Application based Data Warehouse

Author: Paul Coward

Version: 1.0

Date: January 17, 2022

DOCUMENT HISTORY			
Author	Description of Changes	Date	Version

DOCUMENT REVIEW		
Position	Name	Date

DOCUMENT APPROVAL			
Position	Name	Signature	Date

DOCUMENT DISTRIBUTION			
Position	Name	Org. Unit	Version

Contents

INTRODUCTION

This document contains the technical details of installing and operating a multi-layer Application based Data Warehouse (ADW).

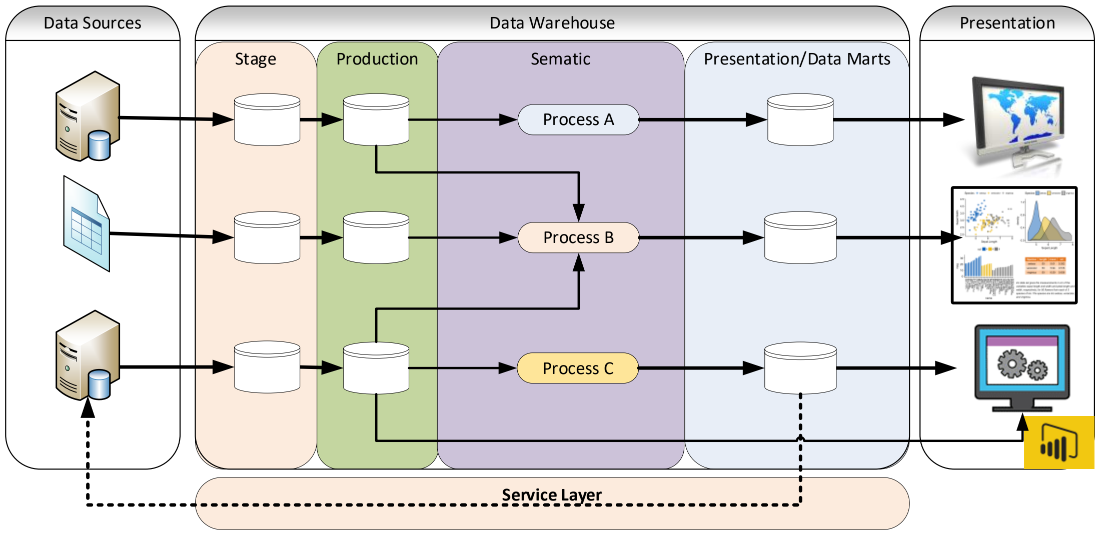
This multi-tier data architecture focuses on scalability, standardization and data reusability. It provides a standardized process to consolidate multiple source datasets into a qualitative dataset that can be leveraged for any purpose.

The ultimate goal of data architecture is creating solutions that deliver value while allowing for future scalability. A multi-layer design combines the benefit of different architectural design standards that are combined with additional methodologies to provide a comprehensive data architecture solution. It integrates relational and dimensional modeling, data quality and validation and the flexibility of providing a scalable solution

ADW is designed around Oracle, the version within this document has been installed and tested on Oracle 19c running on windows 10. All of the tools (PL/SQL and Python) that are developed and used in the installation of ADW are free and in public domain, except for Oracle.

This multi-layer design consists of 5 Data Layers, each data layer in this architecture is dependent upon one more previous data layer to provide the appropriate datasets:

Figure 1: ADW Multi-layer Design



1. Stage

   This is where data is initially EL (Extracted and Loaded) into. An entire application data is staged before it is move to the production area. This is done to ensure that data is completed. Before the move to production the data structure (tables and columns) are verified to ensure that no structural changes to the data has been conducted, which could result in view or processes failing.

2. Production

   This is the production data that can be used by the sematic or presentation layers. When data is moved from the stage to production layer the current production is moved to a backup dataset to ensure that a data used by the presentation layer is not interrupted by the move.

3. Sematic

   This is the T (Translate) in the ETL process. This consists of views, procedures and programs required to create data for either the common data model or presentation layers. The jobs that run in the sematic layer would be scheduled using the Oracle job schedular.

4. Common Data Model (CDM)

   The Common Data Model (CDM) layer, which is not displayed, is optional and would only be used if there was a requirement to maintain a data store within ADW. Basically, the sematic layer would feed data into the CDM layer. The CDM model maybe a star schema or any data model for holding data. Implementation of this layer is optional and will consume a high number of resources to maintain.

