Oracle Technology Network
Developer Day

ORACLE
TECHNOLOGY NETWORK

OTN Developer Day: Oracle Big Data

Hands On Lab Manual

Oracle NoSQL Database: Accessing NoSQL Data from Oracle Database

ORACLE NOSQL DATABASE HANDS-ON WORKSHOP

External Tables

Lab Exercise 1 – Start Oracle NoSQL Database instance and access data from Formatter classes

In this exercise, you will start an Oracle NoSQL Database instance that has movie data preloaded. KVLite will be used as the Oracle NoSQL Database Instance. A very brief introduction to KVLite follows:

Single Node Installation – Introducing KVLite

- Run Oracle NoSQL Database on a single machine.
- Develop and test access to Oracle NoSQL Database.
- Easy to get started.
- KVLite is not intended for production.
- Runs in a single process.

Data that is preloaded into Oracle NoSQL Database has information about customers, movies, a genre, user watch history, recommendation list and much more. All this information is saved as a key and a value (mostly a JSON object). Keys for all the different entities (movies, genre, customer etc) are prefixed with a unique string as the first component of the major key.

These unique prefixes are going to be used while defining external tables from Oracle Database. For now let's look at to three key structures that we are going to deal with in this lab.

- Movie: The key structure used to store movies is like this:

Major-Key[1]	Major-Key[2] (MovieID)	Value
MV	829	{"id":829,"original_title":"Chinatown","release_date":"1974","overview":"JJ 'Jake' Gittes is a private detective who seems to specialize in matrimonial cases...", "vote_count":110050,"popularity":8.4,"poster_path":"/6ybT8RbSbd4AltIDABuv39dgqMU.jpg","runtime":0,**"genres":[{"id":"8","name":"Crime"},{"id":"3","name":"Drama"},{"id":"20","name":"Mystery"},{"id":"9","name":"Thriller"}]**}

- Genre: Each movie belongs to one or more genres. This key-value pair maintains the list of all available genres available for this application: Here's how the KV structure looks like:

Major-Key[1]	Major-Key[2] (MovieID)	Value
GN	3	`Drama`
GN	4	`Comedy`

- Genre_Movies: To search movies by genreID, an association between genreID and movieID(s) is created in the store. Please note that only major-minor Key associations are defined with 'Value' set to an empty string. This would

Major-Key[1]	Major-Key[2] (GenreID)	Minor-Key[1] (MovieID)	Value
GN_MV	8	829	""
GN_MV	3	829	""

Once external tables are setup for these three key-spaces (MV, GN, GN_MV) we will use SQL queries to fetch the data from Oracle NoSQL Database.

Instructions:

Bring up a command prompt (terminal window) and go to your `KVHOME` directory

1. Open a terminal window
2. There are three Oracle NoSQL Database specific environment variables.
    a. KVHOME: where binaries are installed,
    b. KVROOT: where data files and config files are saved
    c. KVDEMOHOME: where source of hands-on-lab project is saved.

```
echo  $KVROOT
echo $KVHOME
echo $KVDEMOHOME
```

3. Make sure you delete old $KVROOT (if exist already)

```
rm  -rf  $KVROOT
```

4. Change directory to /u02

```
cd /u02
```

5. Unzip kvroot.gold.zip file that has movie data already loaded.

```
unzip kvroot.gold.zip
```

6. Change directory to $KVHOME/lib

```
cd  $KVHOME/lib
```

7. Start KVLite in the lib directory:

```
java -jar kvstore.jar kvlite –host localhost -root
$KVROOT
```

```
java -jar $KVHOME/lib/kvstore-2.*.jar kvlite -root  $KVROOT
Created new kvlite store with args:
-root /u02/kvroot -store kvstore -host localhost -port 5000 -
admin 5001
```

This will start the KV instance in the foreground. Keep the instance running and open a new tab.

