DAT 161 – Introduction to SAP HANA Cloud

Contents

[HANA Cloud Hands on exercise 1](#_Toc56689799)

[Pre-requisite 1](#_Toc56689800)

[HANA Cloud, Provisioning 3](#_Toc56689801)

[Exercise 1. Walk Through the Provisioning Process 3](#_Toc56689802)

[Exercise 2. Update a provisioned instance (Not Applicable for Trial Instances) 6](#_Toc56689803)

[Introduction to HANA Cloud, Data Lake 9](#_Toc56689804)

[Exercise 1. Create Schema 9](#_Toc56689805)

[Exercise 2. Create physical tables in Data Lake 12](#_Toc56689806)

[Exercise 3. Create Virtual Tables 14](#_Toc56689807)

[Exercise 4. Load data into physical tables 16](#_Toc56689808)

[Exercise 5. Query data from Virtual Table 17](#_Toc56689809)

[Setting up a Remote Source to Athena 18](#_Toc56689810)

[Exercise 1. Query the remote source and create a virtual table 18](#_Toc56689811)

[Exercise 2. Create virtual tables 26](#_Toc56689812)

[Exercise 3. Query Virtual Tables 28](#_Toc56689813)

[Exercise 4. Turn on Table Replication 29](#_Toc56689814)

[Exercise 5. Re-Run Query and Note Performance Difference 30](#_Toc56689815)

[Operating HANA with Native Storage Extension 31](#_Toc56689816)

[Prerequisite. Open WebIDE and connect to HANA system 31](#_Toc56689817)

[Exercise 1. Execute SQL to create a table in NSE storage 37](#_Toc56689818)

[Exercise 2. View the Load Unit configuration 42](#_Toc56689819)

[42](#_Toc56689820)

[43](#_Toc56689821)

[43](#_Toc56689822)

[Exercise 3. Create and use a table with a specific column configured as page loadable 44](#_Toc56689823)

[Exercise 4. View the load unit configuration of individual columns 46](#_Toc56689824)

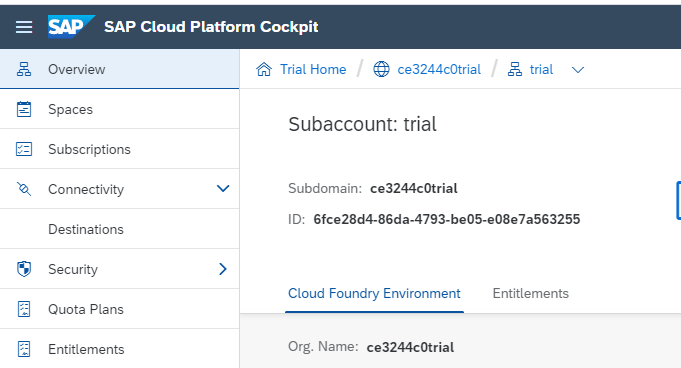
[Exercise 5. Create and use a table with a specific partition configured as page loadable 49](#_Toc56689825)

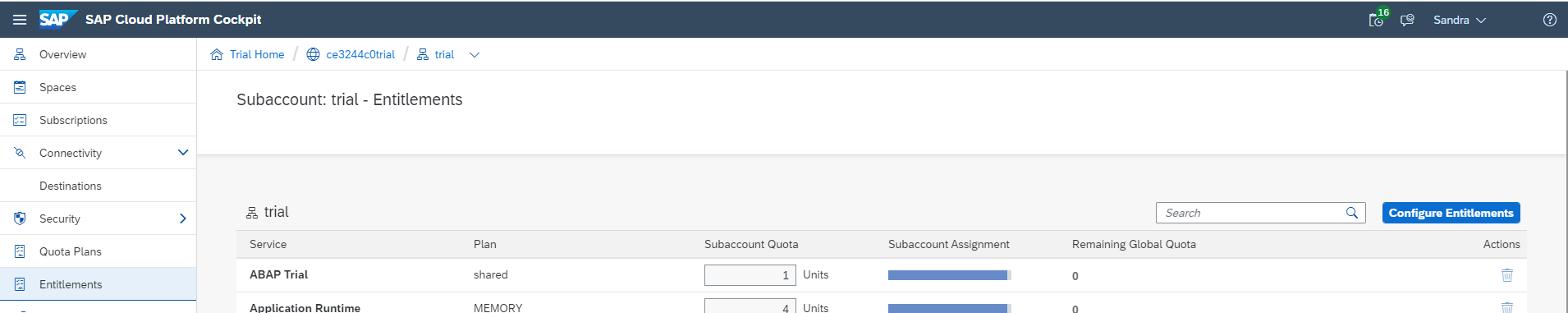
[Exercise 6. View the load unit configuration of individual partitions 53](#_Toc56689826)

