An Oracle White Paper
October 2013

Oracle Data Miner (Extension of SQL Developer 4.0)
Generate a PL/SQL script for workflow deployment

Denny Wong
Oracle Data Mining Technologies
10 Van de Graff Drive
Burlington, MA 01803
USA
denny.wong@oracle.com
# Contents

- Introduction .................................................................................................................. 3
- Importing the Workflow ................................................................................................. 3
  - Create the User ............................................................................................................ 3
  - Load the Table ........................................................................................................... 3
  - Import the Workflow ................................................................................................. 4
- Workflow Overview ........................................................................................................ 5
- Modeling .......................................................................................................................... 5
- Scoring ............................................................................................................................ 6
- Deployment Use Case ...................................................................................................... 6
- Workflow Run .................................................................................................................. 6
- Generating Workflow Script Files ................................................................................ 7
  - Deploy Script Options ............................................................................................... 7
  - Generate Script UI Wizard ....................................................................................... 8
- Script Files Specifications .............................................................................................. 11
  - Master Script: `<Workflow name>_Run.sql` .............................................................. 11
  - Cleanup script: `<Workflow name>_Drop.sql` ......................................................... 11
  - Workflow Image: `<Workflow name>.png` .............................................................. 11
  - Node Script: `<Node name>.sql` ............................................................................. 11
- Running Workflow Script Files ..................................................................................... 12
  - Variable Definitions .................................................................................................. 12
- Control Table .................................................................................................................. 13
- Table Structure ............................................................................................................. 13
  - Column Descriptions ............................................................................................... 14
- Scheduling Workflow Script Files ................................................................................ 17
  - SQL Developer .......................................................................................................... 17
  - Oracle Enterprise Manager Jobs ............................................................................... 24
- Conclusions ..................................................................................................................... 29
- Appendix ......................................................................................................................... 30
**Introduction**

Integrating data mining with an end user application has been one of the more challenging assignments for developers of advanced analytics. With Data Miner 4.0, this effort has been made considerably easier to accomplish. Data analysts and application developers can now work together to develop advanced analytics and easily imbed them into applications.

Data analysts can use the Data Miner workflows to define, build and test their analytic methodologies. When they are satisfied with the results, they can use the new script generation feature to hand off a set of SQL scripts to the application developer. The application developer will be able to take these standard Oracle SQL scripts and integrate them easily into their applications. Since all of the results of Oracle Data Miner are database objects, there is no concern about moving data and models from non Oracle database systems.

Although the primary audience for this paper are application developers; data analysts, database administrators and IT management can benefit from reading it as well.

**Objectives/Benefits:**

- **Application Developers**
  - Describes the structure of the generated scripts and run time behavior.
  - Shows how the generated script can be scheduled to run in the database.
- **Data Analysts**
  - Describes how to generate a SQL script for all or part of a workflow.
- **Database Administrators and IT Management**
  - Provides an understanding of the entire process and how it all runs within the Oracle Database environment.

The business benefit will be in the ability to quickly integrate advanced analytics into existing applications without incurring the high cost and complexity of integrating additional non-database platforms.

**Importing the Workflow**

**Create the User**

This white paper demo requires a Data Miner installation and a Data Miner user account. Please refer to the [Oracle By Example Tutorials](#) to review how to create a Data Miner user account and install the Data Miner Repository.

**Load the Table**

The example in this white paper involves a demo database table INSUR_CUST_LTV_SAMPLE, which comes with the Data Miner Repository installation by default. If it is not loaded in your user account, you can install it manually using the `instInsurCustData.sql` SQL script (available in the companion .zip download for this white paper).
The `instInsurCustData.sql` SQL script can be invoked in the SQL Worksheet of SQL Developer:

```sql
@instInsurCustData.sql
```

Make sure that the script is run from within the user schema.