5. Presentation or Data Marts

   This layer is created by the sematic layer and consists of data marts that would be available to the presentation tools or other applications. This may contain actual derived data or views into the production data layer through the sematic layer.

1. INSTALLATION

Before we can install any code, we will need to create an oracle schema to hold the warehouse. The schema name must be ***ADWADMIN***. The schema ADWADMIN should have elevated Oracle privileges to create, update all objects within its oracle instant

1.1. ADWADMIN

This is the oracle schema which holds ADW. The account will require access to all objects within the schema.

1.2. Install Scripts

The code will first have to be extracted from GITHUB (**https://github.com/PaulCoward53/ADW**) into a directory (C:\ADW).

There are two installations are required 1) Oracle and 2) Python.

1.2.1 Oracle Install

The oracle installation script that you will need to load into SQL-Developer and execute is ***C:\ADW\ADW_INSTALL\ADW_INSTALL.sql***. The result of the execution will be logged to ***C:\ADW\ADW_INSTALL\ADW_INSTALL.log***

1.2.2 Python Install

You will need to install Python 3.9. It is also recommended that you install a program editor such as VSCode.

You will need to install the following pip packages

```
>pip install cx_oracle
>pip install pyodbc
>pip install pyqt5
```

1.3. Setup Environment

There are a number of routines you need to edit before proceeding to use ADW.

1.3.1. Data Encryption

Edit the python library routine ***c:\ADW\PythonLib\ ADWUtility.py*** to add your own encryption and decryption keys to:

def ADWEncode(inputText,Encodekey=r"Your ADW Key"):

def ADWDecode(encodedText,Decodekey=r"Your ADW Key"):

- Change ***Your ADW Key*** to the desired text

1.3.2. ADW Connection

Run the python program ***c:\ADW\ADW_Connections\ADW_Connections.py.*** Set ADW_PROD to the connection parameters in your environment.

1.3.3. ADW Mail

You will need to modify or create your own process for sending mail for entries within the table **ADW_MAIL_SEND** see the sample code provided (***c:\ADW\ADW_MAIL\ADW_SendMail.py.***)

2. NAMING CONVENTION

Naming standards are an important standard that will assist you in managing all the objects (tables, views, sequences and indexes) within the data warehouse.

All objects within the application data warehouse ***MUST*** be prefixed with an application code stored in the **ADW_APPLICATION** (APP_ID) data table. On setup the APP_ID = 'ADW' is reserved for ADW objects. This means the all objects with the prefix ***"ADW_"*** are use in the administration of ADW.

2.1. TABLES

Tables only need to be prefixed with the **APP_ID**. However, within ADW a suffix is added to show layer table is in:

Table Name	Layer	Purpose
ADW_<name>_S	Stage	Table is used to stage information from ETL process
ADW_<name>_P	Production	Table is used for production as a result of ETL
ADW_<name>_B	Backup	This is the last production version maybe required for backup

2.1.1 Primary Key

A primary Key consist of the table name suffixed by ***"_PK".***

Example: **ADW_NOTIFY_LOG_PK** PRIMARY KEY (NOTIFY_LOG_ID),

2.1.2 Foreign Key

A foreign Key consists of the table name suffixed by ***"_FK#"*** where the # is a numeric value starting a 1 which will make the key name unique

Example: CONSTRAINT **ADW_NOTIFY_LOG_NAME_FK01** FOREIGN KEY (NOTIFY_NAME)
REFERENCES ADW_NOTIFY_LIST(NOTIFY_NAME)

2.1.1 ADW Index

Any index created by the ADW ETL process will be in the following format:

<APP_ID>_#####_IDX

Where the #### is created from the sequence ***ADW_ETL_INDEX_SEQ***

2.1.2 Constraints

A constraint name consists of the table name suffixed by an identify info on the constraint.

Example: CONSTRAINT **ADW_NOTIFY_LIST_ACT_CHK** CHECK (ACTIVE_IND IN ('Y','N'))

2.2. VIEWS

There are two types of views used within ADW.

1. Normal View
   These view names are suffixed by ***_VW*** , the view uses tables, views or materialized tables within ADW.
2. Virtual View
   These view names are suffixed by ***_V*** , the view uses tables, views or materialized tables **NOT** within ADW but uses data directly from the application.