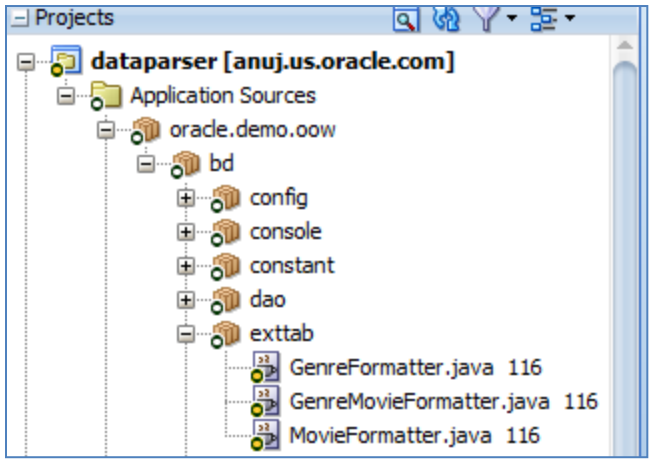
Lab Exercise 2 – Run Formatter classes to display the NoSQL data.

In this lab, you will configure compile the formatter classes and execute the classes to see how the data is going to display from the classes. We a will also configure the *nosql_stream* script based on the environment settings.

1. Open JDeveloper

```
$ jdev &
```

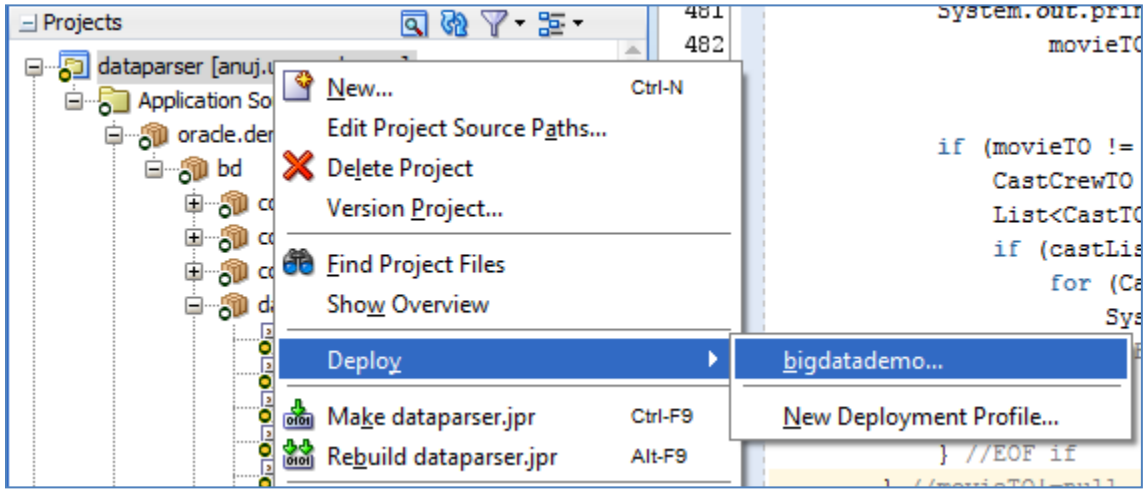
2. In the *dataparser* project three Formatter class already exist under package **oracle.demo.oow.bd.exttab**