**Import the Workflow**

The Oracle Data Miner interface provides a mechanism for exporting and importing workflows that capture complex data mining methodologies. The next step in this example is to create a project and import a pre-defined workflow into existing Data Miner user account. The workflow, `codegen_workflow.xml` (available in the companion .zip download for this white paper) will be imported into the user account as shown below.

To create the project and import the workflow, right-click in the Data Miner navigator as shown in the two images below. The first step will create a new project, and the second will perform the import of the pre-defined workflow, `codegen_workflow.xml`. 
Once imported, the workflow should look like the picture below:

**Workflow Overview**

The workflow is comprised of two distinct processes contained within a single lineage: modeling (top) and scoring (bottom). Both processes use the demo data INSUR_CUST_LTV_SAMPLE as input data source.

**Modeling**

The modeling process is used for building a classification SVM model to predict whether the customer will buy insurance or not. The model coefficients are persisted to a database table for viewing and this table may provide a basis for application integration.
Scoring
The scoring process is used to make predictions for the customer data using the SVM model created by the modeling lineage. The prediction result is persisted to a database view for viewing. This view always reflects the predictions of the current input data. For example, if the input table is refreshed with new data, this view will automatically capture the predictions of the new data. Moreover, this view may provide a basis for application integration.

Deployment Use Case
For this demo, we assume a Data Analyst defines the above workflow for model build and score purposes. When he is satisfied with the results, he then uses the new script generation feature to hand off a set of SQL scripts to the Application Developer for deployment. The Developer deploys the scripts to the target or production database, where he can schedule the scripts to run periodically, so that the model will be rebuilt with fresh customer data. In this demo, you will see how to use both SQL Developer/Oracle Scheduler and Oracle Enterprise Manager to schedule the scripts to run on the database. Although not described in this paper, the output results generated by the scripts can easily be integrated into end user reporting applications like Oracle Business Intelligence Enterprise Edition. The output results are described below to give you some ideas on what insights they could provide for the end user.

- MODEL_COEFFICIENTS – underlying database table can be queried to provide insight on what predictors are used to determine whether the user will buy insurance or not.
- SCORED_CUSTOMERS – underlying database view can be queried to display whether a customer will purchase insurance or not, and the likelihood of the prediction. This view always reflects the real time prediction of current customer data.

Workflow Run
The workflow must be run before it can be deployed. Right-click the INSUR_CUST_LTV_SAMPLE BUILD node and select Force Run | Selected Node and Children from the menu to run the whole workflow.
Once workflow run complete, the workflow should look like the picture below:

Generating Workflow Script Files
The next step in this demo is to generate a SQL script from the workflow.

Deploy Script Options
There are three deploy options of how the script can be generated:

- Selected node and dependent nodes
  - Generate script for the selected node(s) and all its parent nodes
  - For example, if the Apply node is selected, a script will be generated for these nodes: INSUR_CUST_LTV_SAMPLE_BUILD, Class Build, INSUR_CUST_LTV_SAMPLE APPLY, and Apply.

- Selected node, dependent nodes, and children nodes
  - Generate script for the selected node(s) and all its parent and children nodes
For example, if the Apply node is selected, a script will be generated for these nodes: INSUR_CUST_LTV_SAMPLE BUILD, Class Build, INSUR_CUST_LTV_SAMPLE APPLY, Apply, and SCORED_CUSTOMERS.

- **Selected node and connected nodes**
  - Generate script for the selected node(s) and all nodes that are connected to this node.
  - For example, if the Apply node is selected, a script will be generated for all nodes in the workflow.

Alternatively, to generate a script for the entire workflow, you can multi-select all nodes in the workflow and select any of above options from the selected node.

To generate a script for the demo workflow, right-click the INSUR_CUST_LTV_SAMPLE BUILD node and select the Deploy | Selected node and connected nodes item from the menu.

