DOI: 10.53469/jgebf.2025.07(05).02

Strategic Data Pipeline Design: Improving Operational Efficiency from Oracle to Single Storage Using Airflow S3 Data Pipeline

Piyali Debnath

piyali812@gmail.com

Abstract: ***This paper investigates an innovative ETL pipeline managed by Apache Airflow, integrating Oracle databases with SingleStore through Amazon S3. The architecture enhances efficiency, scalability, and reliability of data integration processes. By implementing a sequence of orchestrated tasks, the study demonstrates improvements in data throughput and process automation compared to traditional ETL techniques. This article is significant as it addresses the need for scalable and efficient ETL processes in enterprise data management, demonstrating the potential improvements over traditional methods.***

Keywords: Data Integration, ETL, Apache Airflow, Oracle Database, Oracle Functions, Oracle External Directories, SingleStore Pipelines, SingleStore Procedure, Amazon S3 Storage, Data Transformation.

1. Introduction

This research outlines a data pipeline process through Apache Airflow and the tool for data source would be Oracle and SingleStore as a target. For the staging area, we use Amazon S3. This integration exploits the native features of these technologies for overcoming typical bottlenecks in data migration and processing scenarios. With the advent of the SingleStore pipeline, it has become easy to make sure that data transfers are smoother and faster in the enterprise world. The purpose of this article is to present an advanced ETL pipeline that enhances the efficiency, scalability, and reliability of data integration processes from Oracle to SingleStore using Apache Airflow and Amazon S3.

2. Problem Statement

Robust data integration solutions have become increasingly dynamic and scalable to meet the growing demand for efficient ETL processes. This paper introduces a sophisticated ETL pipeline leveraging Apache Airflow to orchestrate workflows, utilizing Oracle as the data source, SingleStore for storage through pipelines and procedures, and Amazon S3 as a buffering layer. This setup ensures high scalability and superior performance optimization in enterprise data management. Within the context of an oil and gas company, we were tasked with designing a process capable of rapidly loading data ranging from millions to billions of rows, addressing industry - specific challenges of speed and volume efficiently and to replace the existing traditional ETL Process.

Solution Implemented:
The solution consists of a meticulously designed sequence:

1. **Data Export from Oracle:** Using Oracle's built - in export capabilities, data is extracted and formatted into CSV files, which are subsequently uploaded to an S3 bucket. This extraction process is tailored for large datasets, employing parallel export operations to enhance efficiency. The operation utilizes an Oracle external directory that is configured to point directly to Amazon S3 storage. The data export task is managed by an Oracle function, which executes a SQL query specified in the configuration file to enable parallel data export. To set up the external directory in Oracle that points to Amazon S3 storage, use the following syntax:

Oracle Export Directory:

```
DROP DIRECTORY EXP_DIR_NFS_ODM_REF_VALUE_COM;

CREATE OR REPLACE DIRECTORY
EXP_DIR_NFS_ODM_REF_VALUE_COM AS
'/itc-data-replication/oracle_dumps/P3DATE_EOGRESOURCES_COM/ODM_DBA/ODM_REF_VALUE/CURRENT/COMPLETE';


GRANT EXECUTE, READ, WRITE ON DIRECTORY EXP_DIR_NFS_ODM_REF_VALUE_COM TO ORACLE_CDC WITH GRANT OPTION;
```

Oracle SQL stored in the configuration file:
Oracle SQL Stored with session parameters to define parallelism and others.

