Oracle® Life Sciences Data Hub

Implementation Guide

Release 2.4.8

E95828-01

May 2018

This guide introduces Oracle Life Sciences Data Hub and contains information you need to design your Oracle LSH implementation to fill your company's needs.

Oracle Life Sciences Data Hub Implementation Guide, Release 2.4.8

E95828-01

Contents

Preface

This guide introduces Oracle Life Sciences Data Hub (Oracle LSH) and contains information you need to design your Oracle LSH implementation—to set up classification, security, validation, and organizational systems—to fill your company's needs.

Finding More Information

Oracle Help Center

The latest user documentation for Oracle Life Sciences Data Hub is available at `http://docs.oracle.com/en/industries/health-sciences/lsh-248`.

My Oracle Support

The latest release notes, patches and white papers are on My Oracle Support (MOS) at `https://support.oracle.com`. For help with using MOS, see `https://docs.oracle.com/cd/E74665_01/MOSHP/toc.htm`.

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 that have purchased support 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.

1 Overview

This section includes information on the following topics:

About the Oracle Life Sciences Data Hub

The Oracle Life Sciences Data Hub (Oracle LSH) is a powerful and flexible data integration and statistical analysis tool. It is closely integrated with several external tools, notably the Oracle Business Intelligence Suite and SAS and can also be integrated with other tools using the Oracle LSH Adapter Toolkit. Oracle LSH allows you to load and analyze data from diverse systems, including clinical data management and remote data capture systems.

Oracle LSH is designed to help you comply with pharmaceutical industry regulations: it maintains an audit trail of data transactions, stores all programs and tables as database objects under version control, and provides tools for validating these objects and, by extension, the data they touch within Oracle LSH. Using labeled or timestamp-based data snapshots and program versions, you can recreate submitted data.

However, the Oracle Life Sciences Data Hub is not restricted to use with clinical data; you can use it to store and analyze financial, administrative, or any other kind of data as well.

Oracle LSH's functionality covers three basic areas:

- Data Storage

- Data Manipulation
- Information Retrieval

Data Storage Oracle LSH serves as a single repository of data originating in other systems, including Oracle, SAS, or any system from which you can produce a text file. You set up a connection with each source data system and load or refresh the data you specify as necessary.

Through the connection Oracle LSH can read the data sets or tables you use in external SAS or Oracle systems and reproduce them automatically in Oracle LSH before loading data into them. You can continue to use these structures in Oracle LSH or adopt new industry standards such as CDISC and LOINC and move your current data into tables based on standards. If those standards change over time, you can adjust your Oracle LSH tables accordingly.

Oracle LSH maintains data blinds on the tables you specify and allows blind breaks and unblinding with a combination of security privileges.

Oracle LSH maintains an audit trail on data and maintains version control over all programs and other objects that manipulate data. In addition, for each report output produced in Oracle LSH, the system maintains a record of all the programs and other objects that manipulated the data displayed in the report beginning when the source data entered Oracle LSH.

You can use Oracle LSH to integrate data in many ways. For example:

- Create a single data repository for your company and a company you have acquired, even though the acquired company uses a different clinical data management system.
- Compare data from any number of ongoing and closed studies.
- View and analyze data collected in a system that is now obsolete.

Data Manipulation To combine, analyze, and report on the data you load into Oracle LSH, you write programs. These programs can be developed in SAS, Oracle PL/SQL, or Oracle Reports.

Each of these development environments is closely integrated with Oracle LSH. Oracle LSH stores the program code under version control. When you run a program in Oracle LSH, the system sends the job to the appropriate engine for execution. If the program writes data to tables, Oracle LSH tables receive the data. If the program generates a report, you can view the report, as well as the program's log file, through Oracle LSH.

If you are currently using SAS to analyze clinical data, you can upload your programs, macros, and formats and continue using them in Oracle LSH. If you have SAS installed on your personal computer, you can open SAS directly from Oracle LSH and view Oracle LSH source data.

You can combine a series of programs into a single executable process; for example, automatically load fresh data into Oracle LSH at regular intervals, generate a report on the new data after each load, and send an email notification that the report is ready to the appropriate personnel.

Information Retrieval You can retrieve information from the data repository in several ways:

- **Reports**. Write programs to generate reports—including tabulations, figures, and listings—on data.

- **Report Sets**. Create formal sets of reports suitable for submission to regulatory agencies with a single table of contents, configurable page numbering, and hyperlinks. Using Oracle XML Publisher, you can create custom templates for these reports for A4- or US letter-sized paper with graphics and watermarks, and produce a single or multivolume PDF output.
- **Data Visualizations**. Oracle LSH is integrated with Oracle Business Intelligence Enterprise Edition so that you can create ad hoc visualizations of Oracle LSH data in either tool, subject to Oracle LSH security requirements.
- **Data Marts**. To send data out of Oracle LSH—to a partner organization or for longterm storage, for example—create Oracle LSH Data Marts, files of potentially very large amounts of data, in several possible formats including Oracle Export, SAS transport (XPORT and CPORT), SAS data sets, or text files.

Developing Programs and Applications

To support the functionality outlined above, Oracle LSH supports a definitional methodology. Each time you write a program in Oracle LSH you must also *define* the program as an *object*, create an *instance* of the object definition, and *install* it in the database so that it can interact with data. You can reuse a definition by creating multiple instances of it.

Oracle LSH has seven predefined primary object types: Tables (which are equivalent to SAS data sets and Oracle tables), Load Sets, Programs, Workflows, Report Sets, Data Marts, and Business Areas (the basis for data visualizations). Each object type interacts in a predefined way with the internal system to accomplish a particular data handling task as described later in this chapter.

You can develop libraries of standard object definitions that you have validated and declared suitable for reuse. You can then use these library definitions as modular building blocks to accomplish different business tasks, called *applications* in Oracle LSH.