**Generate Script UI Wizard**

After you select the deploy option, the Generate SQL Script wizard launches. You must select Target Database Version, so that the generated script will be compatible for that database version. If the workflow nodes included in the script operation contain features that are not compatible with the target database, a warning dialog will be displayed along with unsupported feature information.
Next you need to specify the Script Directory where the scripts will be saved. The Script Directory name defaults to the workflow name. A new directory will be created to store all the generated scripts.
Click Finish to generate the script.
Script Files Specifications
The generated script consists of the following script files:

Master Script: `<Workflow name>_Run.sql`
The Master Script invokes all the required node level scripts in the appropriate order. It performs some validation, such as testing to determine if the version of the script is compatible with the version of the Data Miner Repository that is installed. It creates a workflow master table that contains entries for all the underlying objects created by the workflow script.

Cleanup script: `<Workflow name>_Drop.sql`
The Cleanup Script drops all objects created by running the master script. It drops hidden objects (such as the table generated for Explore Data) as well as public objects (such as Model Names created by Build nodes, and Tables created by a Create Table node).

Workflow Image: `<Workflow name>.png`
An image (.png file) of the workflow at the time the script was generated. The entire workflow is displayed.

Node Script: `<Node name>.sql`
A Node Script performs node specific operation (such as model creation in Build node). A node script is generated for each node that participates in the script generation. The node script should be invoked by the master script only; it should not be invoked directly. Refer to the Appendix for output result for each node type.

For the demo workflow, the following script files are generated:

<table>
<thead>
<tr>
<th>Name</th>
<th>Date modified</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply.sql</td>
<td>10/3/2013 10:10 AM</td>
<td>SQL File</td>
<td>3 KB</td>
</tr>
<tr>
<td>Class Build.sql</td>
<td>10/3/2013 10:10 AM</td>
<td>SQL File</td>
<td>31 KB</td>
</tr>
<tr>
<td>codegen_workflow.png</td>
<td>10/3/2013 10:10 AM</td>
<td>PNG File</td>
<td>21 KB</td>
</tr>
<tr>
<td>codegen_workflow_Drop.sql</td>
<td>10/3/2013 10:10 AM</td>
<td>SQL File</td>
<td>3 KB</td>
</tr>
<tr>
<td>codegen_workflow_Run.sql</td>
<td>10/3/2013 10:10 AM</td>
<td>SQL File</td>
<td>6 KB</td>
</tr>
<tr>
<td>Coefficients.sql</td>
<td>10/3/2013 10:10 AM</td>
<td>SQL File</td>
<td>5 KB</td>
</tr>
<tr>
<td>INSUR_CUST_LTV_SAMPLE APPLY.sql</td>
<td>10/3/2013 10:10 AM</td>
<td>SQL File</td>
<td>4 KB</td>
</tr>
<tr>
<td>INSUR_CUST_LTV_SAMPLE BUILD.sql</td>
<td>10/3/2013 10:10 AM</td>
<td>SQL File</td>
<td>4 KB</td>
</tr>
<tr>
<td>MODEL_COEFFICIENTS.sql</td>
<td>10/3/2013 10:10 AM</td>
<td>SQL File</td>
<td>4 KB</td>
</tr>
<tr>
<td>SCORED_CUSTOMERS.sql</td>
<td>10/3/2013 10:10 AM</td>
<td>SQL File</td>
<td>3 KB</td>
</tr>
</tbody>
</table>

Master script: codegen_workflow_Run.sql.
Cleanup script: codegen_workflow_Drop.sql.
Workflow image: codegen_workflow.png.
Node scripts: Apply.sql, Class Build.sql, Coefficients.sql, INSUR_CUST_LTV_SAMPLE APPLY.sql, INSUR_CUST_LTV_SAMPLE BUILD.sql, and MODEL_COEFFICIENTS.sql, and SCORED_CUSTOMERS.sql.
Running Workflow Script Files

Variable Definitions
The Node level scripts have variable definitions that provide object names for the public objects created by the scripts. The Master script is responsible for invoking all the underlying node level scripts in proper order, so all variable definitions must be defined in the Master script.

