Oracle® Health Sciences Omics Data Bank

Installation Guide

Release 2.5

E40250-02

June 2013

This document provides instructions to install Oracle Health Sciences Omics Data Bank (ODB) 2.5.

This document contains the following topics:

1 Installing the Schema

This section provides instructions to install the ODB schema.

1.1 Prerequisites

You must have Oracle Database 11.2.0.2 or 11.2.0.3 for installing the ODB schema.

1. ODB 2.5 requires the following three tablespaces:

   a. **ODB_DATA**: This is the default tablespace assigned to the ODB Schema user. Use the following command to create this tablespace:

   Note: Production tablespaces may need logging and flashback.

   Commands listed in this document are sample commands. You may have to change them to suit to your environment.

   See Also: For information on creating a tablespace refer to, CREATE TABLESPACE section of *Oracle® Database SQL Language Reference 11g Release 2* at http://docs.oracle.com/cd/E11882_01/server.112/e17118/statements_7003.htm#i2231734

   ```
   CREATE BIGFILE TABLESPACE "ODB_DATA" DATAFILE
   '/home/oracle/app/oradata/TRC/odb_data01.dbf' SIZE 25M REUSE AUTOEXTEND ON
   NEXT 25600K MAXSIZE 33546240M NOLOGGING EXTENT MANAGEMENT LOCAL SEGMENT
   SPACE MANAGEMENT AUTO;
   ```

   b. **ODB_INDEX**: This tablespace is used to store ODB schema indexes. The ODB schema user will require to be granted access to this tablespace. Use the following command to create this tablespace:

   ```
   CREATE BIGFILE TABLESPACE "ODB_INDEX" DATAFILE
   '/home/oracle/app/oradata/TRC/odb_index01.dbf' SIZE 25M REUSE AUTOEXTEND ON
   ```

```
NEXT 25600K MAXSIZE 33546240M NOLOGGING EXTENT MANAGEMENT LOCAL SEGMENT
SPACE MANAGEMENT AUTO;
```

c. **ODB_LOB**: This tablespace is used to store ODB LOBs. The ODB schema user will require to be granted access to this tablespace. After this tablespace is created, it can be altered to set this data to be compressed. Use the following command to create this tablespace:

```
CREATE SMALLFILE TABLESPACE "ODB_LOB" DATAFILE
'/home/oracle/app/oradata/TRC/odb_lob01.dbf' SIZE 25M REUSE AUTOEXTEND ON
NEXT 25600K MAXSIZE 32767M NOLOGGING EXTENT MANAGEMENT LOCAL SEGMENT SPACE
MANAGEMENT AUTO;
alter tablespace ODB_LOB default compress for all operations;
```

Note: You can give any name to the above tablespaces. For the purpose of description, we have used ODB_DATA, ODB_INDEX and ODB_LOB.

---

2. Connect as a user with administrator privileges and create the following roles in the database before executing master install script:

   - OmicsDatamartUser
   - OmicsDatamartAdmin
   - OmicsDatamartContributor

   To create the preceding roles, execute the following commands:

```
Sql> create role OmicsDatamartUser;

Sql> create role OmicsDatamartAdmin;

Sql> create role OmicsDatamartContributor;
```

3. Create ODB schema user using the ODB_Data tablespace. The schema user should have sufficient privileges on the ODB_INDEX and ODB_LOB tablespaces. Following is an example for creating the ODB schema user:

```
CREATE USER ODB identified by odb default tablespace ODB_DATA quota unlimited
on ODB_INDEX quota unlimited on ODB_LOB
```

4. Connect to the database instance with sysdba user and execute the following grants to the schema user created in previous step.

```
grant CREATE INDEXTYPE to odb_schema_user;
grant CREATE OPERATOR to odb_schema_user;
grant CREATE PROCEDURE to odb_schema_user;
grant CREATE SEQUENCE to odb_schema_user;
grant CREATE SESSION to odb_schema_user;
grant CREATE SYNONYM to odb_schema_user;
grant CREATE ANY TABLE to odb_schema_user;
grant CREATE TRIGGER to odb_schema_user;
grant CREATE TYPE to odb_schema_user;
grant CREATE VIEW to odb_schema_user;
grant create database link to odb_schema_user;
grant create job to odb_schema_user;
grant select any dictionary to odb_schema_user;
grant connect, resource to odb_schema_user;
grant CREATE ANY DIRECTORY to odb_schema_user;
```

```
commit;
```

5. To install the Omics Data Bank schema, you must install Python 2 on the system from where the schema installation would be done. The directory where **python.exe** executable is located, must be added to the path environment variable.