2.3. MATERIALIZED VIEWS

These objects are prefixed with the **APP_ID** and suffixed with ***"_MV".*** It is recommended that a materialized view is created using a view ***(_VW).***

Example: Materialized view ***APP_SAMPLE_MV*** would be created using the view ***APP_SAMPLE_VM***

2.4. FUNCTIONS, PROCEDURES or PACKAGES

These objects only need to be prefixed with the **APP_ID.**

2.5. PACKAGES

These objects only need to be prefixed with the **APP_ID.** All packages will have at least 2 entry points.

1) Function Version
   a. This is a function which returns the package name, Version and date

2) Procedure HELP.
   a. This will print to DMS output the definition of all entry points within the package

3. UTILITIES

3.1. ADW_UTILITY

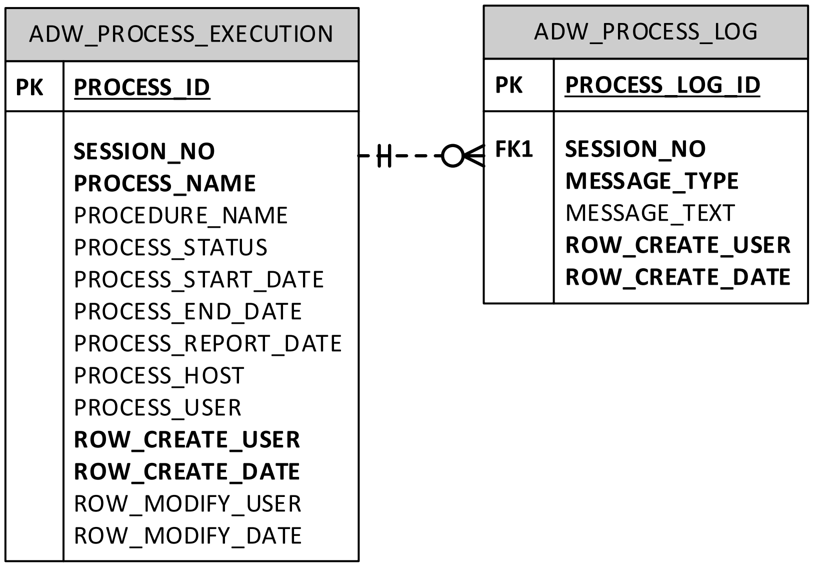
This is a package which contains general PL/SQL utilities used within ADW.

Routine	Purpose
VERSION	This is a function which will return the current version number of the package
GET_DURATION_TIME	This is a function which will return a text of the # hrs # min # sec between two dates.
GET_DURATION_MINUTES	This is a function which will return a number of minutes between two dates.
GET_TABLE_HTML	This is a function which will return the HTML string of the results of a provide query as a table with the field names being the title of each column.
MVIEW_REFRESH	This a procedure which will refresh a materialize view.
REMOVE_APP_STAGE_TABLES	This is a procedure which will remove all STAGE, PRODUCTION and BACKUP tables for a given application. *Once done you **CANNOT** recover these tables.*
BUILD_APP_PROD_VIEWS	This a procedure which will create a view for each table in the production data layer, it will only create views for those missing.
REMOVE_TABLE	This is a procedure which will drop a given table if it exists.
REMOVE_VIEW	This is a procedure which will drop a given view if it exists.
SQL_EXECUTE	This is a procedure which will execute a set of SQL statements each separated by a semi colon “;”.
SLEEP	This is a procedure which will sleep a given number of milli seconds.
HELP	This is a procedure which will print package help to dbms_output

4. PROCESS SERVICE

Within ADW all jobs, including Python based, will log their execution using the process package. All jobs success or fail are recorded within the process service.