Example To develop an application whose purpose is to produce a set of adverse event reports for Study 01, you could do the following:

- Create a Load Set definition to load the adverse event and demography data sets for Study 01 from your SAS system into Oracle LSH.
- (Optional) Create CDISC-compliant adverse event and demography Oracle LSH Table definitions and a SAS Program definition to write adverse event data into the CDISC adverse event table and demography data into the CDISC demography table.
- Write Program definitions that pull data from the adverse event and demography tables and generate reports (listings, figures, and tables).

You can create instances of the same object definitions in a different Work Area to quickly create the same application for Study 02, and again for Study 03, and so on.

Advantages Although this definitional approach initially requires more work, it has the following advantages:

- **Regulatory Compliance**. Oracle LSH helps your company comply with industry regulations by maintaining version control over all definitional objects, and enabling you to set and meet criteria for validating all objects (see "Validating Object Definitions and Instances" on page 1-10).

Note: You can recreate submitted (or other) data by running the same version of the program(s) that you originally used to generate the submitted data on the data that was current at that time, using a labeled or timestamp-based data snapshot.

- **Reuse**. You can reuse object definitions, reducing work and increasing consistency. You can create new instances of existing object definitions or you can copy object definitions and modify them as necessary (see "Reusing Object Definitions" on page 1-10).
- **Security**. You can develop a security system to control access to, and operations on, objects and outputs,which are stored as objects (see Chapter 4, "Designing a Security System").
- **Classification**. You can develop a classification system and classify objects and outputs accordingly. Classification allows you to label the many objects and outputs created in the course of a clinical trial to make them easier to find through searching and browsing (see Chapter 5, "Designing a Classification System for Searching and Browsing").

The following sections describe each data handling stage and the object type(s) predefined for use in each stage. Further information on object definition is included in Chapter 3, "Designing an Organizational Structure" and in "Applications User Interface" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Loading Data

You can load data into Oracle LSH from a variety of sources:

- **Oracle Databases, including Oracle Clinical**. To load data from any Oracle system, you must enter connection information for the external system into Oracle LSH, including a username and password (encrypted in Oracle LSH) with access to the data in the external system that you want to load, so that Oracle LSH can connect directly to the source data system. You can require user-specific connections or allow the use of shared connections.
- **SAS**. To load data from a SAS system, you must have access to SAS data set, CPort, or XPort files.
- **Text Files**. Oracle LSH also handles loading data from fixed-format or delimited text files, but in this case you must manually define the target table in Oracle LSH before loading data. You must have access to the text file.

Note: You can convert a spreadsheet to a comma-delimited text file and load its data into Oracle LSH.

Definitional Object: Load Set The Oracle LSH definitional object used in loading data is called a Load Set. In addition to loading data, the Load Set reproduces the meta-data of the external Oracle table or SAS data set in Oracle LSH.

Once defined, Load Sets are saved and each subsequent time they are run they update the data in Oracle LSH to reflect changes in the source data. You can schedule a Load Set to run at regular intervals to refresh data.

If the external system is an Oracle database you can define the Load Set as a view against the source table or data set. This means that without actually loading the data into Oracle LSH you can browse the data dynamically in the external system as if it were a database view.

Adapters Each Load Set has a Type attribute setting corresponding to its source data system (or, in the case of Oracle Clinical, the type of data or meta-data being loaded). Each Load Set type is based on a different predefined adapter that includes the set of programs necessary to copy the data structures and load data and determines the parameters you must set.

Adapters serve as the interface between Oracle LSH and each external system, ensuring that Oracle LSH handles the incoming data and meta-data correctly. If you upgrade to a new version of an external system (for example, SAS), you may also need to upgrade the relevant adapter, but you do not need to upgrade Oracle LSH. If you install a new version of Oracle LSH, it includes any required new versions of the predefined adapters.

Oracle LSH includes the following adapters, each of which is designed especially to load data from one type of external system into Oracle LSH:

- **Oracle Database**. Loads data and meta-data from one or more tables in any Oracle database.
- **SAS**. Loads data and meta-data from a SAS file.
- **Text**. Loads data from one text file into an Oracle LSH table you define to match the data structure in the files.
- **Oracle Clinical**. Although you could use the Oracle Database adapter to load data from Oracle Clinical, Oracle LSH includes a set of adapters to handle Oracle Clinical data more intelligently, as follows:
  - **Global Library**. Loads meta-data—Questions, Discrete Value Groups (DVGs), and Question Groups—that you have defined in the Oracle Clinical Global Library and converts them to Oracle LSH Variables, Parameters with LOVs, and Tables.
  - **Data Extract Oracle Views**. Converts the DX Oracle Views you have defined in Oracle Clinical to Oracle LSH Table instances, and either loads their data or serves as a pass-through view.
  - **Data Extract SAS Views**. Converts the DX SAS Views you have defined in Oracle Clinical to Oracle LSH Table instances and loads their data into Oracle LSH.
  - **Study Data**. Loads study-specific, non-patient data such as discrepancies, Data Clarification Forms (DCFs), page tracking and patient status information.
  - **Study Design and Definition**. Loads study-specific meta-data including Data Collection Modules (DCMs) and Data Collection Instruments (DCIs).
  - **Labs**. Loads lab-related data, including Labs, Panels, Units, and Ranges.
  - **Stable Interface Tables**. Loads data from any Oracle Clinical table that is part of the stable interface as described in the *Oracle Clinical Release 4.5 Stable Interface Guide*.
  - **Randomization**. Loads blinded treatment codes and dummy data. Blinded data remains blinded in Oracle LSH.

For further information on Load Sets, see "Defining Load Sets" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Storing Data

Oracle LSH allows you to store data from a variety of external systems in a single repository.

Oracle LSH allows you to blind data and to break the blind or permanently unblind the data with a combination of security privileges. Blinded data you load from Oracle Clinical is automatically blinded in Oracle LSH. For other source data systems, the first time you load data that should be blinded you must specify that the target Oracle LSH table be blinded. See "Security for Blinded Data" on page 4-11 for further information.

Definitional Object: Table Oracle LSH uses a definitional object called a Table to store data. All Oracle LSH Table instances include characteristics of both standard Oracle database tables and SAS data sets. Therefore you can load data from either system into an Oracle LSH Table instance, and you can write a program that combines data from both systems into a single Oracle LSH Table instance.