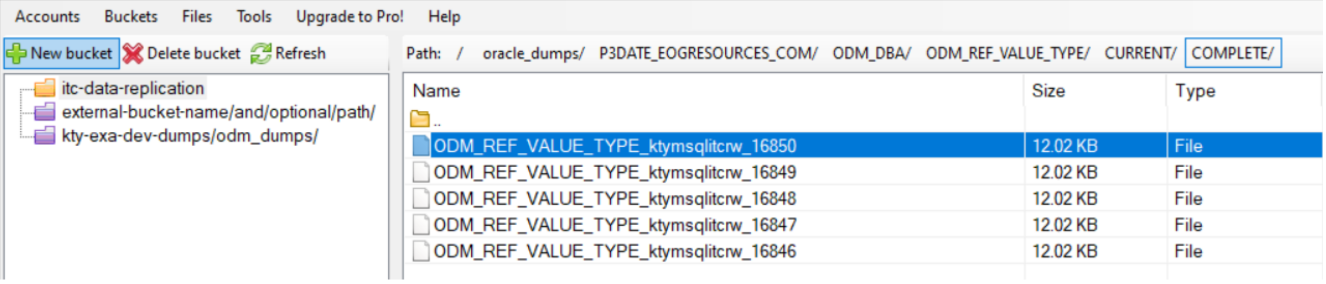
```
alter session set "_parallel_load_balancing"=false;
alter session set parallel_force_local=false;
alter session set "_px_granule_size"=1000000;
alter session set "_px_min_granules_per_slave"=1;
alter session set "_px_max_granules_per_slave"=1;
alter session set "optimizer_dynamic_sampling"=0;
alter session force parallel query parallel 1;
alter session set "_px_granule_batch_size"=4;
alter session set "_px_object_sampling"=0;
alter session set "_px_adaptive_dist_method" = OFF;
alter session set nls_date_format='YYYY-MM-DD HH24:MI:SS';

column sysdate new_value timestart;
select sysdate from dual;

column rowcountinitial new_value rowcountinitial2;
select count(1) rowcountinitial  from  ODM_DBA.ODM_REF_VALUE_TYPE ;

select COLUMN_VALUE processed from table(cast (FUNCTION_ODM_REF_VALUE_TYPE_PROD(cursor(
select  /*+ parallel(t 1) full(t) */
to_char(REF_VALUE_TYPE_ID)|| chr(124) || chr(124)  || chr(124)
||to_char(REF_VALUE_TYPE_NAME)|| chr(124) || chr(124)  || chr(124)
||to_char(CREATE_USER_ID)|| chr(124) || chr(124)  || chr(124)
||to_char(CREATE_TS, 'YYYY-MM-DD HH24:MI:SS')|| chr(124) || chr(124)  || chr(124)
||to_char(UPDATE_USER_ID)|| chr(124) || chr(124)  || chr(124)
||to_char(UPDATE_TS, 'YYYY-MM-DD HH24:MI:SS')|| chr(40) || chr(124)  || chr(124)  || chr(41)
from  ODM_REF_VALUE_TYPE t
)) as numset_t))
;
```

Oracle Function:

The Oracle function described utilizes a reference cursor to execute the SQL stored in a configuration file. Using the UTL_FILE utility, it exports the data to a directory specified by the external directories in Amazon S3.

```
CREATE OR REPLACE function FUNCTION_ODM_REF_VALUE_TYPE_PROD r refcur_p.refcur_t) return numset_t
   PIPELINED PARALLEL_ENABLE (PARTITION r BY ANY)   is    out                         utl_file.file_type;
   out2                        utl_file.file_type;    i                        binary_integer := 0;
   rec                       varchar2(32767);    type array is      table of varchar2(32767) index by binary_integer;
   l_data array;    v_cr       CONSTANT varchar2(1) := CHR(10);   v_crl     CONSTANT PLS_INTEGER := LENGTHB(v_cr);
   v_buf_max CONSTANT PLS_INTEGER := 32767;   v_buffer  varchar2(32767);   l_id number;    l_first number :=1;
begin
   l_id :=SEQ_ODM_REF_VALUE_TYPE.nextval;
   out := utl_file.fopen ('EXP_DIR_NFS_ODM_REF_VALUE_TYPE_COM', 'ODM_REF_VALUE_TYPE_ktymsqlitcrw_' || l_id , 'w',32767);

   loop

       fetch r bulk collect into l_data limit 100;
       for d in 1 .. l_data.count
       loop
         if (l_first = 1) then
            v_buffer := l_data(d);
            l_first :=0;
            continue;
         end if;
         IF case when v_buffer is null then 0 else LENGTHB(v_buffer) end + v_crl + case when l_data(d) is null
         then 0 else LENGTHB(l_data(d)) end <= v_buf_max THEN
                  v_buffer :=  v_buffer || v_cr || l_data(d) ;
         ELSE
               UTL_FILE.put_line(out, v_buffer);
               v_buffer := l_data(d);
         END IF;

       end loop;

       exit when r%notfound;
   end loop;
```

Data Stored in Amazon S3 Files for each week:



Data format: Data for one of the above file stored in S3.