4.1. Data Model



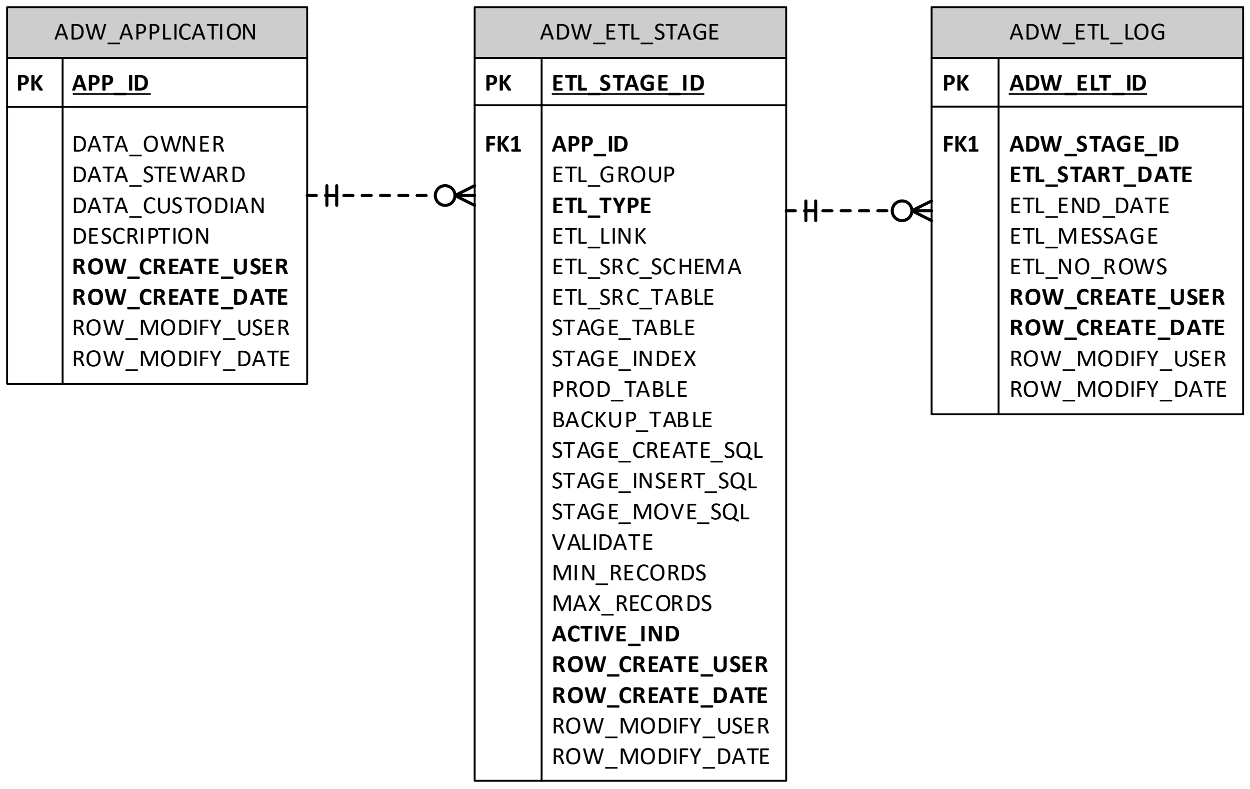
4.2. ADW_PROCESS Package

Routine	Purpose
VERSION	Function which will return the current version number of the package
GET_PROCESS_NAME	Function to return current process name.
GET_SESSION_NO	Function to return current session number.
NEW_SESSION_NO	Generate new session number
CLEAR_SESSION	Clear session number
PROCESS_EXECUTE	Place execution for process as session 0
PROCESS_BEGIN	Begin Process
PROCESS_LOG	Generate log message into log
PROCESS_DEBUG_LOG	Generate debug message into log
PROCESS_ERROR	Generate error message into log
PROCESS_END	To end current process
PROCESS_CLOSE	To close current process
PROCESS_FLUSH	To flush records into log file
HELP	Procedure to output help for the package

5. ETL SERVICE

This module is used to Extract and Load data into ADW. The sources within ADW for data ETL is Oracle, MS/SQL, Teradata and general files. Oracle data loading is done using the PL/SQL package ADW_ETL, most other data sources are managed within Python.

