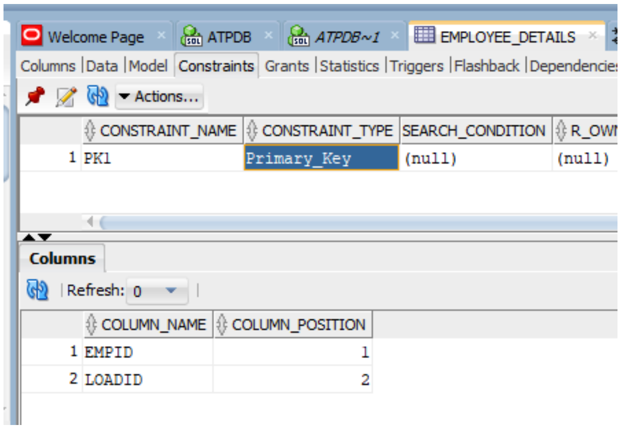
Case Study

Set up for Case Study

1) We have placed 2 files (employee.csv, employee_details.csv) in Object Storage for this case study. Compartment is OCI-DEMO02 and Bucket is DEMO02-BUCKET02. You will be using these files for the data loads.
2) EMPLOYEE_DETAILS, EMPLOYEE_DETAILS_AUDIT and EMPLOYEE table is created in ATPDB02 for this purpose. You will be using these tables for the data loads.
3) EMPLOYEE_DETAILS is configured with Primary Key for merge operations.

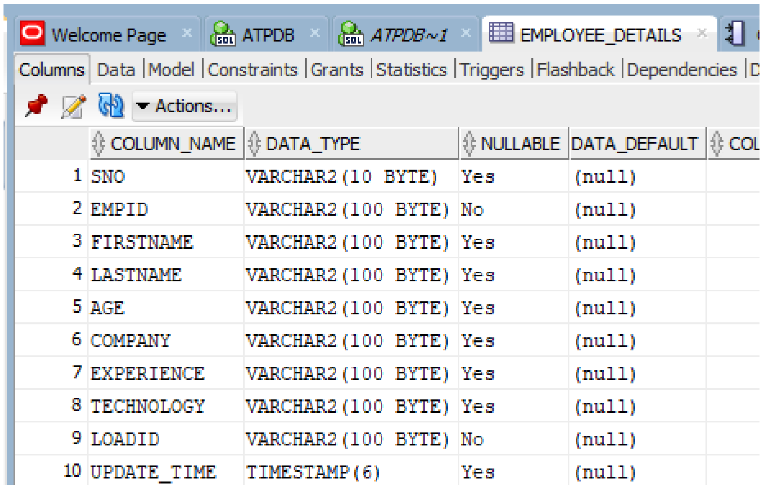


	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWI
1	PK1	Primary_Key	(null)	(null)

	COLUMN_NAME	COLUMN_POSITION
1	EMPID	1
2	LOADID	2

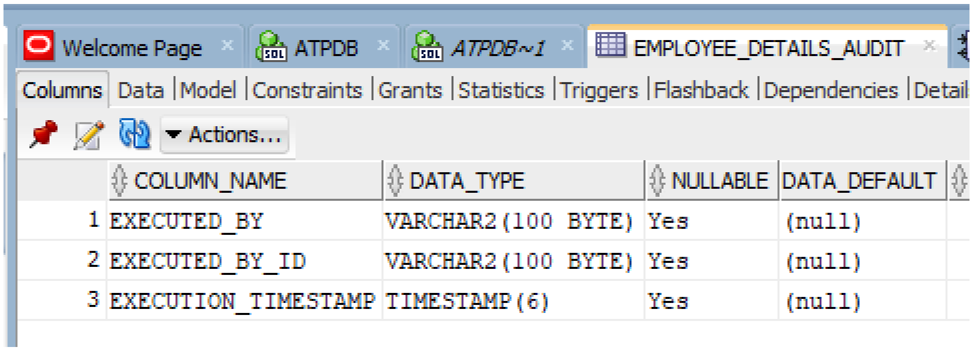
4) PROC_EMPLOYEE_DETAILS_AUDIT is created in ATPDB02, which when executed with values, inserts a row in EMPLOYEE_DETAILS_AUDIT table. It takes 3 values, as in parameters. When values passed, it inserts one record in audit table containing your Name, ID and execution time. For each run one entry is inserted.
    a. p_in_executed_by is Varchar -> Pass your name as value
    b. p_in_executed_by_id is Varchar -> Pass your Employee ID as value
    c. p_in_executed_time is Timestamp-> Pass execution Time as value