   Note: You must install only Python 2. Python 3 is not supported.

6. If you want to install the Omics Data Bank schema from a Windows system, an Oracle Wallet setup with a connection to the Database server is required.

1.2 Installing the Omics Data Bank Schema

1. Extract the contents of ODB.zip on the system from where you have access to the database.
2. Navigate to odb_install folder on the command prompt.
3. To install ODB schema with the following named parameters, execute `install_ODB.sh` on Linux or `install_ODB.bat` on Windows.

   Note: If running `install_ODB.sh`, then it must be first executed by running the `chmod u+x install_ODB.sh` command.

   - **-db_wallet <Oracle Wallet Name>**: Name of an Oracle Wallet with credentials of the ODB schema user that was created earlier. This is a required parameter on Windows whereas the next two parameters can be used as an alternative on Linux.
   - **-odb schema <ODB Schema User>**: Username of the ODB schema user that was created earlier. This parameter can be used only on Linux as an alternative to the Wallet parameter and in conjunction with the next parameter.
   - **-db_conn <Database Instance Name>**: The SID or the service name of the database instance. This parameter can be used only on Linux as an alternative to the Wallet parameter and in conjunction with the previous parameter.
   - **-index_tablespace <Index Tablespace Name>**: Name of the tablespace to be used for indexes. This is a required parameter.
   - **-lob_tablespace <LOB Tablespace Name>**: Name of the tablespace to be used for LOB. This is a required parameter.
   - **-promoter_offset <Promoter Offset>**: This value is specified at installation time and is stored in the W_EHA_PRODUCT_PROFILE table and used by the W_EHA_PROMOTER_REGION_V view. This is a required parameter.

     The W_EHA_PROMOTER_REGION_V view is a view used to find results located in the promoter region of any view definition. Promoter regions are defined as some region before the first exon of the coding region of the gene.

     Different genes may have annotated promoter regions, and different species may also have different offsets that are common for that specific species (especially when comparing prokaryote and eukaryote species).

The W_EHA_PROMOTER_REGION_V view takes all of this into consideration for each gene queried. This view first checks the W_EHA_GENE_COMPONENT table for any record that has a COMPONENT_TYPE set to "promoter". The EMBL file format allows for "promoter" annotation to be loaded. Also, the users may want to configure specific annotations for known genes that are not currently identified in the downloaded reference.

The "promoter" annotation is always used as the first check for any defined gene. If there is no specific "promoter" annotation for a specific gene, then the view uses the value specified in the PROMOTER_OFFSET field of the corresponding W_EHA_SPECIES table.

Users may configure specific PROMOTER_OFFSET values for each species loaded. There is no value for the W_EHA_SPECIES.PROMOTER_OFFSET field specified in the EMBL format. This field would have to be configured after installation of reference. If "promoter" annotation and species level promoter offset are not defined for a specific gene, then the global PROMOTER_OFFSET value stored in the W_EHA_PRODUCT_PROFILE table is used to calculate the promoter region for a gene definition.

Changing the W_EHA_PRODUCT_PROFILE.PROMOTER_OFFSET field will only have consequences for queries that use the W_EHA_PROMOTER_REGION_V view and for genes that do not have "promoter" annotation and species level promoter offset defined.

Note: W_EHA_PROMOTER_REGION_V is a global view and not defined for each user.

- **-flanking_offset <Flanking_Offset>**: This value is specified at installation time and is used to link result records to specific genes. This is a required parameter. This enables the database to use some offset before and after the gene to link specific result records. Each result table uses a foreign key to the W_EHA_GENE table. This foreign key is used for the different partitioning modes allowed. Because of the sheer volume, queries for result data are recommended to always have specific genes as filter criteria.