5.1. Data Model



5.2. ADW_ETL Package

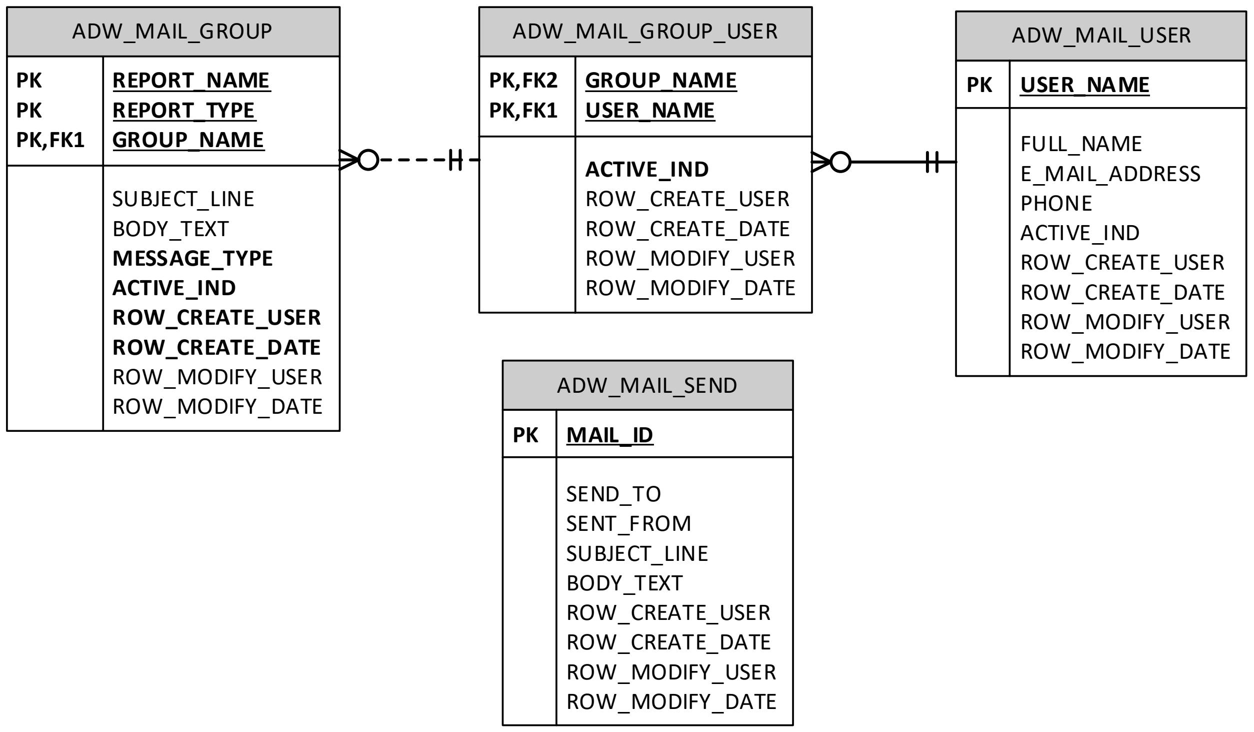
This is an oracle package used to load Oracle data sources into ADW.

Routine	Purpose
VERSION	Function which will return the current version number of the package
LOAD_APP	Procedure to perform Oracle ETL based on specified application.
LOAD_GROUP	Procedure to perform Oracle ETL based on specified group.
HELP	Procedure to output help for the package

6. MAIL SERVICE

This is how ADW sends mail to users. This package is used by all modules and services in ADW to send required information to the administrators and users of ADW.

6.1. Data Model



6.2. ADW_MAIL Package

This package is used to send various mail messages for ADW. Mail is generated and placed into the table ***ADW_MAIL_SEND*** table for the ***ADW_MAIL.py*** python job to pickup and deliver. This module can be changed to use the standard mail service within oracle.

Routine	Purpose
VERSION	Function which will return the current version number of the package
PROCESS	Procedure which will report on a given process in the ADW_PROCESS_EXECUTION table.
MailMessage	Procedure to formulate entry into ADW_MAIL_SEND table
HELP	Procedure to output help for the package

The **BODY_TEXT** is a html text within **ADW_MAIL_GROUP** controls what will be sent. The system will substitute keys words with a given text.

Key Word	Substitute
%LOG%	The output in ADW_PROCESS_LOG for the session
%REPORT_NAME%	The name of the report
%DATE%	The current Date
%CRLF%	Carriage Return/Line Feed (new line)

Sample 1: Simple Log message in BODY_TEXT

```
%REPORT_NAME% generated on %DATE%
%CRLF%
%LOG%
```

Sample 2: Complex with html edits on table in BODY_TEXT

```
<!DOCTYPE html><html><head><style>table, th, td {
  border: 1px solid black;  border-collapse: collapse;}
th, td {  padding: 5px;  text-align: left;}
</style></head><body>
<h2>ADW Error %DATE%%CRLF%</h2>
%REPORT_NAME%%CRLF%
%LOG%
</body>
```