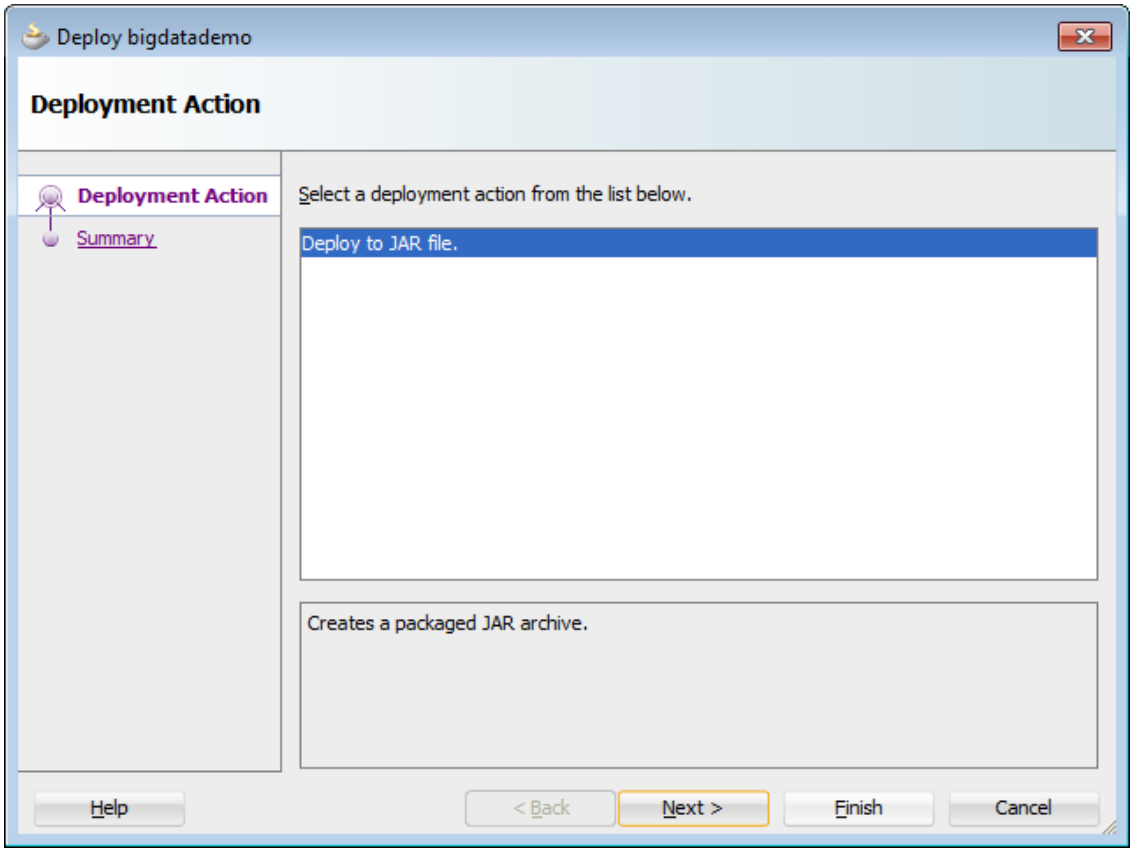
3. Open class GenreFormatter, MovieFormatter, GenreMovieFormatter one by one by double clicking the files from the left pane in JDev.

4. You can also run each of the three formatter classes to see how they return '|' delimited string. To run right click the class name and select Run (make sure Oracle NoSQL Database instance is up).

5. Compile and deploy the project by right clicking the '*dataparser*' project from the left pane and select deploy option (select finish when prompted).



6. On the next screen select 'Deploy to JAR file' and hit 'Finish' button.



This will write *bigdatademo.jar* file under *$KVDEMOHOME/dataparser/deploy* directory.

7. Next we will create couple of directories on the hard disk:
   a. ***/u01/nosql/test/exttab/data,*** config files (*.dat) are going to be stored
   b. ***/u01/nosql/test/exttab/bin*** we need to copy *nosql_stream*

```
mkdir -p /u01/nosql/test/exttab/data
mkdir -p /u01/nosql/test/exttab/bin
```

8. Copy ***nosql_stream*** file from the KVROOT to '/u01/nosql/test/exttab/bin' directory

```
cp $KVHOME/exttab/bin/nosql_stream /u01/nosql/test/exttab/bin
```

9. Open the ***nosql_stream*** file and set the environment variables based on your environment:

```
vi /u01/nosql/test/exttab/bin/nosql_stream
```

```
#!/bin/bash

#
# See the file LICENSE for redistribution information.
#
# Copyright (c) 2010, 2012 Oracle and/or its affiliates.  All rights
reserved.
#
# Fill in site-appropriate values for PATH, KVHOME, and CLASSPATH
# This script will be executed by the Oracle Database server.
# If you are using user-specified Formatters, by sure to include their home
# in the CLASSPATH.
#
export PATH=$PATH:/usr/java/latest/bin
export KVHOME=/u01/nosql/kv-2.0.26
export DEMOHOME=/home/oracle/movie/moviedemo/nosqldb/bigdatademo/dataparser
export CLASSPATH="$KVHOME/lib/*:$DEMOHOME/lib/*:$DEMOHOME/deploy/*"
java oracle.kv.exttab.Preproc $*
```

Lab Exercise 3 – Define External Tables in SQL.