ODM_REF_VALUE_TYPE_ktymsqlitcrw_16850 | new 389 | new 390 | new 383 | new 366 | new 370 | new 374

```
22083|||OVERRIDE_DIVISIONS|||0|||2014-01-23 18:42:25|||0|||2014-01-23 18:42:25(||)
23082|||RIG_TYPE|||0|||2014-03-03 19:59:31|||0|||2014-03-03 19:59:31(||)
24082|||MAJOR|||-1|||2014-03-06 20:33:27|||-1|||2014-03-06 20:33:27(||)
24083|||CURRENT_NRI_GAS|||-1|||2014-03-06 20:33:28|||-1|||2014-03-06 20:33:28(||)
24084|||CURRENT_NRI_NGL|||-1|||2014-03-06 20:33:28|||-1|||2014-03-06 20:33:28(||)
24085|||CURRENT_NRI_OIL|||-1|||2014-03-06 20:33:29|||-1|||2014-03-06 20:33:29(||)
24086|||CURRENT_WI|||-1|||2014-03-06 20:33:29|||-1|||2014-03-06 20:33:29(||)
32083|||DO_RIG_NAME|||0|||2014-09-09 18:45:45|||0|||2014-09-09 18:45:45(||)
35086|||ROUTE_TYPE|||-1|||2015-08-18 08:00:12|||-1|||2015-08-18 08:00:12(||)
35087|||TEAM|||0|||2015-08-26 14:27:16|||0|||2015-08-26 14:27:16(||)
36086|||PUMPDOWN_MISRUN_TYPE|||0|||2015-09-02 15:50:17|||0|||2015-09-02 15:50:17(||)
39086|||PRIOR_YR_CAT|||0|||2016-01-20 13:30:05|||0|||2016-01-20 13:30:05(||)
28082|||PROGRAM_YEAR_ID|||-1|||2014-05-28 18:52:22|||-1|||2014-05-28 18:52:22(||)
34083|||RIG_NAME_ONLY|||0|||2014-10-30 19:07:13|||0|||2014-10-30 19:07:13(||)
45090|||GEOLOGICAL_SUBTREND|||-1|||2020-11-02 14:13:19|||-1|||2020-11-02 14:13:19(||)
45091|||SUBGROUP|||-1|||2020-11-02 14:13:19|||-1|||2020-11-02 14:13:19(||)
45086|||SUBTREND3|||1|||2020-06-25 16:10:40|||0|||2020-06-25 16:10:40(||)
45092|||DIVISION_PAD|||-1|||2020-11-02 14:13:19|||-1|||2020-11-02 14:13:19(||)
```

2. **Data Loading to SingleStore:** Upon successful storage in S3, SingleStore pipelines are triggered to load the data directly from S3. These pipelines are designed for high throughput and minimal latency, ensuring data is quickly available for processing.

SingleStore Pipeline Syntax to create the pipeline:

```
CREATE PIPELINE `p_zwip_odm_ref_value_type`
AS LOAD DATA LINK link_itc_data_replication
'itc-data-replication/oracle_dumps/P3DATE_EOGRESOURCES_COM/ODM_DBA/ODM_REF_VALUE_TYPE/CURRENT/COMPLETE/ODM_REF_VALUE_TYPE_ktymsqlitcrw_*'
BATCH_INTERVAL 2500
DISABLE OUT_OF_ORDER OPTIMIZATION
SKIP ALL ERRORS
INTO TABLE `zwip_odm_ref_value_type`
FIELDS TERMINATED BY '|||' ENCLOSED BY '' ESCAPED BY ''
LINES TERMINATED BY '(||)\n' STARTING BY '' NULL DEFINED BY ''
(
    `zwip_odm_ref_value_type`.`ref_value_type_id`,
    `zwip_odm_ref_value_type`.`ref_value_type_name`,
    `zwip_odm_ref_value_type`.`create_user_id`,
    `zwip_odm_ref_value_type`.`create_ts`,
    `zwip_odm_ref_value_type`.`update_user_id`,
    `zwip_odm_ref_value_type`.`update_ts`
)
```

SingleStore Pipeline Execution using the following command:

start pipeline p_zwip_odm_ref_value_type foreground
stop pipeline p_zwip_odm_ref_value_type

