ~50 MB for 1M documents
223 MB Index Size	**1.14 GB** Index Size
Queries are **not** atomic	Queries **are** atomic
Lower Insert Time	**Higher** Insert Time
Indexing **works** perfectly	Indexing **not straightforward**
No such errors	**Internal errors** such as “No Data to be read from socket” occur frequently

Conclusion

- Both the databases support JSON completely
- Both databases were successful in executing all the given queries
- Oracle’s JSON library looks promising
  - Has a few errors which will improve as further versions are released
  - Can perform read/writes and queries

Future Work

- Compare the performance with other databases
  - Postgres
  - MySQL

- Indexing with Oracle JSON Library

Thank You!

sartajcorp@gmail.com

www.sartajsingh.in

sartaj10