The following variables are supported:

- Variables that allows you to change the name of the object names that are input to the Node level scripts, such as tables/views and models. By default, these names are the original table/view and model names.
- Variable that allows you to change the name of the Control table (see below). By default, this name is the workflow name.
- Variable that indicates if named objects should be deleted first before they are generated by the script.

The following object types can be named using these definitions:

- Models referenced in Build Nodes or in a Model Node
- Tables referenced in Create Table or Update Table Nodes

For the demo workflow, the following variable definitions are generated in the Master script:

```
-- Substitution Variable Definition Section: Override default object names here
DEFINE WORKFLOW_OUTPUT = 'codegen_workflow'
-- Drop user named objects (e.g. model, output table)?  TRUE or FALSE
DEFINE DROP_EXISTING_OBJECTS = 'TRUE'
-- From Node: "INSUR_CUST_LTV_SAMPLE BUILD"
DEFINE DATA_SOURCE_1 = "DMUSER"."INSUR_CUST_LTV_SAMPLE"
-- From Node: "INSUR_CUST_LTV_SAMPLE APPLY"
DEFINE DATA_SOURCE_2 = "DMUSER"."INSUR_CUST_LTV_SAMPLE"
-- From Node: "Class Build"
DEFINE MODEL_1 = "DMUSER"."CLAS_SVM_MODEL"
-- From Node: "MODEL_COEFFCIENTS"
DEFINE CREATE_TABLE_2 = "DMUSER"."MODEL_COEFFCIENTS"
-- From Node: "SCORED_CUSTOMERS"
DEFINE CREATE_VIEW_3 = "DMUSER"."SCORED_CUSTOMERS_V"
```

The Control table name variable, WORKFLOW_OUTPUT defaults to the workflow name. The drop variable, DROP_EXISTING_OBJECTS defaults to TRUE, which indicates all existing named objects should be removed before they are generated. The Node specific object variables default to their original names in the workflow. For example, the generated Node variable DATA_SOURCE_1 allows user to override the input table used in the INSUR_CUST_LTV_SAMPLE BUILD Data Source node. It is expected
that the input data sources have the same columns as the originally referenced input data sources. Missing column names could result in a run time failure.

Control Table
When the Master script is run, it first creates the Control table using the name specified in the control table name variable. The purposes of the Control table are the followings:

- Generated objects, such as views, models, text specifications, are registered in this table.
- Logical nodes in the workflow are able to look up their input objects and register their output objects in the control table.
- The Cleanup script uses the table to determine what objects need to be dropped.
- For advanced users, the Control table provides internal name of objects that are not readily accessible via the workflows today. For example, users can find the model test result tables by viewing the Control table.
- By using different control file names along with different output variable names, the generated script can be used concurrently to generate and manage different results. This may be useful if the input data sources continue different sets of data that you wish to mine independently. In this use case, the application would be responsible for saving the name of the control table so that it can be utilized when rerunning or dropping the generated results.

Table Structure
The Control table is defined as following structure:

```sql
CREATE TABLE "&WORKFLOW_OUTPUT"

( NODE_ID VARCHAR2(30) NOT NULL,
  NODE_NAME VARCHAR2(30) NOT NULL,
  NODE_TYPE VARCHAR2(30) NOT NULL,
  MODEL_ID VARCHAR2(30),
  MODEL_NAME VARCHAR2(65),
  MODEL_TYPE VARCHAR2(35),
  OUTPUT_NAME VARCHAR2(30) NOT NULL,
  OUTPUT_TYPE VARCHAR2(30) NOT NULL,
  ADDITIONAL_INFO VARCHAR2(65),
  CREATION_TIME TIMESTAMP(6) NOT NULL,
  COMMENTS VARCHAR2(4000 CHAR)
)
```
## Column Descriptions

The followings describe how the columns in the table are used:

<table>
<thead>
<tr>
<th><strong>Column Name</strong></th>
<th><strong>Description</strong></th>
<th><strong>Examples</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>NODE_ID</td>
<td>Workflow Node id. This uniquely identifies each node.</td>
<td>10001, 10002</td>
</tr>
<tr>
<td>NODE_NAME</td>
<td>Workflow Node name.</td>
<td>Class Build, MINING_DATA_BUILD_V</td>
</tr>
<tr>
<td>NODE_TYPE</td>
<td>Node type is the category of node type that the node is based on.</td>
<td>Data Source, Class Build</td>
</tr>
<tr>
<td>MODEL_ID</td>
<td>Workflow Model id. This uniquely identifies each model referenced within a workflow.</td>
<td>10101, 10102</td>
</tr>
<tr>
<td>MODEL_NAME</td>
<td>Name of Model.</td>
<td>CLAS_GLM_1_6</td>
</tr>
<tr>
<td>MODEL_TYPE</td>
<td>Model type is the algorithm type used by the model.</td>
<td>Generalized Linear Model</td>
</tr>
<tr>
<td>OUTPUT_NAME</td>
<td>The name of the output. These will be internally generated names unless the names are under the control of the user.</td>
<td>Table/View Name, Model Name, Text object names Such as: ODMR$15_37_21_839599RMAFRXI - table name &quot;DMUSER&quot;.&quot;CLAS_GLM_1_6&quot; - fully qualified model name</td>
</tr>
<tr>
<td>OUTPUT_TYPE</td>
<td>Qualifies what type of object.</td>
<td>Table, View, Model</td>
</tr>
<tr>
<td>ADDITIONAL_INFO</td>
<td>Used to qualify what the purpose of the object is in context of the script execution.</td>
<td>Target class for test lift result</td>
</tr>
<tr>
<td>CREATION_TIME</td>
<td>Object creation time.</td>
<td>11-DEC-12 03.37.25.935193000 PM (format determined by locale)</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>Comment to qualify the object's role in the context of the script execution.</td>
<td>Output Data (displayed for nodes like Data Source) Data Usage (displayed for the view passed into model build) Weights Setting (displayed for a weights table passed into model build) Build Setting (displayed for a build settings table passed into model build) Model (displayed for a Model object)</td>
</tr>
</tbody>
</table>
To run the deployed workflow, invoke the Master script in the SQL Plus:

```sql
> @ "C:\codegen\codegen workflow\codegen_workflow_Run.sql"
```

For subsequent run, invoke the Cleanup script first to delete previously generated objects, and then run the Master script:

```sql
> @ "C:\codegen\codegen workflow\codegen_workflow_Drop.sql"
> @ "C:\codegen\codegen workflow\codegen_workflow_Run.sql"
```

After the script is run successfully, you can query the Control table to examine the generated objects:

```sql
> select * from "codegen_workflow"
```

For example, the Create Table Node, MODEL_COEFFICIENTS, produced an output table MODEL_COEFFICIENTS that persisted the coefficient data extracted from the generated SVM model.
To examine the coefficient data, you can query the output table:

```sql
> select * from "DMUSER"."MODEL_COEFFICIENTS"
```