Oracle LSH can create Tables with the same structure as the Oracle tables or SAS data sets in your source data systems.

If you are not already using industry standards such as CDISC or LOINC, you can define Tables that conform to those standards and write programs in Oracle LSH to move your current data into them.

Work Areas All data storage, manipulation and viewing in Oracle LSH must be performed in Work Areas. In a Work Area you create instances of Load Sets, Programs, Tables, and any other definitional objects that are required to accomplish a particular business purpose, and install the Work Area in the database.

When you install a Work Area for the first time, Oracle LSH creates a database schema and creates a database object from each object instance in the Work Area. Oracle LSH stores the data you load or write to Table instances in the Work Area in the corresponding database tables in the schema.

You can use Work Areas to create your development, quality control, and production environments. When you are ready for formal testing of the Programs and other objects you have created in the development Work Area, you can clone the development Work Area to create the quality control Work Area and install the same objects to a new schema, load fresh data, and run tests. Then clone the quality control Work Area, install it and load production data to create your production Oracle LSH environment. Oracle LSH supports this usage by enforcing a number of rules; see "Work Area Usage Intent and Validation Status" on page 6-5 for further information.

Audit Trail Oracle LSH can store data in a manner that is compliant with industry regulations: with an audit trail of every change to every record.

Whether or not Oracle LSH maintains an audit trail on the records in a particular Table instance depends on the Data Processing type you define for the Table instance. In most cases when a record is deleted it remains in the database associated with an end timestamp and an additional row explicitly recording the deletion. It is therefore possible to reconstruct the data in a given Table instance as it was at any point in time.

Data Processing Oracle LSH can use several different methods to process data internally when you run a Program that writes data to a table. The method used depends on the setting of an attribute of the target Table instance, not the Program, though the Program type must be compatible. The way Oracle LSH handles record deletion and auditing in a table depends on the processing mode specified for the Table instance.

The data processing types available are:

- **Reload**. Reload processing requires a target Table instance with a primary or unique key defined. Each time you run a Load Set or a Program whose target Table instances have the Reload processing type, Oracle LSH processes all data, comparing the primary or unique key of each incoming record with the primary or unique key of existing records. The system updates records that have changed since the last load, inserts new records, and updates the refresh timestamp of all records. Reload processing is always audited. Reload processing is appropriate for use with Load Sets and with SAS Programs that write to tables.

  If you specify Full Reload processing at runtime, the system also soft-deletes all records that are no longer present in the source. If you choose Incremental Reload, the system does not delete any records, making the processing go faster.

- **Transactional**. Transactional processing is appropriate for use with Oracle Reports and PL/SQL Programs. The Program's source code includes explicit Insert, Update, and Delete statements. You can specify whether or not you want the data changes to be audited. If you choose to audit, the system performs soft-deletions only; deleted data remains in the database associated with an end timestamp.

- **Staging**. Staging processing is appropriate for internal use with any Program type. The system deletes, or appears to delete, all data from the Table instance immediately before each Program execution. You can specify whether or not you want to use auditing. If you choose to audit, the system leaves all records in the Table instance, but future executions of the same Program cannot see them. If you choose not to audit, the data is hard-deleted and cannot be reconstructed later.

See "Execution and Data Handling" in the *Oracle Life Sciences Data Hub Application Developer's Guide* for further information.

Snapshots Using the audit information, you can create and name a snapshot recreating the state of data in an Oracle LSH schema at any given point in time. When you run an executable in Oracle LSH, by default it runs on the most current data in the source Table instance(s). However, you can specify a previous data snapshot if you prefer.

Data File Storage Oracle LSH provides two ways to store data in files:

- **Original Source System Data Files**. Oracle LSH places data loaded from files (SAS and text) into the Oracle LSH Table instances specified in the Load Set so that the data can be operated on within Oracle LSH. However, you can choose to also store the original data file in the Oracle LSH database, with classifications and security requirements. This option is available only for Load Sets that load data contained in files.

- **Oracle LSH Data Marts**. You can create a Data Mart containing data from any Oracle LSH Table instances in one of several formats: SAS Transport, Oracle Export, or text. You can transfer a Data Mart to an external system or leave it in Oracle LSH for long-term storage.

Operating on Data

After you load data into Oracle LSH, you can combine, transform and analyze the data as necessary.

Definitional Object: Program To manipulate data in Oracle LSH, you must create a definitional object called a Program.

You can upload SAS or PL/SQL source code you have already developed or write new programs in PL/SQL, Oracle Reports, and SAS (if you purchase SAS). Oracle LSH stores the source code in Oracle LSH under version control. When you run the program, Oracle LSH launches the appropriate engine to run the code.

For further information, see "Defining Programs" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Reporting Data

Three Oracle LSH primary definitional objects can report data. Programs can generate report outputs (tabulations, figures, and listings) directly. Report Sets and Workflows can generate report outputs because they contain Programs.

For information on Workflows, see "Automating Multiple Processes" on page 1-8.

Definitional Object: Program Oracle LSH Programs are the primary means of reporting data as well as transforming data. To create a report in Oracle LSH, you must create or reuse an Oracle LSH Program definition that includes source code to produce the report(s) you want and a Planned Output definition for each report to be generated by the Program. You can view report outputs as files online or print them.

For further information, see "Defining Programs" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Definitional Object: Report Set A Report Set is a definitional object whose purpose is to present a set of reports and text narratives divided into logical chapters and sections, with a single table of contents and hyperlinks. You can use Oracle XML Publisher to create templates with watermarks and graphics and generate a single- or multi-volume integrated PDF file containing all the reports.

For further information, see "Defining Report Sets" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Automating Multiple Processes

Oracle LSH uses the Oracle Workflow product to enable you to link any number of executable objects—Load Sets, Programs, Report Sets, and Data Marts—into a single complex object executed as a whole. You define the sequence of the executables, which may include branching, and the necessary conditions, if any, to pass from one to the next.

