Oracle Converged Database 23c

Developing document-oriented applications with JSON, REST, SODA and MongoDB API

I. Purpose of this demonstration

This demonstration has been prepared to present functionality of Oracle MongoDB API with Oracle 23c, including support for MongoDB tools (mongoimport, mongosh, mongodump), basig MongoDB operations, JSON Relational Duality Views and SODA

II. Initial requirements

1. Oracle Database 23c
2. A database schema with the following privileges and roles

   CREATE JOB, CREATE PROCEDURE, CREATE SEQUENCE, CREATE VIEW, CREATE TABLE,

   UNLIMITED TABLESPACE, CREATE SESSION, SODA_APP

3. ORDS v. 23.4 or later using above schema
4. SQL Cli v.23.4 or later
5. The following MongoDB client packages
   a. mongodb-database-tools-100.9.4-1.x86_64.rpm
   b. mongodb-mongosh-2.1.1-1.el8.x86_64.rpm
   c. mongocli-1.31.0-1.x86_64.rpm

III. Scenario

1. Using sql connect to the database and run hr_schema.sql script to create DEPARTMENTS table

```
sql <user>/<password>@<machine>/<service>
SQL>@hr_schema.sql
```

2. Using mongoimport tool connect to the Oracle Database and import zips.json file

```
mongoimport -v --tlsInsecure --uri
'mongodb://<user:password>@<machine>:27017/<user>?authMechanism=PLAIN\
&authSource=$external&tls=true&retryWrites=false&loadBalanced=true' --file
/home/opc/mongodemo/zips.json -c zips --drop
```

3. Connect to the database using mongosh tool

```
mongosh  --tlsAllowInvalidCertificates \
'mongodb://<username>:<password>@<machine>:27017/<user>?authMechanism=PLAIN\
&authSource=$external&tls=true&retryWrites=false&loadBalanced=true'
```

4. Display all the collections

```
user> show collections
zips
user>
```

5. Calculate the number of JSON documents in the zips collection

```
user> db.zips.countDocuments()
29353
user>
```

6. Find the first document in the zips collection

```
user> db.zips.findOne()
{
  _id: '01001',
  city: 'AGAWAM',
  loc: [ -72.622739, 42.070206 ],
  pop: 15338,
  state: 'MA'
}
user>
```

7. Display first 5 documents orderd by CITY field

```
user> db.zips.find().limit(5).sort({city: 1})
[
  {
    _id: '42601',
    city: 'AARON',
    loc: [ -85.199114, 36.812827 ],
    pop: 270,
    state: 'KY'
  },
...
  {
    _id: '31001',
    city: 'ABBEVILLE',
    loc: [ -83.306845, 31.96484 ],
    pop: 1991,
    state: 'GA'
  }
]
```

8. Find all JSON documents with CITY field equal to "CHICOPEE"

```
user> db.zips.find({"city": "CHICOPEE"})
[
  {
    _id: '01013',
    city: 'CHICOPEE',
    loc: [ -72.607962, 42.162046 ],
    pop: 23396,
    state: 'MA'
  },
  {
    _id: '01020',
    city: 'CHICOPEE',
    loc: [ -72.576142, 42.176443 ],
    pop: 31495,
    state: 'MA'
  }
]
user>
```

9. Calculate number of all JSON documents with CITY field set to "CHICOPEE"

```
user> db.zips.find({"city": "CHICOPEE"}).count()
2
user>
```

10. Find all locations with population greater than 100 000

```
user> db.zips.find({pop: {$gt:100000}})
[
  {
    _id: '10021',
    city: 'NEW YORK',
    loc: [ -73.958805, 40.768476 ],
    pop: 106564,
    state: 'NY'
  },
...
  {
    _id: '60623',
    city: 'CHICAGO',
    loc: [ -87.7157, 41.849015 ],
    pop: 112047,
    state: 'IL'
  }
]
user>
```

11. Calculate number of citizens for every city. Please, note, that a single one city can contain different locations with different zip code. Since then to solve this problem there is need to aggregate the data by CITY field (use MongoDB Aggregation Pipeline)

```
user> db.zips.aggregate(
...   [{"$group" : {"_id"    : "$city",
...                 "sumPop" : {"$sum" : "$pop"}}},
...    {"$sort"  : {"sumPop" : -1}}])
[
  { _id: 'CHICAGO', sumPop: 2452177 },
  { _id: 'BROOKLYN', sumPop: 2341387 },
  { _id: 'HOUSTON', sumPop: 2123053 },
  { _id: 'LOS ANGELES', sumPop: 2102295 },
...
  { _id: 'SAN FRANCISCO', sumPop: 723993 },
  { _id: 'CLEVELAND', sumPop: 687451 }
]
Type "it" for more
user>
```