After the Pipeline is executed, the data is loaded into the temporary staging table as show below:

```
select *
from zwip_odm_ref_value_type
```

odm_ref_value_type 1

t * from odm_dba.zwip_odm_ref_value_type | Enter a SQL expression to filter results (use Ctrl+Space)

ref_value_type_id	ref_value_type_name	create_user_id	create_ts	update_user_id	update_ts
111	PROD_TEAM_NAME	-1	2011-08-04 15:59:31.0	-1	2011-08-04 15:59:31.0
29,083	SUB_DIVISION_ID	0	2014-06-25 21:44:33.0	0	2014-06-25 21:44:33.0
4,084	OPERATED_FL_VAL	0	2012-09-29 11:13:34.0	0	2012-09-29 11:13:34.0
16,090	AFE_STATUS	4,151,115	2013-06-13 18:50:03.0	4,151,115	2013-06-13 18:50:03.0
30,087	EQ_DISP_CODE_DESC	-1	2014-07-31 18:06:41.0	-1	2014-07-31 18:06:41.0
3,081	COMMENT_SOURCE	0	2012-08-01 15:07:39.0	0	2012-08-01 15:07:39.0
17,082	GNG_PROSPECT	0	2013-06-12 18:40:04.0	0	2013-06-12 18:40:04.0

3. **Data Processing in MemSQL**: Once the data reaches MemSQL, it is initially stored in a temporary staging table. A stored procedure then manages this data by employing a delete - and - insert strategy to update the target tables. This method is essential for ensuring data consistency and integrity, especially when managing updates and deletions.

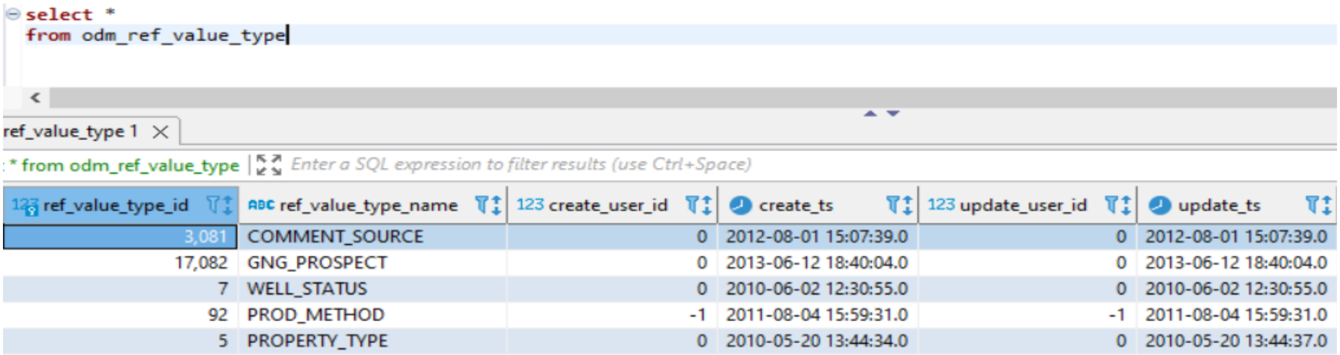
SingleStore Procedure Parameters:

This procedure requires the following parameters to load data into the target table:

1. **pv_schema_name**: Refers to the name of the database where the table resides.
2. **pv_table_name**: Specifies the table to be loaded.
3. **pv_sync_mode**: Indicates the synchronization mode, which can be 'complete' to refresh the entire table or 'incr' for incremental refresh based on a specific condition.

```
CREATE OR REPLACE PROCEDURE `load_o2m_target_table_pipeline`(pn_update_id bigint(20) NULL
, pv_schema_name varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
, pv_table_name varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
, pv_sync_mode varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL) RETURNS void AS
DECLARE

BEGIN

    lv_entry = 'load_o2m_target_table_pipeline';
    lv_subentry = pn_update_id;
    lv_table_name = pv_table_name;
    lv_schema_name = pv_schema_name;
    lv_zwip_table_name = CONCAT('zwip_', pv_table_name);
    lv_sync_mode = pv_sync_mode;

    CALL save_log_prc(ln_app_id, lv_entry, lv_subentry,
    CONCAT('START: SCHEMA NAME --> ',lv_schema_name,' TABLE NAME --> ',lv_table_name,' SYNC MODE --> ',lv_sync_mode), 1);
    lv_sql = CONCAT('SELECT COUNT(1) FROM ', lv_schema_name, '.', lv_zwip_table_name);
    EXECUTE IMMEDIATE lv_sql INTO ln_ingest_count;
    CALL save_log_prc(ln_app_id, lv_entry, lv_subentry, CONCAT('INGEST COUNT:',ln_ingest_count,' SQL: ',lv_sql), 1);

    IF ln_ingest_count > 0 AND lv_sync_mode = 'comp' THEN
        lv_del_sql = CONCAT('DELETE FROM ', lv_schema_name, '.', lv_table_name,' OPTION (columnstore_table_lock_threshold = 1)');
        lv_ins_sql = CONCAT('INSERT INTO ', lv_schema_name, '.', lv_table_name, ' SELECT * FROM ', lv_schema_name, '.', lv_zwip_table_name);

        START TRANSACTION;

        EXECUTE IMMEDIATE lv_del_sql;
        ln_del_count = ROW_COUNT();

        EXECUTE IMMEDIATE lv_ins_sql;
        ln_ins_count = ROW_COUNT();

        CALL save_log_prc(ln_app_id, lv_entry, lv_subentry, CONCAT('DEL:',ln_del_count,' SQL: ',lv_del_sql), 1);
        CALL save_log_prc(ln_app_id, lv_entry, lv_subentry, CONCAT('INS:',ln_ins_count,' SQL: ',lv_ins_sql), 1);
```