  Users may need to query for results upstream and downstream of different genes. It also may be required that different users have different offsets that they are interested in associating with specific genes. Before installation, some consensus value should be determined as the global offset used for all results.

  This will most likely be the largest value required by all users. Once set, all result data will be linked to genes using this FLANKING_OFFSET before and after each gene definition. This may cause one specific result to be linked to multiple genes since the offsets may overlap. This is expected behavior since the queries to find specific results are used as filter criteria for identifying patients.

  There are existing gene definitions already in the reference that overlap as well. Extreme caution is required if there is a need to change the FLANKING_OFFSET value after result data has been loaded. When this value is changed, all new loaded data may be linked to genes using a different offset than the existing data. This will require that the result tables should be re-partitioned using the new offset before any new data is loaded.

There are various ways to re-partition, but each record to be moved should use a distinct query based on the specific location, specimen, file, and value(s) (That is, sequencing results will have specific variant foreign keys, and gene expression will have different hybridization and calculated fields). Any change required in FLANKING_OFFSET should take this into consideration and plan for the re-partitioning of all result tables.

- **-partition mode <Results Partitioning Parameter>**: The ODB install script permits you to specify the result partitioning as either "study" or "gene". This parameter describes two different partitioning frameworks used for all result tables. This is a required parameter.

  You need to analyze your future use of data to choose the correct partitioning framework. Data can be repartitioned later, but it is very time consuming to rebuild the result tables if there is lot of data loaded. Results tables are the largest tables in the entire ODB schema.

  Before installation you must decide which partitioning scheme is best suited for the queries that need to be executed. If every query is based on a specific study (That is, restricting the results to a specific group of patients that are participating in a study) then the study based partitioning is appropriate. If there is only one major study group or if all queries do not need to filter results by a specific study, then the gene based partitioning is appropriate.

  Once a partitioning scheme has been chosen, you must decide on the appropriate hash values used for partitioning. There is a file named "define_partition_hash_gene.sql" for gene based partitioning. There is a file named "define_partition_hash_study.sql" for study based partitioning. Each of these files define hash values used for partition creation. A detailed description of each partitioning scheme follows.

  Study based partitioning uses a primary range interval partition on the foreign key to W_EHA_RSLT_STUDY. There is also a sub partition declared using a hash partition on the foreign key to W_EHA_GENE. The hash value is used to allocate a specific number of partitions whenever a new result is added that has a new STUDY value. The hash values should be some value using a power of 2. The default value in the "define_partition_hash_study.sql" is 128. The value used for each result table should estimate the total number of expected records for each study, and then decide what is the optimum number of records in each partition. Note that the hash value can be modified later, but this may cause some down time to rebuild the result tables. Changing the hash value is described in the programmer's guide.

  There is an implication about what value is chosen for the hash. Oracle database has a limit of 1048575 (1024K - 1). This means that the maximum value for STUDY key cannot exceed the maximum number of partitions divided by the hash value. With the default value of 128, the maximum STUDY value is 8191. Using a larger value reduces the number of studies that can be used in study based partitioning.

  Gene based partitioning uses a primary list partition on the foreign key to W_EHA_GENE. There is also a sub partition declared using a hash partition on the foreign key to W_EHA_GENE. The reason for a primary list partition is to have one partition capture all result records that are not linked to any gene and store in one large partition since these records are rarely queried. The list partition then has a default partition for all other results linked to any gene. The default value for hash is set to 1024 in the "define_partition_hash_gene.sql" file. This value should be changed to reflect the expected number of

records. Care has to be used since some GENE key values may map to the same hash index. So in order to have proper partition size, the value of the hash should reflect the expected number of result records.

- **-db_platform <Database Platform>**: This parameter expects one of the two values "exadata" or "default". If you are installing on Exadata system, set this value as "exadata" and for all other database platforms as "default". This is a required parameter.
- **-install_mode <Install Mode>**: This parameter expects one of the two values "install" or "upgrade". This parameter is optional and defaults to "install". Moreover, in ODB 2.5 the upgrade mode is not supported.
- **-log_file <Log File Name>**: The name of the log file where the installation log is written. If this file already exists, it will be deleted by the installer and created again. This parameter is optional and defaults to "install_ODB.log".
- **-err_file <Error Log File Name>**: The name of the file where the installation errors are logged. If this file already exists, it will be deleted by the installer and created again (only if there is at least one error). This parameter is optional and defaults to "install_ODB.err".