<table>
<thead>
<tr>
<th>CLASS</th>
<th>ATTRIBUTE_NAME</th>
<th>ATTRIBUTE_SUBNAME</th>
<th>ATTRIBUTE_VALUE</th>
<th>COEFFICIENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>AGE</td>
<td>(null)</td>
<td>(null)</td>
<td>-0.6179644874673404490</td>
</tr>
<tr>
<td>Yes</td>
<td>(null)</td>
<td>(null)</td>
<td>(null)</td>
<td>-1.97661457868544</td>
</tr>
<tr>
<td>Yes</td>
<td>T_AMOUNT_AUTO_PAYMENTS</td>
<td>(null)</td>
<td>(null)</td>
<td>-1.06031511465694641</td>
</tr>
<tr>
<td>Yes</td>
<td>TIME_AS_CUSTOMER</td>
<td>(null)</td>
<td>(null)</td>
<td>0.210421616984473001</td>
</tr>
<tr>
<td>Yes</td>
<td>TIME_AS_CUSTOMER</td>
<td>(null)</td>
<td>(null)</td>
<td>-6.24519013840785201</td>
</tr>
<tr>
<td>Yes</td>
<td>TIME_AS_CUSTOMER</td>
<td>(null)</td>
<td>(null)</td>
<td>6.23644263635179901</td>
</tr>
<tr>
<td>Yes</td>
<td>TIME_AS_CUSTOMER</td>
<td>(null)</td>
<td>(null)</td>
<td>-5.52070099233518501</td>
</tr>
<tr>
<td>Yes</td>
<td>TIME_AS_CUSTOMER</td>
<td>(null)</td>
<td>(null)</td>
<td>-0.1199442020370146</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>0.11328601998712999</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>-0.252635090667702499</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>-0.28321165989969202</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>0.0785645959947660196</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>-0.4511472910056499</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>0.54130666275650597</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>0.928223121703924</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>-0.41626986234750396</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>-0.55999416209296601</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>0.57940282272203197</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>-0.1588032250872055</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>-0.16888281866678499</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>0.480810139781444</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>-0.141614679210222</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>-0.164202636864256601</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>-0.0154035456959544</td>
</tr>
<tr>
<td>Yes</td>
<td>STATE</td>
<td>(null)</td>
<td>(null)</td>
<td>0.1115766867414519</td>
</tr>
</tbody>
</table>
Scheduling Workflow Script Files
We will use Oracle Database to schedule the SQL script files to run on the target or production database. All generated SQL script files need to be deployed to the target or production database host machine where they are accessible by the database instance, and these files should be stored together in the same directory. In addition, Data Miner repository is required to run the script files because some node scripts make use of repository provided services at runtime. Moreover, Data Miner user account is required to run the script files because it has necessary grants to the repository provided services. In the following sections, we will show you how to use both SQL Developer and Oracle Enterprise Manager web application to schedule the master script to run.

Add full directory path in the master file
A full directory path needs to be added to each node script invocation in the master script, so that these individual node script files can be called by the master script during runtime.

In the master script, codegen_workflow_Run.sql, find the following node script invocations:

```
-- Workflow run
@@"INSUR_CUST_LTV_SAMPLE BUILD.sql";
@@"INSUR_CUST_LTV_SAMPLE APPLY.sql";
@@"Class Build.sql";
@@"Coefficients.sql";
@@"MODEL_COEFFCIENTS.sql";
@@"Apply.sql";
@@"SCORED_CUSTOMERS.sql";
```

Then add the full directory path where the scripts will be stored in the target or production database host machine. In this example, we assume the script files will be deployed to the /home/workspace directory.

```
-- Workflow run
@@"/home/workspace/INSUR_CUST_LTV SAMPLE BUILD.sql"
@@"/home/workspace/INSUR_CUST_LTV SAMPLE APPLY.sql"
@@"/home/workspace/Class Build.sql"
@@"/home/workspace/Coefficients.sql"
@@"/home/workspace/MODEL COEFFCIENTS.sql"
@@"/home/workspace/Apply.sql"
@@"/home/workspace/SCORED_CUSTOMERS.sql"
```

SQL Developer
SQL Developer provides a graphical interface for developers to define Scheduler Jobs. A SQLPlus script job is required to run the SQL script files, but the Script job type is only supported in the 12c database. The Job definition defines the master script invocation as a script file using a full file path. The user can
decide on whether the Job should be run on a schedule or on demand. The Job execution can be monitored within the application. The result will either be a success or a reported failure.