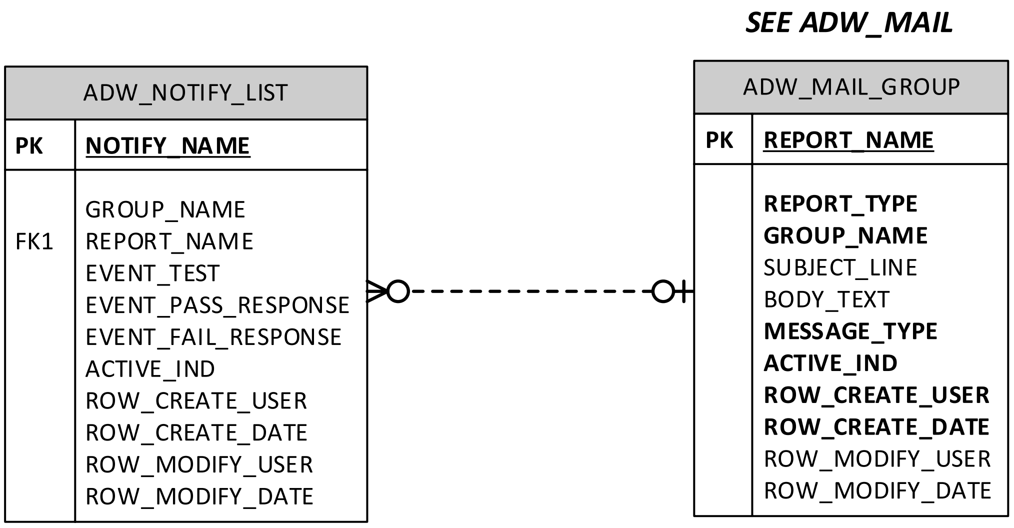
To setup the mail service for you will first need to create a mail group in ***ADW_MAIL_GROUP*** the *group_name = 'ALL'* would apply to all mail messages sent. Within the group you need to define the *message_type* there are 2 message types, N-Normal or E-Error or Exception, which is used by PROCESS to determine what type of session status to mail. This was done so a user would only see errored responses (mail based on exception or failure).

7. NOTIFY SERVICE

This service is used to send application event notifications to users. You need to be careful that once you notify the user of an event that you do not keep notifying them. The event is determined if the sql in EVENT_TEST returns one or more rows. There can be a number of SQL test each separated by a semi-colon ***";".***

If the result of the **EVENT_TEST** yields one or more rows then the user will be sent the **EVENT_PASS_RESPONSE** (if not null) otherwise the **EVENT_FAIL_RESPONSE** (if not null) will be sent.

7.1. Data Model



7.2. ADW_NOTIFY Package

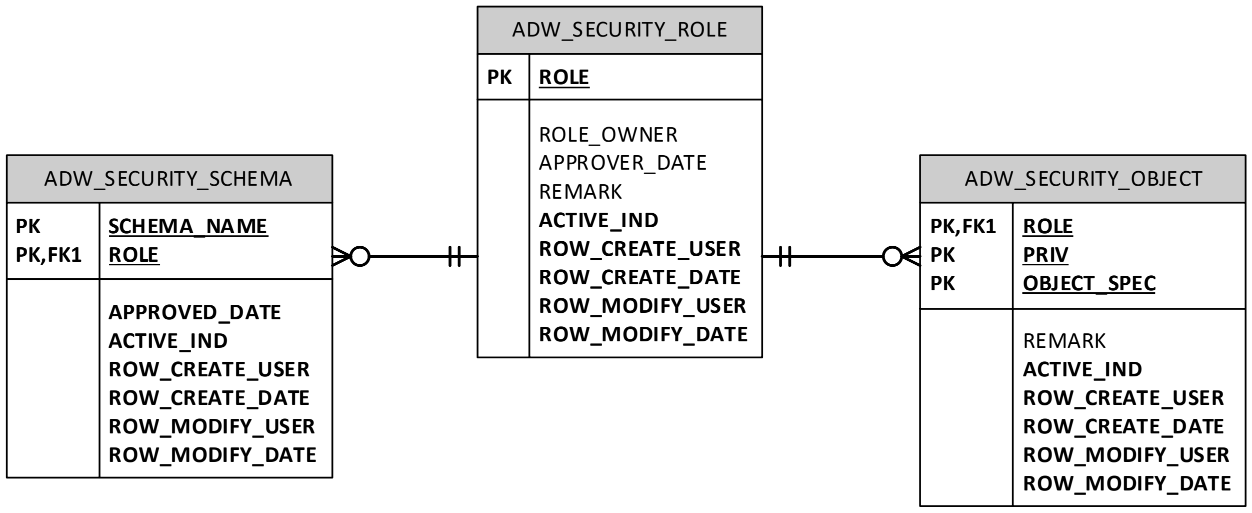
This is a SQL package to conduct a notification service.