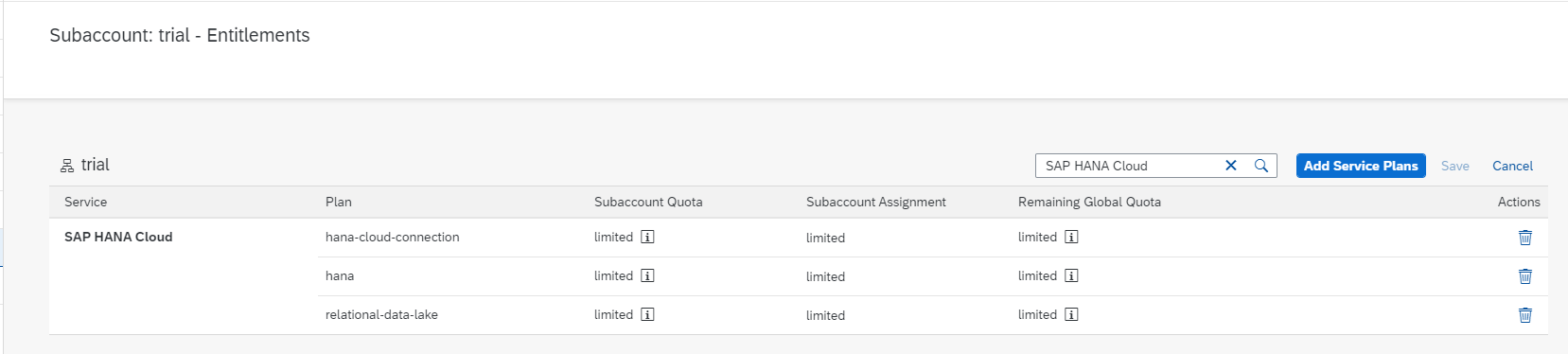
# HANA Cloud Hands on exercise

## Pre-requisite

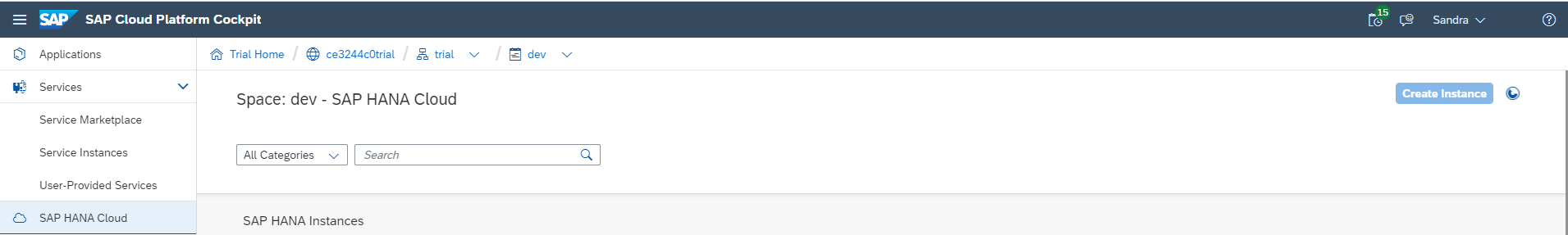
* 1. The following exercise uses HANA Cloud Trial. If you have a trial account, go to Subaccount Entitlements and add SAP HANA Cloud if you haven’t already. If you do not have an active trial account, sign-up through our HANA Cloud Trial registration form – available here: <https://www.sap.com/cmp/td/sap-hana-cloud-trial.html>







* 1. Once you log on to SAP Cloud Platform Cockpit, ensure that you have navigated to the correct global account, sub account, space, and you have selected SAP HANA Cloud from the left sidebar



# HANA Cloud, Provisioning

## Exercise 1. Walk Through the Provisioning Process

The goal of this exercise is to create a HANA Cloud instance and a Data Lake instance connected to it. Note, trial users may only create one instance per subaccount per geographic location. If you have already created an instance with your trial account, but have not created a data lake instance, complete steps 2 and 3. If you already have a data lake instance, skip to step 4 to ensure it is running.