The following system privileges are required for the account that runs the scheduled SQL script files:

- CREATE CREDENTIAL
- CREATE EXTERNAL JOB

A credential is an Oracle Scheduler object that has a user name and password pair stored in a dedicated database object. A SQLPlus script job uses a host credential to authenticate itself with a database instance or the operating system so that the SQLPlus executable can run. In addition, the job may point to a connect credential that contains a database credential, which is used to connect SQLPlus to the database before running the script.

To create credentials, right-click the Credentials item in the Connections navigator as shown below.
First we will create the database host credential, and then the database credential.

Next we will use the Job wizard to define a new Scheduler job. Click the Job item in the Connections navigator to launch the wizard.
In the Job Details step, enter the following Job name and description. Select the Script job type and SQLPlus script type, and enter the full path names of the cleanup and master scripts in the script window. You can specify if the job will be run one time only or repeating. The following shows the job will be scheduled to run on the first Sunday of each month at midnight.
In the Destination step, select the Local database and the credentials created above.
In the Notification step, you can set up an email notification based on the job status. For this to work, email server should be set up in the host machine. For now, we will skip to the Summary step and click Finish to create the job.

Once the job is created, you can monitor it within the SQL Developer. To monitor the job, right-click the job name, CODEGEN WORKFLOW, in the Jobs folder in the Connections navigator to open the job viewer as shown below.
Click the Run Log tab to see the status of the job. Here you can see when the job started running, how long it took to complete, CPU usage, etc. In case of failure, you should see an error message.

To examine the script run result, click the Job Script Output tab in the bottom window, select the result in the OUTPUT column and right-click to bring up the context menu. Then select the Single Row View to bring up the viewer with the script run result.
Oracle Enterprise Manager Jobs

Oracle Enterprise Manager allows DBAs to define Jobs through the application. The Job definition defines the master script invocation as a script file using a full file path. The user can decide on whether the Job should be run on a schedule or on demand. The job execution can be monitored in the application. The result will either be a success or a reported failure.

User needs to log in to the Oracle Enterprise Manager using the DBA account.
Once logged in, use the Jobs link in the Job Activity section at the bottom of the page to launch the Job creation page.

Select the SQL Script type in the Create Job drop-down list and click Go. This will take you to the Create Job page where you can define the new job.
In the General tab, enter the following Job name and description and add a target database where the job will be run.

![Create 'SQL Script' Job](image1)

In the Parameters tab, enter the full path names of the cleanup and master scripts in the SQL Script edit control.

![Create 'SQL Script' Job](image2)
In the Credentials tab, enter the database host credential and the database credential.

In the Schedule tab, you can specify if the job will be run one time only or repeating. The following shows the job will be scheduled to run on the last day of each month at midnight.
In the Access tab, you can set up an email notification based on the job status. For this to work, email server should be set up in the host machine.

Once you have entered all the settings, click Submit to create the job.
Conclusions

This white paper shows how easy it is to deploy Data Miner workflow to the target or production database using the SQL script generation feature. Furthermore, Oracle Enterprise Manager in the target database can be used to schedule the scripts to run in any desirable time interval. Given that Oracle's data mining and advanced analytics operate natively inside the Oracle Database, mining insights and predictions also remain inside the database so can be accessed by SQL queries from OBIEE. Predictive model results can be called interactively using OBIEE reports and dashboards. With Oracle, all the data mining and advanced analysis and results stay inside the database providing a simpler architecture, better security, better scalability and a single source of truth.
# Appendix

The following table describes the objects that are generated by each node script.