EMPLOYEE_DETAILS table structure:



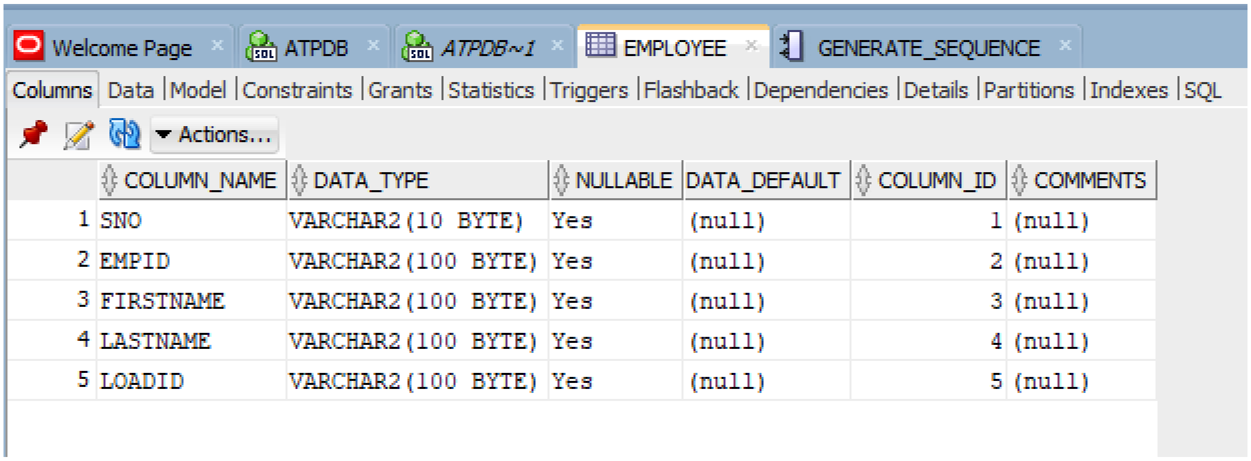
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COL
1	SNO	VARCHAR2(10 BYTE)	Yes	(null)	
2	EMPID	VARCHAR2(100 BYTE)	No	(null)	
3	FIRSTNAME	VARCHAR2(100 BYTE)	Yes	(null)	
4	LASTNAME	VARCHAR2(100 BYTE)	Yes	(null)	
5	AGE	VARCHAR2(100 BYTE)	Yes	(null)	
6	COMPANY	VARCHAR2(100 BYTE)	Yes	(null)	
7	EXPERIENCE	VARCHAR2(100 BYTE)	Yes	(null)	
8	TECHNOLOGY	VARCHAR2(100 BYTE)	Yes	(null)	
9	LOADID	VARCHAR2(100 BYTE)	No	(null)	
10	UPDATE_TIME	TIMESTAMP(6)	Yes	(null)	

EMPLOYEE_DETAILS_AUDIT table structure:



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT
1	EXECUTED_BY	VARCHAR2(100 BYTE)	Yes	(null)
2	EXECUTED_BY_ID	VARCHAR2(100 BYTE)	Yes	(null)
3	EXECUTION_TIMESTAMP	TIMESTAMP(6)	Yes	(null)

EMPLOYEE table structure:



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	SNO	VARCHAR2(10 BYTE)	Yes	(null)	1	(null)
2	EMPID	VARCHAR2(100 BYTE)	Yes	(null)	2	(null)
3	FIRSTNAME	VARCHAR2(100 BYTE)	Yes	(null)	3	(null)
4	LASTNAME	VARCHAR2(100 BYTE)	Yes	(null)	4	(null)
5	LOADID	VARCHAR2(100 BYTE)	Yes	(null)	5	(null)