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1. From your SAP Cloud Platform trial account, open the “SAP HANA Cloud” item and click “Create Database”  Follow through the wizard to create a HANA Cloud instance.  Be sure to create a HANA data lake instance when given the option. |  |
| 2. Complete step 2 and 3 if you already have a HANA Cloud instance, but have not created a HANA data lake instance.  From the HANA Cloud service tile inside of SCP, click ‘Actions’ and choose “Monitor landscape” |  |

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 3.  When the HANA Cloud manager opens, choose “Add Data Lake” from the “Actions” menu in the landscape monitor. |  |
| 4.  When the HANA Cloud instance is ready the status will change to *Running*.  If the instance was previously created and is in the “Stopped” state, you can start it from the “Actions” menu. |  |

## Exercise 2. Update a provisioned instance (Not Applicable for Trial Instances)

The goal of this exercise is to update the resources allocated to a provisioned data lake instance. HANA Cloud trial instances cannot be edited. This exercise only applies to provisioned instances of the full HANA Cloud service.

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1.  Open the HANA Cloud management interface and click on the “Edit” option under the Actions extended menu. |  |
| 2.  Increase Compute to 10 vCPU and increase storage to 4 TB.  Note: we have now allocated 8 vCPUs to workers, and 2 vCPUs are coordinators.  Click on *Save.* |  |
| 3.  Wait for the updated instance to start again. Once *Starting* turns to *Running*, the changes have been reflected in the Data Lake instance. This process could take a few minutes. |  |

# Introduction to HANA Cloud, Data Lake

## Exercise 1. Create Schema

The goal of this exercise is to create a schema in the HANA Cloud instance.

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1.  Navigate to the HANA Cloud instance we just created: *HDL\_EXERCISE-hana*. Click on open in and select SAP HANA database explorer |  |
| 2.  Enter Credential if prompted:  User: “DBADMIN”  Password: “HDLexercise1” |  |
| 3.  Click on HDL\_EXERCISE-hana to ensure it is selected. Then click on the “SQL” icon on the top left corner to launch a SQL console. |  |
| 4.  Once the SQL Console is launched, type  *Create Schema HDLExercise;*  Note: *SQL Console 1.sql* is connected to HDL\_EXERCISE-hana instance. |  |
| 5.  Click on the green run button to execute the query.  Make sure the query executes successfully by verifying in the *Messages* tab. |  |
| 6.  Check the Schema “HDLExercise” has been created by navigating to Catalog, Schemas, and then type “HDLExercise” in the Choose schema input form.  Ensure that HDLExercise exists. |  |

## Exercise 2. Create physical tables in Data Lake

The goal of this exercise is to create physical tables in the data lake.

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1.  Copy and paste the command from DAT161*Exercise\_CreateTable.sql* into SQL console. Execute the command by pressing the run button. Ensure the commands execute successfully.  This command creates the following tables in data lake:  EVENT, GKG, MENTIONS.  Note the create table statements are wrapped inside the *SYSRDL#CG.remote\_execute* function. *remote\_execute()* is a stored procedure which allows you to execute SQL natively in data lake. |  |
| 2.  Verify the tables have been created in Data lake.  First, navigate to *Catalog*, then *Remote Sources* from the left side bar. Click on *Remote Sources*. You should see one remote source called *SYSRDL#CG\_SOURCE*.  Double Click on *SYSRDL#CG\_SOURCE*. |  |
| 3.  This opens a tab next to *SQL Console 1.sql.*  Choose *SYSRDL#CG* from drop down menu for Schema.  Click on the Search button.  The tables that we just created should appear in the *Remote Objects* section. |  |

## Exercise 3. Create Virtual Tables

The goal of this exercise is to create virtual objects referring to the physical tables in the data lake.