Routine	Purpose
VERSION	Function which will return the current version number of the package
EVENT_TEST	Procedure which will perform notification based on the NOTFIY_NAME.
GROUP_TEST	Procedure which will perform notification based on the GROUP_NAME.
HELP	Procedure to output help for the package

8. SECURITY SERVICE

This service handles who can see or edit tables within ADW. You can use wild cards so that you can grant select to all tables of a given application to a set of users within a role.

8.1. Data Model



8.2. ADW_SECURITY Package

This is a SQL package to applies security rules to oracle objects with ADW.

Routine	Purpose
VERSION	Function which will return the current version number of the package
SET_ACCESS	Procedure which grants and/or revokes for a given schema.
UPDATE_ETL_ACCESS	Procedure which updates grants based on ETL Stage file (revoke on backup tables)
HELP	Procedure to output help for the package

9. PYTHON JOBS

9.1. Python Library

The python library is a set of python general purpose utilities. These program are assumes to be located at *"c:/ ADW/PythonLib"*. Each routine that uses this library will have to have the sys path set correctly.

9.1.1. ADWUtility

These are a set of general-purpose routines that will be helpful in python development.

Routine	Purpose
ADWEncode	Function to encode a given text using the provided seed key.
ADWDecode	Function to decode a given text using the provided seed key.
RunCommand	Procedure to execute a given windows procedure or comma.
RemoveNonPrintString	Function to remove non printing characters from string
SQLCleanString	Function to clean string from Oracle insert.
SQLConvertNameUpper	Function to convert string to a name upper. (replace space with underscore)
GetPrintTime	Function to convert seconds to string. *'Total 99 hrs 99 min 99.99 sec'*

Example of usage

```
v_Encode=ADWEncode(r"This is a text encoded")
v_Decode=ADWDecode(v_Encode)
print('Encoded: %s' %(v_Encode))
print('Decoded: %s' %(v_Decode))

v_Encode=ADWEncode(r"This is a text encoded",'Different Key')
v_Decode=ADWDecode(v_Encode,'Different Key')
print('Encoded: %s' %(v_Encode))
print('Decoded: %s' %(v_Decode))

for lines in RunCommand("dir"):
   print(lines)
#end for

vString='This is a name Upper conversion'
print('%s converted to upper %s' %(vString,SQLConvertNameUpper(vString)))
```

9.1.2. OracleConnect

This python class provides interface into an oracle database from python.

Routine	Purpose
OracleConnect	Defines oracle connect class.
cursor	Returns an oracle cursor using the connection
commit	Performs a commit on the connected database
close	Performs a close on the connected database
processBegin	Executes procedure ADW_PROCESS.PROCESS_BEGIN
processLog	Executes procedure ADW_PROCESS.PROCESS_LOG
processError	Executes procedure ADW_PROCESS.PROCESS_ERROR
processEnd	Executes procedure ADW_PROCESS.PROCESS_END
processExecute	Executes procedure ADW_PROCESS.PROCESS_EXECUTE

Example:

```
cur =OracleConnect("ADW_PROD")
cur.processExecute("JOB")

cur.processBegin('test Cursor')
cur.processLog('test log message)
cur.processEnd()

cur.commit()

cur.processBegin(“test Error”)
cur.processError('Generate Error’)
cur.processEnd()

testCur=cur.cursor()
testCur.callproc('ADW_PROCESS.PROCESS_EXECUTE', [‘JOB','PROC','host','User'])
```

9.1.3. MSSQLConnect

This python class provides interface into an SQL Server database from python.

* Uses ADWDecode within ADWUtility.

Routine	Purpose
MSSQLConnect	Defines SQL Server connect class. Default config file *"c:\ADW\ADW_Connections.ini"*
cursor	Returns an sql server cursor using the connection
commit	Performs a commit on the connected database
close	Performs a close on the connected database