Definitional Object: Workflow For example, you can create a Workflow to load data from two systems, wait until both loads are successfully completed, and then run a Program that writes data from both systems into a single set of combined Tables and generates a Report Set on the combined data.

You can also insert Notifications into a Workflow. Notifications are sent to recipients you specify either to simply inform them of something, such as the fact that the Workflow has just generated a report on fresh data, or to request their approval, for example of a load of lab data before the data is included in a statistics analysis report generated later in the Workflow. In the case of approval requests, the Workflow waits for a response. You define the next Workflow step in the case of both an approval and a rejection.

For further information, see "Defining Workflows" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Viewing Data

There are several ways to see data stored in Oracle LSH:

- Reports
- Visualizations
- User Interface
- Integrated Development Environment

Reports Report outputs of Programs display Oracle LSH data. These may be generated directly by an Oracle LSH Program or as part of a Report Set or Workflow (see "Reporting Data" on page 1-8). Anyone with the necessary security privileges can execute a particular Program, Report Set, or Workflow to generate the report on current data, or simply view a previously generated report, either onscreen or printed. For further information, see "Defining Programs" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Visualizations Oracle LSH is designed to integrate with data visualization tools that allow users without programming skills to perform ad hoc, interactive explorations of data in tabular or graphical format.

Oracle LSH includes a user interface for defining Business Areas, Table Descriptors, Joins, and Hierarchies that determine what Oracle LSH data is available to visualizations. Security for the data displayed by visualizations is determined by the security requirements of the corresponding Business Area instance in Oracle LSH.

For further information, see "Defining Business Areas for Visualizations" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

User Interface Through the Oracle LSH user interface you can browse data in a single Table instance; either the current data or a snapshot. You can select the columns you want to view, limit the data displayed with a Where clause and order the display of data with an Order By clause. If you have the required security privileges, you can see real, blinded data if you choose to. For further information, see "Common Development Tasks" in the*Oracle Life Sciences Data Hub Application Developer's Guide*.

Integrated Development Environment Users with the necessary privileges can query data in Oracle LSH Tables through a Program instance, working in an integrated development environment (IDE) such as SAS or SQL Developer. For further information, see "Defining Tables" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Transferring Data Out of the Oracle Life Sciences Data Hub

You may want to transfer Oracle LSH data to a partner, regulatory agency, or data storage facility.

Definitional Object: Data Mart To transfer data out of Oracle LSH you can create a definitional object called a Data Mart. Oracle LSH supports several formats, including fixed length or delimited text, SAS Transport (XPORT or CPORT), SAS Data Sets, or Oracle Export. To reduce the size of these export files the executable can optionally compress them into a zipped file.

To transfer the Data Mart to another system, use FTP or email.

You can also use Data Marts for secure but inaccessible long-term storage within Oracle LSH. For further information on Data Marts, see "Defining Data Marts" in the *Oracle Life Sciences Data Hub Application Developer's Guide*

Validating Object Definitions and Instances

Pharmaceutical regulatory agencies require that you certify that data submitted to them for approval of a drug is valid. Oracle LSH cannot certify that data entering Oracle LSH from a different system is valid. You must validate incoming data in the source system.

However, Oracle LSH allows you to validate all definitional objects that interact with data within Oracle LSH, and to publish a report coversheet that records the validation status of all objects that touched the report's source data in Oracle LSH, so that you can certify that data remained valid within Oracle LSH.

In addition, Oracle LSH allows you to manually validate report outputs in a Report Set and to reuse the validated outputs even though other reports in the same Report Set are still being developed and must be regenerated.

Validation Status Oracle LSH allows you to associate a validation status—Development, Quality Control, Production, or Retired—with an object or output. You must determine your own criteria for promoting an object or output to a new validation status.

Oracle LSH uses object validation statuses and Work Area usage intent values to enforce rules for development, quality control, and production environments.

Validation Documentation Oracle LSH allows you to associate job IDs and documents with an object to record the object's validation progress. For example, you may decide that requirements specifications for an object such as a Report Set are required for its promotion from Development to Quality Control. You can associate a pointer to the requirements specifications document with the Report Set definition. For promotion to both Quality Control and Production, your criteria may include successful execution of an object such as a Program. In the Program definition you can include a link to the ID of the job that successfully executed the Program.

For further information, see Chapter 6, "Validating Objects and Outputs".

Reusing Object Definitions

As you begin to use Oracle LSH, you must define many new objects (though you can use tables, views, data sets, and programs that you have already developed in an external Oracle or SAS system as the basis for Oracle LSH Tables and Programs). But as time goes by, you will be able to reuse existing object definitions more and more, building up libraries of valid object definitions approved for reuse. For example, if you develop and test an enrollment Report Set for one study, you may be able to use the same Report Set definition, with modifications if necessary, for another study.

Object Definitions and Instances Object definitions are stored in a Library, either in an Application Area or a Domain. To use an object definition of any type—Table, Program, Load Set, for example—you must create an instance of it in a Work Area and install the instance to the database.

The instance is itself a defined object. Oracle LSH includes a predefined object instance type corresponding to each primary definitional object type: Table instance, Load Set instance, Program instance, Report Set instance, Workflow instance, Data Mart instance, and Business Area instance. An instance object contains only a pointer to the definition, a name, description, and, depending on the object type, may have a few other characteristics.

Secondary object types, such as Source Code and Parameter, also have a corresponding predefined object instance type: Source Code instance, Parameter instance. For information on primary and secondary Oracle LSH object definitions and instances, see the chapter on each primary object type in the *Oracle Life Sciences Data Hub Application Developer's Guide*. Also see Appendix A, "Object Types with Operations" and Appendix B, "Object Ownership".