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1.  From the same interface as the previous exercise, select all the tables, and click on *Create Virtual Objects*. |  |
| 2.  Set *Object Names Prefix* to V\_  Then select the schema created from the previous exercise *HDLEXERCISE*, and click on create.  This creates a virtual table for each physical table, and names the virtual table V\_<name of physical table>. For instance, the virtual table referring to EVENT will be named V\_EVENT.  The virtual tables are associated with the schema HDLEXERCISE. |  |
| 3.  To verify the virtual tables have been created successfully, navigate to *Catalog*, then *Tables*.  Type *HDLExercise* in the choose schema input form.  You should see the virtual tables that you have just created. |  |

## Exercise 4. Load data into physical tables

The goal of this exercise is to load data from an external source (an AWS s3 bucket), into the physical tables we have previously created in data lake.

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1.  Copy and paste the command from DAT161Exercise\_*LoadTable.sql* into the SQL console. Execute the command by pressing the run button. Ensure the commands execute successfully.  This command loads data from a public AWS s3 bucket into the following tables in data lake:  EVENT, GKG, MENTIONS.  Note that we supply the location of the s3 bucket, its region, its access key id, as well as its secret access key so that data lake can read the data directly from the S3 bucket, and the data is not going through the Hana Cloud instance. |  |
| 2.  This process could take a few minutes. Verify that the statements execute successfully. |  |

## Exercise 5. Query data from Virtual Table

The goal of this exercise is to query data from virtual tables.

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1.  Copy and paste the command from *Query1.sql* into SQL console. Execute the command by pressing the run button. Ensure the commands execute successfully. |  |
| 2.  Copy and paste the command from *Query2.sql* into SQL console. Execute the command by pressing the run button. Ensure the commands execute successfully. |  |

# Setting up a Remote Source to Athena

## Exercise 1. Query the remote source and create a virtual table

The goal of this exercise is to set up a remote source to Athena and create virtual tables using the SAP HANA database explorer.

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1. Create a remote source called “ATHENASOURCE” by running the first statement in DAT161*Exercise\_Athena.sql* under *“-- Create Remote Source”* in a SQL console |  |
| 2. Add AWS and S3 certificates by copying the `CREATE CERTIFICATE FROM …` statement under *“-- Add AWS certificate and S3 certificate”* from DAT161Exercise\_Athena.sql |  |
| 3. Create PSE HTTPS by running the following statement in the SQL console:  *CREATE PSE HTTPS;* |  |
| 4. Get the CERTIFICATE\_ID value for AWS by running the following statement in the SQL console:  *SELECT CERTIFICATE\_ID FROM CERTIFICATES WHERE COMMENT = 'AWS';*  Note: keep track of the CERTIFICATE\_ID as it will be used in the next step |  |
| 5. Alter PSE HTTPS to add the AWS CERTIFICATE\_ID by running the following statement in the SQL console:  ALTER PSE HTTPS ADD CERTIFICATE [CERTIFICATE\_ID]  Note: replace [CERTIFICATE\_ID] with the AWS CERTIFICATE\_ID from previous step |  |
| 6. Now repeat step 4-5 for the S3 certificate.  First, get the CERTIFICATE\_ID value for S3 by running the following statement in the SQL console:  *SELECT CERTIFICATE\_ID FROM CERTIFICATES WHERE COMMENT = 'S3';*  Note: keep track of the CERTIFICATE\_ID as it will be used in the next step |  |
| 7. Alter PSE HTTPS to add the S3 CERTIFICATE\_ID by running the following statement in the SQL console:  ALTER PSE HTTPS ADD CERTIFICATE [CERTIFICATE\_ID]  Note: replace [CERTIFICATE\_ID] with the S3 CERTIFICATE\_ID from previous step |  |
| 8. Set the purpose of PSE HTTPS to remote source by running the following statement:  *SET PSE HTTPS PURPOSE REMOTE SOURCE;* |  |
| 9. Now, verify that AthenaSource has been added as a remote source to the HANA Cloud instance by navigating to Catalog, and then Remote Sources. Double click on ATHENASOURCE. |  |
| 10. Select the Schema *tpchsf01* from the drop down. Click on Search to reveal a list of remote objects. |  |

## Exercise 2. Create virtual tables

The goal of this exercise is to create virtual tables pointing to remote objects in ATHENASOURCE.