Example

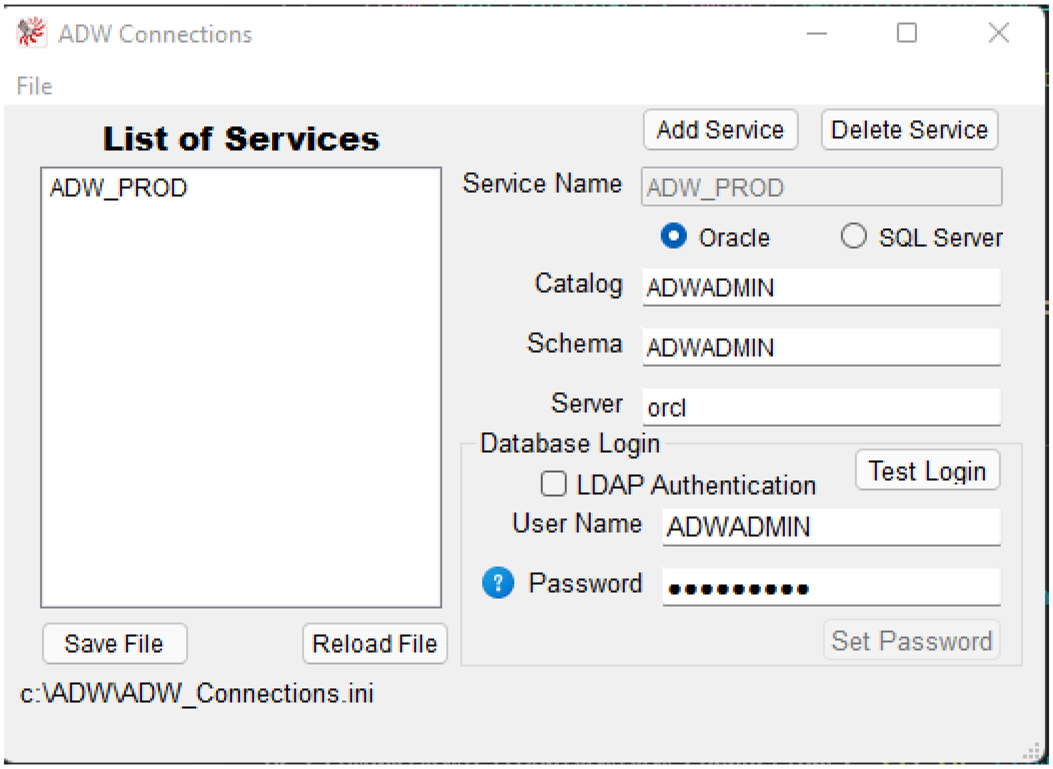
```
cur =MSSQLConnect("MSAP_PROD")
cur.commit()
testCur=cur.cursor()
sqlstr = "SELECT COLUMN_NAME,DATA_TYPE FROM INFORMATION_SCHEMA.columns WHERE TABLE_NAME = 'Project'"

TableColResult = testCur.execute(sqlstr).fetchall()

g_ColumnName = []
g_ColumnType = []
for r_ColumnName,r_ColumnType in TableColResult:
    g_ColumnName.append(r_ColumnName)
    g_ColumnType.append(r_ColumnType.upper())
#END FOR

 if len(g_ColumnName) < 1:
    print (sqlstr)
    print ('-- No Columns in table')
#END IF

v_sqlFields = ''
v_sqlType   = ''
v_sep = ' '
for ii in range(len(g_ColumnName)):
    v_sqlFields = v_sqlFields + v_sep + '"' + g_ColumnName[ii] + '"'
    v_sep = ','
#END FOR
print(v_sqlFields)
```

9.2. ADW_Connections

This python is a python program which allows you to control the settings within the connection file used by other python jobs to make database connections.

*Assumes python lib is a c:\ADW\pythonLib uses routines ADWDecode and ADWEncode

* Uses pyqt5 for GUI and needs ADWCWindow.py in home directory.



ADW_PROD

Service Name: ADW_PROD

[✓ Oracle] SQL Server

Catalog: ADWADMIN

Schema: ADWADMIN

Server: orcl

Database Login

☐ LDAP Authentication

User Name: ADWADMIN

Password: ●●●●●●●●●

c:\ADW\ADW_Connections.ini

Each service is given a unique service name. It is recommended that you name the service prefixed with the application ID. For example: ADW_PROD is ADW production service and ADW_TEST would be for an ADW test service.

The service type can be one of 2 types (Oracle or SQL Server). The Catalog, Schema and Server define the connection. The type of database login can be USER/Password or LDAP Authentication. Once you define the connection you can Test the Login by pressing the Test Login button.

There is a special feature when you change the password for a given user you can press the Set Password button which will set the password for given user in all defined connections.

Once completed you need to press the Save File button to save to the connection file.

9.3. ADW_SendMail

This a python program which will us the ADW_PROD connection to the Application Data Warehouse to send the entries in **ADW_MAIL_SEND** table to the desired person. Once sent the entry will be removed from the table.

This program will have to be modified to satisfy your internal mailing process.