In this lab, you will expose movie & genre information stored in Oracle NoSQL Database as external tables.

1. Login as sysdba to your running instance of ORA

```
sqlplus / as sysdba
```

1. Next create a directory where external tables *.dat files are going to be stored

```
CREATE OR REPLACE DIRECTORY ext_tab AS  '/u01/nosql/test/exttab/data';
```

2. Create a directory where ***nosql_stream*** script is saved

```
CREATE OR REPLACE DIRECTORY  nosql_bin_dir AS '/u01/nosql/test/exttab/bin';
```

3. Now create a new user and give some required roles and permission to it

```
CREATE USER nosqluser IDENTIFIED BY welcome1;
GRANT CREATE SESSION TO nosqluser;
GRANT EXECUTE ON SYS.UTL_FILE TO nosqluser;
GRANT READ, WRITE ON DIRECTORY ext_tab TO nosqluser;
GRANT READ, EXECUTE ON DIRECTORY nosql_bin_dir TO nosqluser;
GRANT CREATE TABLE TO nosqluser;
```

4. Now connect as nosqluser (that we just created).

```
CONNECT  nosqluser/welcome1;
```

5. Check if there are any existing tables

```
SELECT TABLE_NAME FROM USER_TABLES;
```

6. (optional) If there are any tables then drop them first

```
DROP TABLE MOVIE ;
DROP TABLE GENRE;
DROP TABLE GENRE_MOVIE ;
```

7. Lets create external tables

- Create GENRE table first:

```
CREATE TABLE GENRE (ID NUMBER(5), NAME VARCHAR2(30))
        ORGANIZATION EXTERNAL
            (type oracle_loader
             default directory ext_tab
             access parameters (records delimited by newline
             preprocessor nosql_bin_dir:'nosql_stream'
            fields terminated by '|')
       LOCATION ('genre.dat'))
       PARALLEL;
```

- Movie table next:

```
CREATE TABLE MOVIE (MOVIEID NUMBER(10), TITLE VARCHAR2(512),
YEAR NUMBER(4), POPULARITY NUMBER, VOTE NUMBER(10),
DESCRIPTION VARCHAR2(3072))
        ORGANIZATION EXTERNAL
            (type oracle_loader
             default directory ext_tab
             access parameters (records delimited by newline
             preprocessor nosql_bin_dir:'nosql_stream'
            fields terminated by '|')
       LOCATION ('movie.dat'))
       PARALLEL;
```

- And finally GENRE_MOVIE table:

```
CREATE TABLE GENRE_MOVIE (GENREID NUMBER(5), MOVIEID
NUMBER(10))
        ORGANIZATION EXTERNAL
            (type oracle_loader
             default directory ext_tab
             access parameters (records delimited by newline
             preprocessor nosql_bin_dir:'nosql_stream'
            fields terminated by '|')
       LOCATION ('genmov.dat'))
       PARALLEL;
```

8. View tables to make sure three tables are created:

```
SELECT TABLE_NAME FROM USER_TABLES;
```

9. Also describe all three tables one by one to see what are all the columns per tables

```
DESC MOVIE;
DESC GENRE;
 DESC GENRE_MOVIE;
```

Lab Exercise 4 – Define Configuration file (use template).

In this lab, we will create external-tables config file (from the template provided under ***$KVHOME/example/externaltables/config.xml*** directory) and edit them to reflect environment details.

1. Open genre_config file for edit

```
vi  $KVHOME/examples/externaltables/genre_config.xml
```

2. Notice database connection details:

```
    <property name="oracle.kv.exttab.connection.url"
        value="jdbc:oracle:thin:/@localhost:1521:orcl"
        type="STRING"/>
    <property name="oracle.kv.exttab.connection.user"
        value="nosqluser" type="STRING"/>
    <property name="oracle.kv.exttab.tableName" value="genre"
        type="STRING"/>
```