How to Reuse Object Definitions There are several ways to reuse object definitions:

- **Create a new instance of an existing definition**. For example, if you have a standard Demography Table, you can use it in multiple studies simply by creating a new instance of it for each study. No additional validation of the definition is necessary.
- **Copy a definition and modify it as necessary**. If a new study requires an extra column in the Demography Table, for example, you can copy the standard Demography Table definition and add a column to it for use in the new study. You must revalidate the definition, but the definition process is simpler than recreating the whole Table.
- **Modify an existing definition**. If you discover a bug in a Program, for example, that should be fixed in all instances of the Program, you can modify the definition (if you have the necessary privileges) working through an instance of the Program. You can upgrade any or all instances of the original Program to the fixed version of the definition by running a job. (If you have the necessary privileges, you can also modify the definition directly in the Library. However, you cannot run the Program without creating an instance of it in a Work Area.)

  This method is also appropriate for modifying standard company definitions as they require change over time. For example, you may decide to add the new column to your standard Demography Table. In that case, you can create a new instance of the standard Table and add the column to the standard Table definition through the new instance.

  Existing instances of the original standard Demographic Table continue to point to the original version. You can upgrade any or all of them to the new definition if you so choose.

Developing Standard Objects and Modular Applications You can reduce the amount of time required to develop Oracle LSH applications by developing standard object definitions and reusing them as much as possible. To encourage programmers to reuse standard definitions, make it clear which definitions are standard by storing them in a special library (for example, a Standards Domain) and/or classifying them as Standard, for example.

Following are a few strategies for developing standards in object definition:

- Develop standard Table definitions. If you have a standard set of Tables, with standard data types and lengths for corresponding columns in different Tables, then you can easily reuse the same Table definition at different points in the data flow, and write standard Programs to read from and write to the standard Tables.

  You can develop your own company standards or use external standards. For example, if you plan to submit drug approval applications to the FDA using the CDISC data model, you may want to use the CDISC standards as your company standards and use Oracle LSH Programs to convert your existing data.

The standard data model you use must be compatible with the data model of the source system(s) you use.

- Define more Programs with a smaller scope instead of fewer Programs with a larger scope. You can then use the smaller-scope Programs as modules in multiple applications.

  For example, rather than define a single Program to combine data from different studies and then analyze the data, define two separate Programs, one to combine data and the other to analyze data. Both Programs are reusable in more situations. You can reuse the combining Program before different analytical Programs, and you can reuse the analytical Program, perhaps with minor modifications, on data from either a single study or multiple studies, for example.

  You can include a series of Programs in a single Workflow, so that you can run all the Programs in a single process.

- Make executable object definitions more reusable by defining Parameters to control the data to be processed

  For example, if you want to use the same Program to generate a Demography Report in several different studies, create a Parameter to contain the study name. You can either make the Study Parameter enterable by the user who runs the report, or, in the Execution Setup for each instance, bind the Study Parameter to the value appropriate for each study.

Migrating Existing Programs, Data Sets, and Tables from Other Systems

In Oracle LSH you can continue to use programs, data sets, and tables you have developed for use in other systems. Although this feature may be particularly useful as you begin to use Oracle LSH, you can continue to upload external programs, data sets, and tables and convert them to Oracle LSH Programs and Tables at any time.

Load Sets can convert Oracle tables and views, or SAS data sets, to Oracle LSH Tables, as well as load the data they contain or point to into Oracle LSH. Load Sets also automatically convert your Oracle Clinical Global Library definitions to Oracle LSH definitional objects and load any Oracle Clinical stable interface tables and data you choose. For further information, see "DefiningLoad Sets" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

You can upload SAS source code files, including macros and formats, into Oracle LSH Source Code definitions. For further information, "Defining Programs" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

2 Designing and Using the Oracle Life Sciences Data Hub

This section contains the following topics:

- Designing Your Implementation of the Oracle Life Sciences Data Hub on page 2-1
- Setting Up and Using the Oracle Life Sciences Data Hub on page 2-5

In the previous section you learned about some of the features and functions of the Oracle Life Sciences Data Hub (Oracle LSH). This section introduces some of the implementation decisions required to configure Oracle LSH for the most effective use at your company.

Over time, your Oracle LSH installation will probably store huge volumes of data. It will also contain a large number of defined objects such as SAS programs, and many outputs such as tabulations, figures, and listings. You will need to allow some people security access to some objects, data, and outputs, and allow other people security access to other objects, data, and outputs.

Before you begin using Oracle LSH, you must decide how best to organize your objects, data, and outputs so that you can create a security system that fits your needs and easily find objects and outputs. Oracle LSH provides several overlapping tools for this purpose. Just as in a library books are physically stored in rooms and on shelves in a logical order, yet you can also search for them in the card catalogue or library software by title, author, subject, or keyword; Oracle LSH lets you organize objects, data, and outputs physically in one way and categorize and search for them in other ways as well.

You must design the following:

- Organizational Structure on page 2-2
- Security System on page 2-2
- Classification System on page 2-3
- Subtypes (optional) on page 2-4
- Object Validation Standards on page 2-4

Designing Your Implementation of the Oracle Life Sciences Data Hub

This book, the *Oracle Life Sciences Data Hub Implementation Guide*, is intended for the people who will design these systems. We recommend that you assemble a team of people to plan how your company will use Oracle LSH and to design your implementation of Oracle LSH accordingly. The team should include representatives of all groups whose data will be stored in Oracle LSH, all groups who will view data from Oracle LSH, and all groups who will either administer Oracle LSH or develop the programs that will manipulate Oracle LSH data.

If you will use Oracle LSH for clinical data you should include, for example, study designers, data managers, programmers, quality control managers, statisticians, and information technology engineers. If you will use Oracle LSH for administrative, financial, or other types of data, you will need representatives from additional groups.

This book includes a chapter on each of the systems you must design. Each chapter ends with a summary of the decisions you must make, the implications of each decision, and where to find documentation on each follow-up task. This chapter introduces each system and outlines how they fit together:

- Organizational Structure
- Security System
- Classification System

This chapter also describes the following:

- **Subtypes**. Both the security system and the classification system make use of Subtypes, which are described on page 2-4.
- **Validation**. Before you use Oracle LSH to store or process production clinical data, you must develop standards for validating the defined objects that comprise your applications (see "Object Validation Standards" on page 2-4).