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1. Select all object in the remote source and click on the `Create Virtual Object(s)` button |  |
| 2. Set prefix as ATH\_ and leave Schema as DBADMIN. Click on Create. |  |
| 3. Once the virtual objects are created, navigate to Catalog, Tables to verify the tables are created successfully. |  |

## Exercise 3. Query Virtual Tables

The goal of this exercise is to query virtual tables for results and evaluate its performance

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1. Select all the rows in the ATH\_nation virtual table that we have just created by running the following statement:  SELECT \* FROM DBADMIN."ATH\_nation"; |  |
| 2. Note the performance of the statement which queries a virtual table (3226 ms). |  |

## Exercise 4. Turn on Table Replication

The goal of this exercise is to execute the statement which enables table replication – this will move data from a remote source into HANA Cloud’s main memory.

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1. Replicate the table in HANA Cloud by running the following statement:  ALTER VIRTUAL TABLE DBADMIN."ATH\_nation" ADD SHARED SNAPSHOT REPLICA;  Note: to avoid excessive memory usage this statement should ideally be used with smaller tables |  |

## Exercise 5. Re-Run Query and Note Performance Difference

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1. Rerun the statement  SELECT \* FROM DBADMIN."ATH\_nation";  Note: since the table is now in HANA Cloud’s main memory, the query should execute much faster. The performance difference between Exercise 3 and Exercise 5, while executing the same query, **went from 3226 ms to 2 ms.** |  |

# Operating HANA with Native Storage Extension

## Prerequisite. Open WebIDE and connect to HANA system

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1. Navigate to the subaccount level, click on Subscriptions on the left |  |
| 2. Find SAP Web IDE tile, and click on Go to Application |  |
| 3. When prompted to add a database to the database explorer, click on yes |  |
| 4. Navigate to your HANA Cloud instance and copy the end point |  |
| 5. Select database type as SAP HANA Database, and paste the end point into  Host field. Remove the port number from host url, and input it as Port number.  Note: the port number is typically the last 3 digits of the host url. For example, for the endpoint:  1b132f97-cf25-45dd-a098-699a7b2418e9.hana.trial-us10.hanacloud.ondemand.com:443  Input:  1b132f97-cf25-45dd-a098-699a7b2418e9.hana.trial-us10.hanacloud.ondemand.com as the **Host** and 443 as the **Port Number** |  |
| 6. Enter DBADMIN as user and the password. Select Save password and Connect to the database securely using TLS/SSL. Leave the third one unchecked. Optionally edit “Name to Show in Display” to something easily recognizable. Click on OK. |  |
| 7. You should now see the database |  |

## Exercise 1. Execute SQL to create a table in NSE storage

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1. Open a SQL console by selecting the database and clicking on the SQL icon |  |
| 2. First create a table in HANA Cloud main memory by running the following statement:  CREATE COLUMN TABLE "DBADMIN"."T1\_IN\_MEMORY"  ( COL1 INTEGER PRIMARY KEY,  COL2 VARCHAR(30)  ); |  |
| 3. Now create an NSE table by first copying and pasting the following statement to the SQL console:  CREATE COLUMN TABLE "DBADMIN"."T2\_NSE"  ( COL1 INTEGER PRIMARY KEY,  COL2 VARCHAR(30)  ) PAGE LOADABLE;  Note: notice the only difference in syntax when creating an NSE table compared to an in-memory table is the PAGE LOADABLE clause |  |
| 4. Select and highlight the statement and press the run button to execute it |  |
| 5. Insert values into both tables by running the following statements:  INSERT INTO "DBADMIN"."T1\_IN\_MEMORY" VALUES (1,'First in-memory record');  INSERT INTO "DBADMIN"."T1\_IN\_MEMORY" VALUES (2,'Second in-memory record');  INSERT INTO "DBADMIN"."T2\_NSE" VALUES (1, 'First page loadable record');  INSERT INTO "DBADMIN"."T2\_NSE" VALUES (2, 'Second page loadable record');  Note: there is no syntax difference between the INSERT INTO statements. Similarly, no extra syntax is required for any DML operations (UPDATE and DELETE) on page loadable data. |  |
| 6. Query the tables by running the following SELECT statements:  SELECT \* FROM "DBADMIN"."T1\_IN\_MEMORY" WHERE COL1 = 2;  SELECT \* FROM "DBADMIN"."T2\_NSE" WHERE COL1 = 1;  Note: The result from the first query is returned under the tab “Result 1” and the result from the second query is returned under the tab “Result 2”. |  |
| 7. The tables can also be viewed from the UI by navigating to Catalog and then Tables |  |