  When the installation script runs, it prompts you for the ODB schema password (on Linux, if not using the Oracle Wallet only). It executes the Python script install_ODB.py and displays the contents of the error log file on the console, if there are errors during installation. If any required parameters are missing, the script displays an error message and a usage hint on the console.

  For example (Linux with Oracle Wallet),

  ```
  sh  install_ODB.sh -db_wallet MYWALLET -index_tablespace TSI -lob_
  tablespace TSL -promoter_offset 200 -flanking_offset 250 -partition_mode
  gene -db_platform default
  ```

  For example (Linux without Oracle Wallet),

  ```
  sh install_ODB.sh -odb_schema ODB -db_conn MYDATABASE -index_tablespace TSI
  -lob_tablespace TSL -promoter_offset 200 -flanking_offset 250 -partition_
  mode gene -db_platform default
  ```

  For example (Linux getting help with the parameters),

  ```
  sh install_ODB.sh -help
  ```

  For example (Windows),

  ```
  install_ODB.bat -db_wallet MYWALLET -index_tablespace TSI -lob_tablespace
  TSL -promoter_offset 200 -flanking_offset 250 -partition_mode gene -db_
  platform default
  ```

  For example (Windows getting help with the parameters),

  ```
  install_ODB.bat -help
  ```

4. If the CDM schema of TRC 2.0.2 will be used to validate specimens, then execute the datasrc_seed_data.sql script to insert the seed data for W_EHA_ DATASOURCE table. This script assumes the CDM schema exists in the same instance as ODB. If there is a different configuration then edit this script with the correct parameters.

Before the script is loaded, a grant is needed on the CDM schema to enable the ODB schema user to access the specimen table used for validation. The following grant must be executed logged in as CDM user. The specific reference to ODB will be replaced with the ODB schema user name created above. As CDM user execute the following SQL:

```
grant select on W_EHA_SPECIMEN_PATIENT_H to ODB;
```

The datasrc_seed_data.sql script expects 4 parameters:

- ODB schema user: Username of the ODB schema user that was created earlier.
- ODB schema user's password: Password of the ODB schema user that was created earlier.
- Database instance name: The SID or the service name of the database instance.
- CDM schema user: Username of the CDM schema from TRC 2.0.2 installation.

  For example,

  ```
  datasrc_seed_data.sql {odb_schema_user} {odb_schema_pass} {DB_INSTANCE_
  NAME} {cdm_schema_name}
  ```

  More details about the W_EHA_DATASOURCE table:

  The W_EHA_DATASOURCE table permits the user to easily implement other types of validation for specimens that may not use the default validation. This table has the following columns that are used for validation:

  – SCHEMA_NAME: This column will be used to construct dynamic SQL to include schema names required for validating specimens.

  – DB_LINK_NAME: This column will be used to construct dynamic SQL to append database link names if the specimen data exists in another database instance.

  – VALIDATION_PROC: This column has the name of a stored procedure that exists in the ODB schema to perform validation.

  By default, installation will create a record in this table to validate specimen information in the CDM schema. There is a default validation stored procedure used to validate specimens in the CDM schema. This validation stored procedure is named ODB_UTIL.VALIDATE_CDM_SPECIMEN. This stored procedure constructs dynamic SQL to query the corresponding tables in CDM schema to validate the specimen information.

  All the result loaders use the validation stored procedure, which first checks the ODB.W_EHA_RSLT_SPECIMEN table to see if the specimen information has already been validated. If the specimen has not been validated, then the validation code will use the information in VALIDATION_PROC to construct a dynamic SQL statement to validate the specimen.