Figure 2–1 on page 2-6 shows a high-level workflow for building and using the Life Sciences Data Hub and where to find documentation for each phase.

Organizational Structure

You must design an organizational structure in which to physically store your defined objects, the data on which they operate, and the outputs—reports such as tabulations, listings, and figures—that they produce. In the library analogy, the Oracle LSH organizational structure is equivalent to the rooms and sections of shelves where books are stored.

You can use the organizational structure in the security system you design. Using the library analogy, you can allow some groups access to the medical reference room, for example, but not the history reference room. (What people in the group can do in the room depends on their role; see "Security System" on page 2-2.)

Oracle LSH's organizational structure consists of three nested defined objects: Domains, Application Areas, and Work Areas. Domains are intended to contain Application Areas and a library of validated object definitions (such as Tables, Programs, Report Sets, Load Sets, and Workflows) suitable for reuse. Application Areas contain Work Areas and a library of object definitions in various stages of development. Work Areas contain instances of the object definitions contained in Domains and Application Areas. By maintaining separate Work Areas for development, quality control testing, and production, you can create distinct database environments for each development phase and keep production data clean.

Organizational concepts and design issues are covered in Chapter 3, "Designing an Organizational Structure".

Security System

The security system controls access to everything in Oracle LSH: defined objects, data, outputs, and the user interface itself.

In the library analogy above, we see that you can allow access to different rooms or sections to different user groups. However, the actions a particular user can take in the room or section depends on his or her role within the group. Someone with the role of Librarian, for example, might be able to add new books to the collection and remove books that become outdated. Other people might be allowed to check books out, while others might be allowed only to look at books.

In Oracle LSH, you can assign user groups to containers in the organizational structure: Domains, Application Areas, and Work Areas. By default, objects inherit the user group assignments of their container. For example, a Work Area inherits the user group assignments of the Application Area. However, you can revoke the inheritance at any level and assign additional groups at any level.

The privileges a particular user in the group has on a particular object in a container depend on the role(s) assigned to the user within the group. For example, you might create a Programmer role with Create and Modify privileges for all object types in Application Areas and Work Areas, and View privileges on all object types in Domains. Any user assigned the Programmer role in a user group assigned to Domain X has all those privileges within that Domain.

In addition, Oracle LSH includes predefined application roles that give access to portions of the user interface. Every user must have at least one application role in order to use Oracle LSH. For example, everyone who needs to view reports on data in Oracle LSH must be able to see the Reports tab in the user interface, which requires the Consumer application role.

Security concepts and design issues are covered in Chapter 4, "Designing a Security System".

Classification System

The classification system allows you to label outputs and defined objects so that users can more quickly find what they are looking for. You can label the same object or output multiple ways to make it easier to find, like looking up a library book by its title or a keyword regardless of where it is physically located.

You can define hierarchies to create labels that are related to each other in a logical way, and use these relations in Oracle LSH's advanced search facility. Oracle LSH also uses these hierarchies to create a custom user interface for displaying outputs in the Reports screen.

For example, you could create a two-level hierarchy called Project/Study, populate the Project level with the names of all your ongoing and past projects, and populate the Study level with the names of all your ongoing and past studies, linking each study to the appropriate project.

In the Reports screen, Oracle LSH displays folders, subfolders, and outputs in about the same way that your personal computer does, using hierarchy values for the folder and subfolder names. If you define a Project/Study hierarchy, Oracle LSH displays the Project/Study hierarchy as a top-level folder. Within it are a set of folders, one for each project, labeled with the name of the project. Within each project folder are a subfolder for each study and any reports that are classified to the project as a whole. Within each study folder are reports that are classified to each particular study.

In addition, if your company is a pharmaceutical company, you could define a single-level "hierarchy" called CROs, and populate it with the names of all the Contract Research Organizations with which you work. Users could then search for a report, or for the Program that generated the report, by its project and study, or by the CRO that collected the data.

Or if your company is a CRO, you could define a single-level "hierarchy" called Pharmas and populate it with the names of all the pharmaceutical companies with which you work. Your users could then search for a report, or its generating Program, by its project and study or by the pharmaceutical company sponsor of the study.

You define hierarchies that reflect your business organization and practices, populated with values, or labels, from your environment, to make the organization of the Reports screen and the categories available for use in searching as intuitive as possible for your users.

You can classify the containers—Domains, Application Areas, and Work Areas—you define as your organizational structure, and set up classification by reference, so that the objects and outputs you specify are classified the same way (for a particular hierarchy level) as their container. For example, if an Application Area is classified, or labeled, as Study 01, you can set up your classification system so that by default all Program definitions there are automatically classified to Study 01 as well. You can change a reference classification to an explicit classification at any time if you have the necessary privileges.

Classification concepts and design issues are covered in Chapter 5, "Designing a Classification System for Searching and Browsing".

Subtypes

You can define one or more subtypes for each of the predefined primary object types: definitions and instances of Tables, Programs, Load Sets, Report Sets, Workflows, Data Marts, and Business Areas. Subtypes allow you to define broad categories of objects and outputs and different classification and security requirements for each category.

For example, if you will be storing both clinical and financial data in Oracle LSH, create a financial and a clinical subtype for most object types. You can then, for example:

- Allow financial personnel to see financial reports but not clinical ones, and clinical personnel to see clinical reports but not financial ones.
- Classify financial reports, but not clinical reports, by fiscal quarter as well as by project and study.

Subtypes are covered in Chapter 4, "Designing a Security System" and Chapter 5, "Designing a Classification System for Searching and Browsing".

Object Validation Standards

Oracle LSH keeps all definitional objects—including Tables, Programs, Report Sets, Workflows, and Data Marts—under version control, and tracks which objects touched which data within Oracle LSH. All Oracle LSH definitional objects have a validation status that represents a stage in the object's life cycle: Development, Quality Control, Production, or Retired. You should develop validation requirements for objects at the Quality Control and Production stages.

Oracle LSH allows you to link documents and outputs with objects to demonstrate that they meet a validation requirement. For example, you can store documents such as Functional Requirements, Test Requirements, or Test Cases for a Program or Report Set, and an output such as a log file or generated report to demonstrate that the Program or Report Set was successfully executed.