## Exercise 2. View the Load Unit configuration

|  |  |
| --- | --- |
| 1. Clear the SQL console. Then copy and paste the following statement into the console. Run the statement.  SELECT SCHEMA\_NAME, TABLE\_NAME, LOAD\_UNIT FROM SYS.TABLES  WHERE SCHEMA\_NAME = 'DBADMIN';  Note: for this exercise, we are specifically looking at the SCHEMA\_NAME, TABLE\_NAME, and LOAD\_UNIT columns in the SYS.TABLES view. |  |
| 2. Click on “1”.  Since we did not explicitly assign a LOAD UNIT in the CREATE TABLE statement for the T1\_IN\_MEMORY table, it shows a LOAD UNIT of **DEFAULT**. The default LOAD UNIT for a table is **COLUMN**. If we had explicitly specified **COLUMN LOADABLE** in the CREATE TABLE statement for the T1\_IN\_MEMORY table, then the LOAD UNIT would show as **COLUMN**. |  |
| 3. click on “2”.  The LOAD UNIT for the **T2\_NSE** table is reported as **PAGE** indicating that this is a page loadable table. |  |

## Exercise 3. Create and use a table with a specific column configured as page loadable

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1. Clear the SQL console and run the following statement:  CREATE COLUMN TABLE "DBADMIN"."T3\_ONE\_COLUMN\_IN\_NSE"  ( COL1 INTEGER PRIMARY KEY,  COL2 VARCHAR(30),  COL3 CLOB PAGE LOADABLE  );  Note that the first 2 column definitions are the same as we used in the T1 and T2 tables. For our T3 table, a third column has been added to the table definition and specifically designated that column to be **PAGE LOADABLE.** Note the location of the PABE LOADABLE clause as part of the column definition. There is no **PAGE LOADABLE** clause after the column definitions. This means that the table will use the DEFAULT load unit (which is COLUMN LOADABLE) for any columns that are not explicitly set to be PAGE LOADABLE. |  |
| 2. Now run the following statement to insert values into T3\_ONE\_COLUMN\_IN\_NSE:  INSERT INTO "DBADMIN"."T3\_ONE\_COLUMN\_IN\_NSE" VALUES (1, 'Col2 is COLUMN LOADABLE', 'Col3 is a CLOB to handle really long verbose content');  INSERT INTO "DBADMIN"."T3\_ONE\_COLUMN\_IN\_NSE" VALUES (2, 'Col2 is always in memory', 'Col3 can be read into the buffer cache when needed but only when needed'); |  |
| 3. Query the table by running the following statement:  SELECT \* FROM "DBADMIN"."T3\_ONE\_COLUMN\_IN\_NSE"; |  |

## Exercise 4. View the load unit configuration of individual columns

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1. Query the SYS.TABLES view to see what LOAD UNIT is configured at the table level by running the following statement:  SELECT SCHEMA\_NAME, TABLE\_NAME, LOAD\_UNIT FROM SYS.TABLES  WHERE SCHEMA\_NAME = 'DBADMIN'; |  |
| 2. Notice the LOAD UNIT for the T3\_ONE\_COLUMN\_IN\_NSE table is shown as “DEFAULT” (which is equivalent to COLUMN). If any 1 column in a table is COLUMN LOADABLE, then the LOAD UNIT for the table will be reported as DEFAULT or COLUMN. |  |
| 3. To see the LOAD UNIT configuration at a **column level**, run the following statement:  SELECT SCHEMA\_NAME, TABLE\_NAME, COLUMN\_NAME, LOAD\_UNIT FROM SYS.M\_CS\_COLUMNS  WHERE SCHEMA\_NAME = 'DBADMIN'; |  |
| 4. Note the **LOAD UNIT** for the third column is shown as **PAGE** because that column was specifically designated as **PAGE LOADABLE** in the CREATE TABLE statement in Exercise 2 Step 1.  Also note that the **LOAD UNIT** for the first 2 columns of **T3\_ONE\_COLUMN\_IN\_NSE** (row 6 and 7) are shown as **COLUMN** because that is the default LOAD UNIT and no other LOAD UNIT was specified for these columns in the CREATE TABLE statement. |  |