12. Connect to the database using SQL cli

```
$ sql <user>/<password>@<machine>/<database_service>
```

13. Display structure of DEPARTMENTS table

```
SQL> desc DEPARTMENTS

Name                 Null?        Type
__________________ ___________ _______________
DEPARTMENT_ID        NOT NULL     NUMBER(4)
DEPARTMENT_NAME      NOT NULL     VARCHAR2(30)
MANAGER_ID                        NUMBER(6)
LOCATION_ID                       NUMBER(4)
SQL>
```

14. Create a JSON Relational Duality View on top of DEPARTMENTS table

```
SQL> CREATE OR REPLACE JSON RELATIONAL DUALITY VIEW DEPT_JSON
     as select json{
     'DepartmentID': department_id,
     'DepartmentName': department_name,
     'LocationId': location_id,
     'ManagerId': manager_id }
     from departments;

View DEPT_JSON created.

SQL>
```

15. Using SODA PL/SQL API create DEPT_COLLECTION JSON collection on top of the view created in previous step

```
SQL> declare
       l_collection  soda_collection_t;
     begin
       l_collection := dbms_soda.create_dualv_collection(
                         collection_name => 'DEPT_COLLECTION',
                         view_name       => 'DEPT_JSON');

       if l_collection is not null then
         dbms_output.put_line('Collection ID : ' || l_collection.get_name());
       else
         dbms_output.put_line('Collection does not exist.');
       end if;
     end;
     /

PL/SQL procedure successfully completed.

SQL>
```

16. Using SODA display all existing collections in your schema

```
SQL> soda list
List of collections:

        DEPT_COLLECTION
        zips

SQL>
```

17. Using SODA calculate number of JSON documents in DEPT_COLLECTION collection and compare it to the number of rows in DEPARTMENTS table calculated using SQL

```
SQL> soda count DEPT_COLLECTION

 27 rows selected.

SQL> select count(*) from DEPARTMENTS;
SQL> select count(*) from DEPARTMENTS;

   COUNT(*)
___________
         27

SQL>
```

18. Using SQL delete department with ID equal to 270 and once again check number of JSON documents in DEPT_COLLECTION. Commit the transaction.

```
SQL> delete
     from departments
     where department_id = 270;

1 row deleted.

SQL> soda count DEPT_COLLECTION

 26 rows selected.

SQL>
```

19. Using SODA interface find in zips collection all locations with CITY set to “CHICOPEE”

```
SQL> soda get zips -f {"city":"CHICOPEE"}

Key:             043031303133
Content:         {"_id":"01013","city":"CHICOPEE","loc":[-
72.607962,42.162046],"pop":23396,"state":"MA"}
-----------------------------------------

Key:             043031303230
Content:         {"_id":"01020","city":"CHICOPEE","loc":[-
72.576142,42.176443],"pop":31495,"state":"MA"}
-----------------------------------------

 2 rows selected.

SQL>
```

20. Connect to the database using mongosh and check available collections

```
mongosh  --tlsAllowInvalidCertificates \
'mongodb://<username>:<password>@<machine>:27017/<user>?authMechanism=PLAIN\
&authSource=$external&tls=true&retryWrites=false&loadBalanced=true'

user> show collections
DEPT_COLLECTION
zips
user>
```

21. Calculate number of JSON documents in DEPT_COLLECTION collection and display the first one

```
user> db.DEPT_COLLECTION.countDocuments()
26
user> db.DEPT_COLLECTION.findOne()
{
  _id: 'FB03C10B00',
  DepartmentID: 10,
  DepartmentName: 'Administration',
  LocationId: 1700,
  ManagerId: 200,
  _metadata: {
    etag: Binary.createFromBase64('Ss3nrX3EokpfB4mw4u6TGA==', 0),
    asof: Binary.createFromBase64('AAAAAAc96pE=', 0)
  }
}
```

Please, note, that the number of JSON documents in this collection is 26, and it reflects the results of DELETE command executed in step #18.

22. To complete the lab, disconnect from the database

```
User> exit
$
```