Program outputs (reports) also have a validation status. You can specify whether you want a Program's outputs to inherit their validation status or to require manual validation. In the context of a Report Set, you can validate one output manually and continue to reuse that output in the Report Set even though you rerun the Report Set to generate other outputs. Further information is included in the *Oracle Life Sciences Data Hub Application Developer's Guide* in the chapters on common development tasks and Report Sets.

Oracle LSH enforces certain system behaviors based on validation status.

You can begin application development in Oracle LSH before arriving at your standards for definitional object validation, but you should not go into production until you have accomplished this task and validated all objects that affect production clinical data.

Validation concepts and design issues are covered in Chapter 6, "Validating Objects and Outputs".

Setting Up and Using the Oracle Life Sciences Data Hub

After you design and set up the organizational structure and security and classification systems, you can begin the task of migrating existing programs, tables, views, datasets, and data into Oracle LSH, and creating new applications for analyzing and reporting data. You can then retrieve information from Oracle LSH in the form of reports, data visualizations, and data marts.

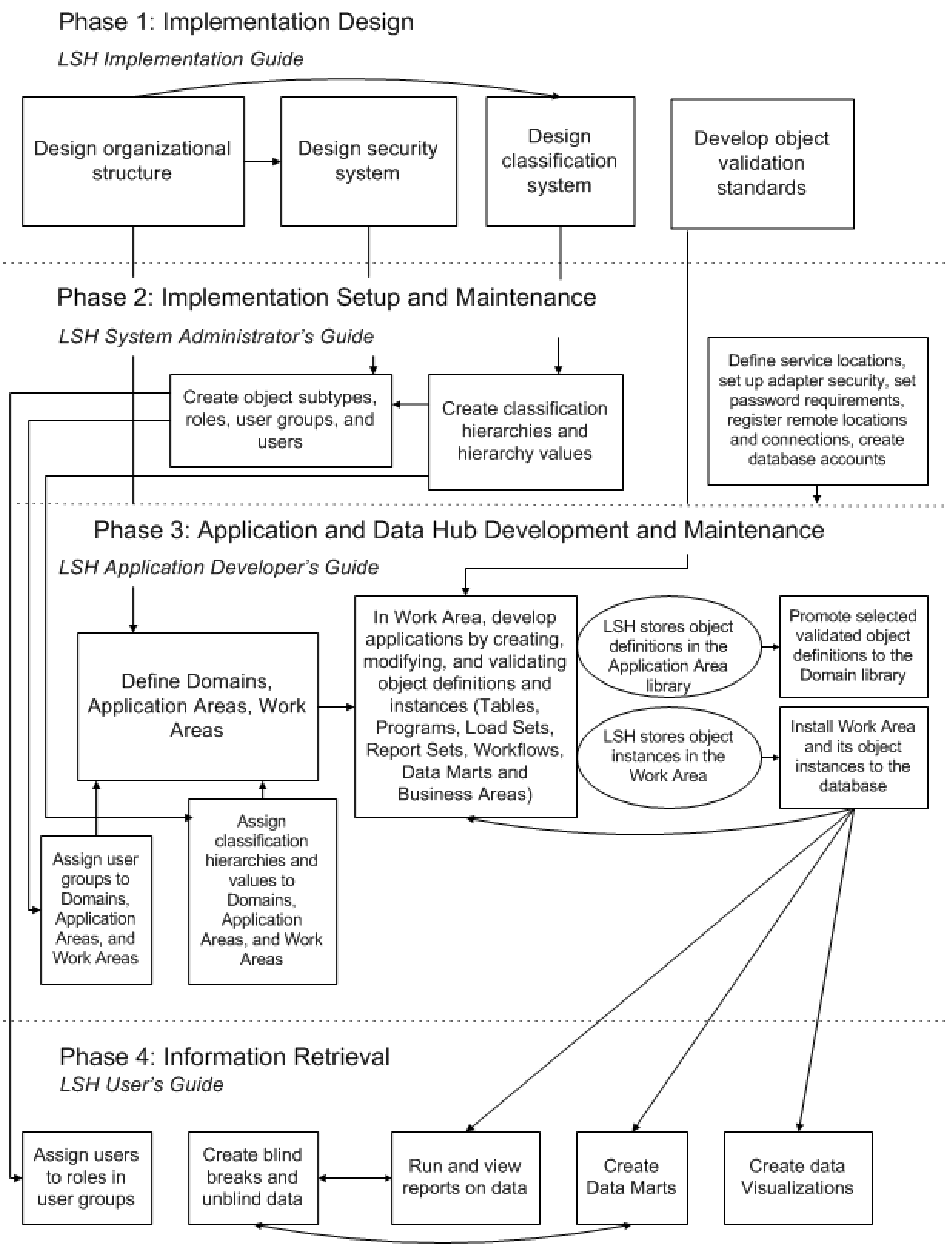
Oracle LSH implementation and usage can be divided into four phases, as shown in Figure 2–1. In general, a different set of people in your organization will be involved in each phase, and there is a different Oracle LSH manual for each phase.

After the first phase, you must complete at least a minimal amount of work for each task in any phase before you can proceed to the next phase. You can then work on all the tasks in parallel as necessary.