This procedure also accounts for situations where the temporary table contains no records. In such cases, it avoids truncating the target table, ensuring that the existing data remains unchanged. This precaution prevents data loss, thereby safeguarding downstream loads from any adverse impacts.



```
select *
from odm_ref_value_type
```

ref_value_type 1

* from odm_ref_value_type | Enter a SQL expression to filter results (use Ctrl+Space)

ref_value_type_id	ref_value_type_name	create_user_id	create_ts	update_user_id	update_ts
3,081	COMMENT_SOURCE	0	2012-08-01 15:07:39.0	0	2012-08-01 15:07:39.0
17,082	GNG_PROSPECT	0	2013-06-12 18:40:04.0	0	2013-06-12 18:40:04.0
7	WELL_STATUS	0	2010-06-02 12:30:55.0	0	2010-06-02 12:30:55.0
92	PROD_METHOD	-1	2011-08-04 15:59:31.0	-1	2011-08-04 15:59:31.0
5	PROPERTY_TYPE	0	2010-05-20 13:44:34.0	0	2010-05-20 13:44:37.0

4. **Orchestration with Airflow**: Apache Airflow orchestrates and schedules the workflow, ensuring each step proceeds in the correct sequence while continuously monitoring for failures. Airflow's comprehensive error handling and retry capabilities significantly boost the reliability of the pipeline.

The DAG outlined manages the processes for each table, as defined in steps 1 - 3 above, and automates these tasks for tables like 'odm_ref_value_type'. Each table follows predefined steps:

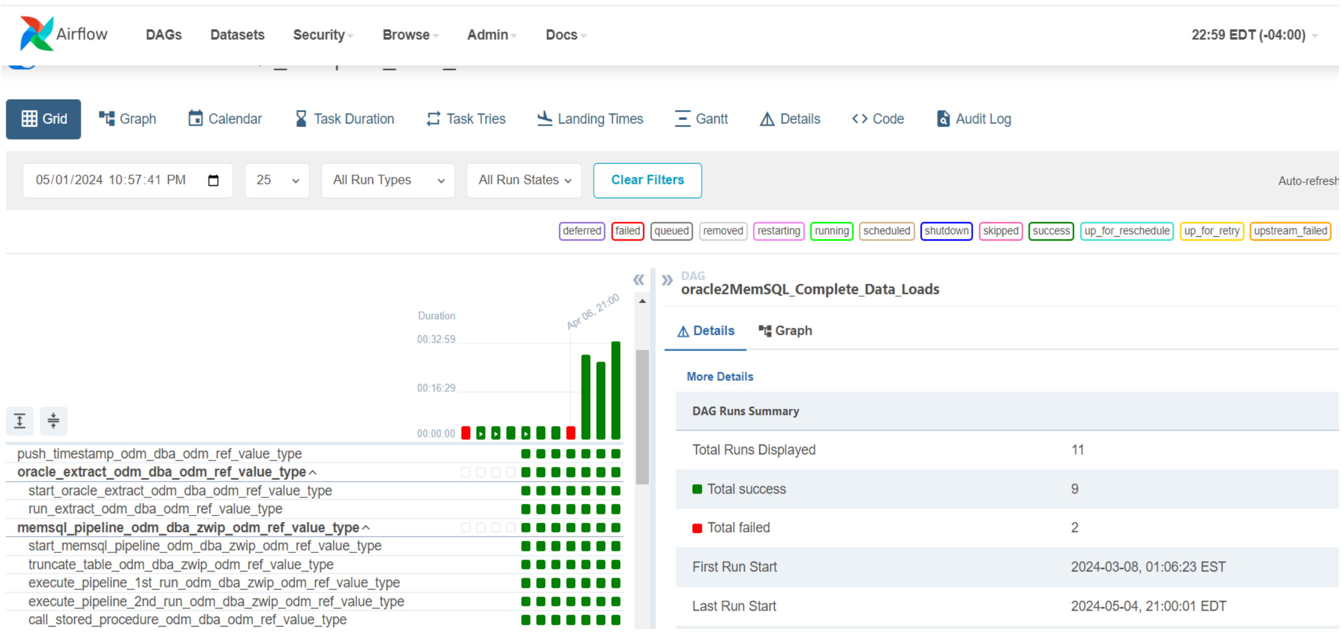
push_timestamp_odm_dba_odm_ref_value_type: Tracks the time when the data load was initiated.