Connection Creation

1) Create Object Storage Connection with name as DA_OBJSTORAGE_<EMP_ID>
2) Create Autonomous Transaction Processing with name as DA_ATPDB_<EMP_ID>

Project and Folder Creation

1) Create Project PROJECT_<EMP_ID>
2) Inside created project, create the below two folders.
   a. OBJSTORAGE_TO_ATP
   b. ATP_TO_OBJSTORAGE

Application Creation

1) Create an Application APPLICATION_<EMP_ID>

Object Storage to ATP Load

Create SQL Loader Task and Integration Task and run them parallelly in Pipeline Task.

Note: Configure the necessary parameters and setups so that UPDATE_TIME column in EMPLOYEE_DETAILS matches with EXECUTION_TIMESTAMP column in EMPLOYEE_DETAILS_AUDIT table while running them parallelly in pipeline task.

SQL Loader Task Creation for Object Storage to ATP

1) Create SQL loader task SQL_<EMP_ID> under OBJSTORAGE_TO_ATP to call procedure PROC_EMPLOYEE_DETAILS_AUDIT. Pass in the parameter values so that an entry is made in the EMPLOYEE_DETAILS_AUDIT table.
2) Publish the SQL Loader task to APPLICATION_<EMP_ID>

Data Flow and Integration Task Creation for Object Storage to ATP

1) Create a data flow DF_<EMP_ID> under OBJSTORAGE_TO_ATP. The data flow should contain the below features.
    a. Apply filter on “company” column in file to load only employees belonging to “Cognizant”.
    b. Join “empid” field from both the files (employee.csv, employee_details.csv) and load the matched records to EMPLOYEE_DETAILS table in ATPDB.
    c. The LOADID in EMPLOYEE_DETAILS table should contain your employee id.
    d. The “firstname” and “lastname” column should be loaded as Caps.
    e. The UPDATE_TIME in EMPLOYEE_DETAILS table should contain execution time.
    f. Use merge operation so that the records are getting merged when you are running multiple times. So that only the UPDATE_TIME column in EMPLOYEE_DETAILS gets updated for multiple runs with same sets of records.
2) Create an Integration Task IT_<EMP_ID> under OBJSTORAGE_TO_ATP and add the above created data flow.
3) Publish the integration task to APPLICATION_<EMP_ID>

Pipeline and Pipeline Task Creation for Object Storage to ATP

1) Create a pipeline PIPELINE_<EMP_ID> under OBJSTORAGE_TO_ATP and configure the above created data loader task and integration task to run in parallel.
2) Create a pipeline task PIPELINETASK_<EMP_ID> and add the above created pipeline.
3) Publish the above created pipeline task to APPLICATION_<EMP_ID>
4) Run the pipeline task and verify results.
    a. Joins data from both the files and loads only the matching and employees belonging to cognizant to EMPLOYEE_DETAILS table
    b. LOADID column in EMPLOYEE_DETAILS table contains the Employee ID passed.
    c. A row is inserted in the EMPLOYEE_DETAILS_AUDIT table containing Employee ID, Name and Execution time.
    d. UPDATE_TIME from EMPLOYEE_DETAILS matches with the EXECUTION_TIME in EMPLOYEE_DETAILS table.

e. When the pipeline task is executed multiple times, the records are getting merged in EMPLOYEE_DETAILS table and for every execution one entry is created under EMPLOYEE_AUDIT_DETAILS table.

ATP to Object Storage Load

Data Loader Task Creation for ATP to Object Storage

1) Create data loader task DL_<EMP_ID> under ATP_TO_OBJSTORAGE to create a file from EMPLOYEE table. The created data loader task should have the below features.
    a. When executed a new file should be created with name employee_<EMP_ID>.csv
    b. The file created should have an extra column FULLNAME. The FULLNAME must be concatenation of FIRSTNAME & LASTNAME columns.
    c. File should be created with header.
    d. File should be created under DEMO02-BUCKET02 bucket.
2) Publish the above created data loader task to APPLICATION_<EMP_ID>
3) Run the above created data loader task and verify results.