3. Notice Oracle NoSQL Database connection details and the Genre prefix 'GN' used as *parentKey*:

```
    <property name="oracle.kv.kvstore"
        value="kvstore"
        type="STRING"/>
    <property name="oracle.kv.hosts"
        value="localhost:5000"
        type="STRING"/>
    <property name="oracle.kv.parentKey"
        value="/GN"
        type="STRING"/>
    <property name="oracle.kv.formatterClass"
        value="oracle.demo.oow.bd.exttab.GenreFormatter"
        type="STRING"/>
```

4. Open movie_config file for edit

```
vi  $KVHOME/examples/externaltables/movie_config.xml
```

5. Notice only three parameters are different in *movie_config.xml* (from *genre_config.xml*):

```
    <property name="oracle.kv.exttab.tableName"
        value="movie"
        type="STRING"/>
    <property name="oracle.kv.parentKey"
        value="/MV"
        type="STRING"/>
    <property name="oracle.kv.formatterClass"
        value="oracle.demo.oow.bd.exttab.MovieFormatter"
        type="STRING"/>
```

6. Open genre_movie_config file for edit

```
vi  $KVHOME/examples/externaltables/genre_movie_config.xml
```

7. Notice again only three parameters are different in *genre_movie_config.xm* :

```
    <property name="oracle.kv.exttab.tableName"
        value="genre_movie"
        type="STRING"/>
    <property name="oracle.kv.parentKey"
        value="/GN_MV"
        type="STRING"/>
    <property name="oracle.kv.formatterClass"
        value="oracle.demo.oow.bd.exttab.GenreMovieFormatter"
        type="STRING"/>
```

Lab Exercise 5 – Use NoSQL Database Publish Utility.

In this lab, you will create movie.dat, genre.dat & genre_movie.dat using publish utility.

1. Change directory to KVHOME

```
cd $KVHOME
```

2. Publish genre_config.xml

```
java -cp $KVHOME/lib/kvstore-ee.jar:$KVHOME/lib/kvstore.jar:$ORACLE_HOME/jdbc/lib/ojdbc6.jar oracle.kv.exttab.Publish -config $KVHOME/examples/externaltables/genre_config.xml -publish -verbose
```

3. Publish movie_config.xml

```
java -cp $KVHOME/lib/kvstore-ee.jar:$KVHOME/lib/kvstore.jar:$ORACLE_HOME/jdbc/lib/ojdbc6.jar oracle.kv.exttab.Publish -config $KVHOME/examples/externaltables/movie_config.xml -publish -verbose
```

4. Publish genre_movie_config.xml

```
java -cp $KVHOME/lib/kvstore-ee.jar:$KVHOME/lib/kvstore.jar:$ORACLE_HOME/jdbc/lib/ojdbc6.jar oracle.kv.exttab.Publish -config $KVHOME/examples/externaltables/genre_movie_config.xml -publish -verbose
```

5. List **ext_tab** directory to make sure you have three .dat files:

```
$ ls  -1 /u01/nosql/test/exttab/data

genre.dat
movie.dat
genmov.dat
```

Lab Exercise 6 – Use SQL to access data

In this lab, you will access NoSQL data from Oracle Database using SQL queries.

1. At this point we have finished all the configurations and have exposed three key spaces as external tables. One can access NoSQL data using SQL queries now

    - Show tables for schema nosqluser

```
SELECT TABLE_NAME
        FROM USER_TABLES;
```

    - Show GENRES

```
SELECT *
       FROM GENRE
       ORDER BY ID;

         ID NAME
---------- ------------------------------
          1 History
          2 Animation
          3 Drama
          6 Comedy
          7 Action
          8 Crime
          9 Thriller
         10 Documentary
         11 Adventure
         12 Fantasy
         14 Family
```

    - Show movies that has popularity more than 8 and at least 10,000 votes

```
SELECT MOVIEID, TITLE, POPULARITY, VOTE
        FROM MOVIE
        WHERE POPULARITY > 8 AND VOTE > 10000;
```

    - Show top 'Drama' movies. This query uses join of all the three tables

```
SELECT  M.TITLE
         FROM MOVIE M, GENRE G, GENRE_MOVIE GM
         WHERE G.ID=GM.GENREID AND GM.MOVIEID=M.MOVIEID  AND
G.NAME='Drama' AND POPULARITY>8 AND VOTE>10000;
```