<table>
<thead>
<tr>
<th>Script Type</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Table Node</td>
<td>- Creates a table or view that persists the input data</td>
</tr>
<tr>
<td>Data Explore Node</td>
<td>- Creates a table that contains statistical data of the input data</td>
</tr>
<tr>
<td>Aggregate Node</td>
<td>- Creates a view reflecting the output of the node</td>
</tr>
<tr>
<td>Anomaly Detection Query Node (12c)</td>
<td></td>
</tr>
<tr>
<td>Apply Node</td>
<td></td>
</tr>
<tr>
<td>Apply Text Node</td>
<td></td>
</tr>
<tr>
<td>Clustering Query Node (12c)</td>
<td></td>
</tr>
<tr>
<td>Data Source Node</td>
<td></td>
</tr>
<tr>
<td>Feature Extraction Query Node (12c)</td>
<td></td>
</tr>
<tr>
<td>Filter Columns Details Node</td>
<td></td>
</tr>
<tr>
<td>Filter Rows Node</td>
<td></td>
</tr>
<tr>
<td>Join Node</td>
<td></td>
</tr>
<tr>
<td>Model Details Node</td>
<td></td>
</tr>
<tr>
<td>Prediction Query Node (12c)</td>
<td></td>
</tr>
<tr>
<td>Sample Node</td>
<td></td>
</tr>
<tr>
<td>SQL Query Node</td>
<td></td>
</tr>
<tr>
<td>Transform Node</td>
<td></td>
</tr>
<tr>
<td>Filter Columns Node</td>
<td>- Creates a view reflecting the output of the Column Filter node (like other transform type nodes) - If the Attribute Importance setting is specified, a table is generated containing the AI result</td>
</tr>
<tr>
<td>Build Text Node</td>
<td>- For each text transformation the following objects are created - Feature Table - Oracle Text Objects (Policy, Lexer, Stoplist) - Creates a view reflecting the output of the Build Text node. This is essentially the same output as the Apply Text node</td>
</tr>
<tr>
<td>Classification Build Node</td>
<td>- Create a Build and Test (if necessary) data - For each imbedded text transformation (12c), Oracle Text Objects are created (Policy, Lexer, Stoplist) - A model is created for each model build specification - GLM Model Row Diagnostics Table (if row diagnostics turned on) - Each Model Test will have one table generated for each of the following test results - Test Metric - Confusion Matrix - ROC (binary classification only)</td>
</tr>
<tr>
<td>Build Node</td>
<td>Tasks</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Regression Build Node</strong></td>
<td>- Create a Build and Test (if necessary) data</td>
</tr>
<tr>
<td></td>
<td>- For each imbedded text transformation (12c), Oracle Text Objects are created (Policy, Lexer, Stoplist)</td>
</tr>
<tr>
<td></td>
<td>- A model is created for each model build specification</td>
</tr>
<tr>
<td></td>
<td>- GLM Model Row Diagnostics Table (if row diagnostics turned on)</td>
</tr>
<tr>
<td></td>
<td>- Each Model Test will have one table generated for each of the following test results</td>
</tr>
<tr>
<td></td>
<td>- Test Metric</td>
</tr>
<tr>
<td></td>
<td>- Residual Plot data</td>
</tr>
<tr>
<td><strong>Anomaly Detection Build</strong></td>
<td>- For each imbedded text transformation (12c), Oracle Text Objects are created (Policy, Lexer, Stoplist)</td>
</tr>
<tr>
<td><strong>Clustering Build</strong></td>
<td>- A model is created for each model build specification</td>
</tr>
<tr>
<td><strong>Feature Extraction Build</strong></td>
<td>- A model is created for each model build specification</td>
</tr>
<tr>
<td><strong>Association Build</strong></td>
<td>- A model is created for each model build specification</td>
</tr>
<tr>
<td><strong>Test Node (Classification)</strong></td>
<td>- Same specifications as Class Build except no models are created</td>
</tr>
<tr>
<td><strong>Test Node (Regression)</strong></td>
<td>- Same specifications as Regression Build except no models are created</td>
</tr>
<tr>
<td><strong>Model Node</strong></td>
<td>- No code generated</td>
</tr>
<tr>
<td><strong>Text Reference Node</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Graph Node</strong></td>
<td></td>
</tr>
</tbody>
</table>