  Additional data source records can be added that perform different validation. The user must create a stored procedure that is loaded into the ODB schema and has the same parameter list as ODB_UTIL.VALIDATE_CDM_SPECIMEN. The declaration for VALIDATE_CDM_SPECIMEN is as follows:

  ```
  function validate_cdm_specimen (
  i_schema_name in varchar2,
  i_db_link in varchar2,
  i_specimen_number in varchar2,
  i_spec_vendor_numb in varchar2
  ```

```
) return number;
```

The validation stored procedure should return NULL if the specimen does not exist. Any user defined stored procedure should be able to handle the passed parameters for schema_name and database link and should use dynamic SQL to validate specimen information. The current ODB_UTIL.VALIDATE_CDM_SPECIMEN stored procedure can be used as a template to validate specimens in other schemas.

Some transformation of the specimen information may also be needed before querying the CDM schema, and the ODB_UTIL.VALIDATE_CDM_SPECIMEN stored procedure can be used as template. An example of transformation may need to strip some suffix from the specimen number parameter in cases where each aliquot is specified in the result files but not in the CDM schema.

The current ODB_UTIL.VALIDATE_CDM_SPECIMENT stored procedure can be copied to a new stored procedure and then add the proper SQL to strip the aliquot suffix. Then a new W_EHA_DATASOURCE record can be added referencing this new stored procedure.

5. If study information needs to be migrated from CDM schema to the ODB schema, there are new scripts required to keep these tables in sync. The W_EHA_RSLT_STUDY table in the ODB schema was modified from 2.0.2 version to store the foreign key to the CDM schema in a separate column since result data may be partitioned using the ODB W_EHA_RSLT_STUDY primary key values. The study tables are kept in sync by the ETL process that updates CDM data. The ETL process calls a stored procedure in the CDM database package named COHORT_PROTOCOL_UTIL.

   There are two versions of this package. One version is used if CDC is installed, and the other version is used if CDC is not installed. If you are unsure about the configuration, load the package without CDC installed first. The CDC version can be loaded later if needed. In all of these steps replace the CDM and ODB references with the current user names existing in the current configuration. The steps to load the package are as follows:

   a. As ODB user grant the following permissions to the CDM user. Replace the CDM and ODB user names with the correct names for the current configuration:

      ```
      grant select on w_eha_rslt_study_s to CDM;
      grant insert, select, update on w_eha_rslt_study to CDM;
      ```

   b. As CDM user create local synonyms to the ODB objects needed by the ETL package:

      ```
      create or replace synonym w_eha_rslt_study for odb.w_eha_rslt_study;
      create or replace synonym w_eha_rslt_study_s for odb.w_eha_rslt_study_s;
      ```

   c. As CDM user load the correct package as needed for the current configuration (that is, with or without CDC support):

      ```
      @cohort_protocol_util_no_cdc.pkb
      ```

      or if CDC is installed:

      ```
      @cohort_protocol_util.pkb
      ```

d. As CDM user execute the stored procedure to trigger an update of the study information to ODB schema:

```
exec cohort_protocol_util.process_protocols;
```

6. Verify all Packages, procedures, and functions are compiled in the ODB Schema. If any object is uncompiled, recompile it.
7. Creating Local Synonym for a Database User:
   - To create a local synonym for a database user, grant appropriate ODB roles to the user first, and then execute the following script connecting as database user.

     ```
     @create_odb_synonyms.sql &&odb_schema_name
     ```

     Note: The database user should have CREATE SYNONYM and CREATE SESSION privileges.

2 Oracle Health Sciences Omics Data Bank Loaders

The loader scripts to load data into ODB are divided into two categories: The reference loader, which loads genomic reference information and result loaders to load patient or specimen specific result data.

2.1 Prerequisites

1. Oracle Client, with SQLPlus, is installed on the system from where the Loader scripts are executed.
2. For EMBL & SwissProt loaders, JRE 1.7 should be installed on the system from where these Loaders are executed.
3. If you plan to run from a Windows system would require an Oracle Wallet setup with a connection to the Database server.
4. For result loaders, ensure the tables W_EHA_RSLT_STUDY and W_EHA_DATASOURCE is populated with at-least one record, pointing to a User defined Study and an installed CDM user data source for specimen lookup.
5. Name of an Oracle Directory Object pointing to the location of input data files folder.

3 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Oracle Health Sciences Omics Data Bank Installation Guide, Release 2.5
E40250-02