oracle_extract_odm_dba_odm_ref_value_type: Handles the extraction of data from Oracle to Amazon S3.

memsql_pipeline_odm_dba_zwip_odm_ref_value_type: Manages the data loading from Amazon S3 into the temporary table by first truncating it, then employing the SingleStore pipeline to populate the temporary table, and finally using the SingleStore procedure to transfer the data to the target table.

Advantages of using Apache Airflow include:

Apache Airflow offers several compelling advantages for managing data workflows. Its dashboard facilitates easy monitoring, providing detailed logs that simplify the oversight of Directed Acyclic Graphs (DAGs) and troubleshooting of issues. Robust error management features allow for specific remedial actions, including process reruns via a user - friendly web interface, enhancing operational reliability. Airflow also supports high availability by running DAGs from multiple executors; if one fails, another can seamlessly take over, minimizing downtime. Additionally, its flexible task management capabilities enable the use of various operators—be they pre - built or customized by integrating the necessary Python libraries. This flexibility is demonstrated in this project through the use of Bash, Oracle, MySQL, Python, and other operators, streamlining complex data integration tasks. These features make Apache Airflow an invaluable tool for managing complex data workflows, enhancing operational efficiency and system resilience.



Potential Extended Use Cases:

In addition to just transferring data, this ETL pipeline can help - Incremental Data loads for near real time analytics - Integrating the data from multiple sources so that you get a consistent view of your application and Infrastructure The processed data being fed into Machine learning models thereby making predictive analysis smarter. For instance, a retail company that leverages this pipeline to route sales data from multiple regions for unified reporting and inventory management.

3. Impact

Channeling similar options, the adoption of this Airflow - managed ETL pipeline minimizes manual overhead drastically and increases data availability much faster leading to rapid decisions making as well for operational efficiency. For example, a financial institution could optimize their data aggregation and reporting tactics that would allow the business to react rapidly to market movements.

4. Scope

While the primary application described involves Oracle, S3, and MemSQL, the principles and methodologies can be adapted for other source and target systems in both on - premise and cloud environments. The study's findings encourage broader applicability across different sectors that require efficient data handling solutions.

5. Conclusion

The developed ETL pipeline significantly improves data workflows in cloud native environments by leveraging Apache Airflow, Oracle, Amazon S3, and SingleStore. This robust and scalable solution addresses modern data challenges, and future research could explore further optimizations and extended capabilities for advanced data processing.

References

[1] Maxime Beauchemin, "The Apache Airflow Book", O'Reilly Media, 2021, pp.45 - 70.

[2] Anirudh Kala, "Apache Airflow: A Real - World Guide to Data Pipelines", Packt Publishing, 2020, pp.115 - 140.

[3] Amazon S3 Storage, Available at https: //aws. amazon. com/s3/

[4] Amazon S3 Storage on Premise using outposts, Available at https: //aws. amazon. com/s3/outposts/?whats - new - cards. sort - by=item. additionalFields. postDateTime&whats - new - cards. sort - order=desc

[5] Oracle PL/SQL Functions, Available at https: //docs. oracle. com/en/database/other - databases/timesten/22.1/plsql - developer/pl - sql - procedures - and - functions. html

[6] Oracle Directory concepts, Available at https: //docs. oracle. com/en/database/oracle/oracle - database/19/sqlrf/CREATE - DIRECTORY. html

[7] SingleStore Pipelines, Available at https: //docs. singlestore. com/cloud/load - data/load - data - with - pipelines/pipeline - concepts/

[8] SingleStore Procedures Available at https: //docs. singlestore. com/cloud/reference/sql - reference/procedural - sql - reference/create - procedure/