## Exercise 5. Create and use a table with a specific partition configured as page loadable

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1. Run the following statement to create a table:  CREATE COLUMN TABLE  "DBADMIN"."T4\_ONE\_PARTITION\_IN\_NSE"  ( COL1 INTEGER PRIMARY KEY,  COL2 VARCHAR(30),  COL3 CLOB  )  PARTITION BY RANGE ( "COL1" )  ( ( PARTITION 1 <= VALUES < 10 COLUMN LOADABLE,  PARTITION 10 <= VALUES < 20 PAGE LOADABLE,  PARTITION OTHERS  )  );  Note: The column definitions are the same as the columns created in T3, but since no **LOAD UNIT** is specified, all the columns will be assigned the default load unit of **COLUMN**.  The new CREATE TABLE clause that is being used for the T4 table is the **PARTITION BY RANGE** clause. This defines a single level range partitioning based on the values in COL1. |  |
| 2. Note: In the previous SQL statement, the first partition will hold records where the VALUES of COL1range from 1 to 9 (COL1 is an INTEGER column). This partition is explicitly assigned to be stored **in-memory** as **COLUMN LOADABLE**.  The second partition will hold records where the VALUES of COL1range from 10 to 19. This partition is explicitly assigned to be stored **on disk** as **PAGE LOADABLE**.  The 3rd partition is the 'catch-all' partition OTHERS. The OTHERS partition is not assigned records in a specific range. Instead the OTHERS partition will hold any records where the value of COL1 falls outside of any of the other partitions. And since no LOAD UNIT is specified for the OTHERS partition, it will be assigned the **DEFAULT** load unit of **COLUMN**. |  |
| 3. Insert a record into each of the 3 partitions by running the following statement:  INSERT INTO "DBADMIN"."T4\_ONE\_PARTITION\_IN\_NSE" VALUES (1, 'This is column 2', 'The COL1 value falls in the range of the 1st partition so this record will be in a COLUMN LOADABLE partition. ');  INSERT INTO "DBADMIN"."T4\_ONE\_PARTITION\_IN\_NSE" VALUES (18, 'Column 2 again', 'The COL1 value falls in the range of the 2nd partition so this record will be in a PAGE LOADABLE partition');  INSERT INTO "DBADMIN"."T4\_ONE\_PARTITION\_IN\_NSE" VALUES (99, 'Hi from COL2', 'The COL1 value for this record falls outside of the ranges of the explicitly defined partitions so this record will be in the catch-all OTHERS partition'); |  |
| 4. Select from the table by running the following statement:  SELECT \* FROM "DBADMIN"."T4\_ONE\_PARTITION\_IN\_NSE"; |  |

## Exercise 6. View the load unit configuration of individual partitions

|  |  |
| --- | --- |
| Step Explanation | Screenshot |
| 1. Review the table level LOAD UNIT for the T4 table by running the following statement:  SELECT SCHEMA\_NAME, TABLE\_NAME, LOAD\_UNIT FROM SYS.TABLES  WHERE SCHEMA\_NAME = 'DBADMIN'  AND TABLE\_NAME = 'T4\_ONE\_PARTITION\_IN\_NSE';  Note: Just as with the T1 and T3 tables, the LOAD UNIT for the **T4\_ONE\_PARTITION\_IN\_NSE** table is reported as **DEFAULT** because no LOAD UNIT was explicitly set for overall table. Remember that DEFAULT is equivalent to **COLUMN**. |  |
| 2. Review the details of individual table partitions, including the LOAD UNIT configuration by running the following statement:  SELECT SCHEMA\_NAME, TABLE\_NAME, PART\_ID, LOAD\_UNIT FROM SYS.M\_TABLE\_PARTITIONS  WHERE SCHEMA\_NAME = 'DBADMIN';  Note: Since the only partitioned table in your schema is the **T4\_ONE\_PARTITION\_IN\_NSE** table, you will only see results for that table.  Partition IDs are assigned in the order that the PARTITION definitions appeared in the **CREATE TABLE** statement. |  |
| 3. Note: In the previous result:  The LOAD UNIT of the first partition was specifically configured as **COLUMN LOADABLE** in the CREATE TABLE statement, so it is shown as fieldicon_147.  The LOAD UNIT of the second partition was specifically configured as **PAGE LOADABLE** in the CREATE TABLE statement, so it is shown as fieldicon_157.  The LOAD UNIT of the **OTHERS** partition, which was defined as the 3rd partition, was not specifically configured in the CREATE TABLE statement, so it defaults to  fieldicon_163. (Note that the OTHERS partition is required to always be COLUMN LOADABLE). |  |