Figure 2–1 graphically represents the tasks in each phase and their relation to tasks in the subsequent phases. These relations are also described in the following sections:

- Phase 1: Implementation Design
- Phase 2: Implementation Setup and Maintenance
- Phase 3: Application and Data Repository Development and Maintenance
- Phase 4: Information Retrieval

Figure 2–1 Phases of Oracle LSH Implementation and Use, with User Documentation for Each Phase


Phase 1: Implementation Design
LSH Implementation Guide
Design organizational structure
Design security system
Design classification system
Develop object validation standards
Phase 2: Implementation Setup and Maintenance
LSH System Administrator's Guide
Create object subtypes, roles, user groups, and users
Create classification hierarchies and hierarchy values
Define service locations, set up adapter security, set password requirements, register remote locations and connections, create database accounts
Phase 3: Application and Data Hub Development and Maintenance
LSH Application Developer's Guide
Define Domains, Application Areas, Work Areas
In Work Area, develop applications by creating, modifying, and validating object definitions and instances (Tables, Programs, Load Sets, Report Sets, Workflows, Data Marts and Business Areas)
LSH stores object definitions in the Application Area library
Promote selected validated object definitions to the Domain library
LSH stores object instances in the Work Area
Install Work Area and its object instances to the database
Assign user groups to Domains, Application Areas, and Work Areas
Assign classification hierarchies and values to Domains, Application Areas, and Work Areas
Phase 4: Information Retrieval
LSH User's Guide
Assign users to roles in user groups
Create blind breaks and unblind data
Run and view reports on data
Create Data Marts
Create data Visualizations

Phase 1: Implementation Design

Before you can begin using the Life Sciences Data Hub (Oracle LSH), you must design and set up three systems, introduced in the following chapters. Detailed design information is included in subsequent chapters in this book, the *Oracle Life Sciences Data Hub Implementation Guide*. The three systems are:

- Organizational Structure on page 2-2
- Security System on page 2-2
- Classification System on page 2-2

In addition, to satisfy regulatory requirements you must develop your own standards for validating objects. See "Object Validation Standards" on page 2-4.

Phase 2: Implementation Setup and Maintenance

The second phase involves setting up the structures and systems designed in the Implementation Design phase. You may need to extend and modify these systems over time, so the instructions are included in the manual intended for the people who will maintain each system.

- Setting up the organizational structure—Domains, Application Areas, and Work Areas—is covered in the *Oracle Life Sciences Data Hub Application Developer's Guide*.
- Setting up the security system includes creating user groups and roles, assigning user groups to Domains, Application Areas, and Work Areas, and assigning users to roles within user groups. Instructions are included in "Setting Up the Security System" in the *Oracle Life Sciences Data Hub System Administrator's Guide*.
- Setting up the classification system includes creating flat and/or multilevel hierarchies of categories with values that represent the organization and/or purpose of your company's work, and assigning these to Domains, Application Areas, and Work Areas. Instructions are included in "Setting Up the Classification System" in the *Oracle Life Sciences Data Hub System Administrator's Guide*.
- Setting up integration with external systems includes setting up the Distributed Processing Server, setting up Adapter Security, Registering Remote Locations, and creating Database Accounts. These tasks are all covered in "Setting Up Adapters to External Systems " in the *Oracle Life Sciences Data Hub System Administrator's Guide*.

The *Oracle Life Sciences Data Hub System Administrator's Guide* also includes monitoring and troubleshooting information on Oracle LSH job execution.

Phase 3: Application and Data Repository Development and Maintenance

In this phase, you work in a Work Area to define all the objects necessary to accomplish a particular business purpose (application), and install them to an Oracle LSH database schema.

You then load data into the schema and run the application to test the programs, tables, and other defined objects. When the application is ready, you clone the Work Area and install it to a different Oracle LSH schema to be used for quality control, load fresh data, and formally test the application.

When the application is thoroughly tested and all its objects validated, clone the quality control Work Area, install it to a new Oracle LSH schema to be used for production, load production data, and allow clinical and other personnel to see Oracle LSH outputs as appropriate. The production environment can be modified as necessary over time, using the development and quality control Work Areas.

Over time, if you design modular applications, you can reuse object definitions in various combinations to create new applications with a minimum of programming effort.

This information is covered in the *Oracle Life Sciences Data Hub Application Developer's Guide*. The programmers and data managers who develop applications in Oracle LSH are called Definers because they define objects.

Phase 4: Information Retrieval

When Oracle LSH applications are running on production data, Oracle LSH Consumers—end users with no knowledge of the database or definitional objects—can find existing reports and data marts and run them on current data at any time if they have the necessary security privileges. They can also use Oracle Business Intelligence Enterprise Edition to create onscreen data visualizations.

These tasks are covered in the *Oracle Life Sciences Data Hub User's Guide*.

3 Designing an Organizational Structure

This section includes information on the following topics:

What is the Organizational Structure?

The Oracle Life Sciences Data Hub (Oracle LSH) provides three nested containers with which you can create an organizational structure that reflects your business and clinical organization. The containers are Domains, Application Areas, and Work Areas. Within these three containers you define and store metadata objects to create your Oracle LSH applications. Through Work Areas you install applications to the database to interact with data.

You must design a logically organized set of these containers Your design should include:

- **Any number of Oracle LSH Domains**, each with a specific purpose and each containing a library of validated, reusable object definitions and/or one or more logically related child Domains and/or Application Areas. See "Domains" on page 3-3 for further information.
- **Any number of Application Areas**, each contained in a logically related Domain and each having a specific purpose; for example, to generate a particular set of reports and/or to contain all the data for a single study or project. See "Application Areas" on page 3-5 for further information.
- **One primary Work Area in each Application Area** for use in application development. You may decide to let individual Definers have their own Work Areas in addition, but in the primary Work Area you install all the objects required for the application together. You clone this primary Work Area to create a test environment, and clone the validated test Work Area to create a production environment. See "Work Areas" on page 3-5 for further information.

Both the security and classification systems make use of this organizational structure. Objects can inherit both their security User Group assignments and their classifications from their parent container. Therefore you can control security access to object definitions and instances by setting up inheritance and assigning user groups to Domains, Application Areas, and Work Areas. You can follow a similar procedure to setup classification for object definitions and instances. You can exclude a particular object at any level from the security access or classification, but if you design a logical structure that works for both security and classification, your Oracle LSH implementation will be much simpler to set up and to use. You should design all three systems at the same time. See Chapter 4, "Designing a Security System" and Chapter 5, "Designing a Classification System for Searching and Browsing".

Domains, Application Areas and Work Areas are themselves objects that you must define. You can add Domains, Application Areas, and Work Areas at any time. The instructions for creating Domains, Application Areas, and Work Areas are contained in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

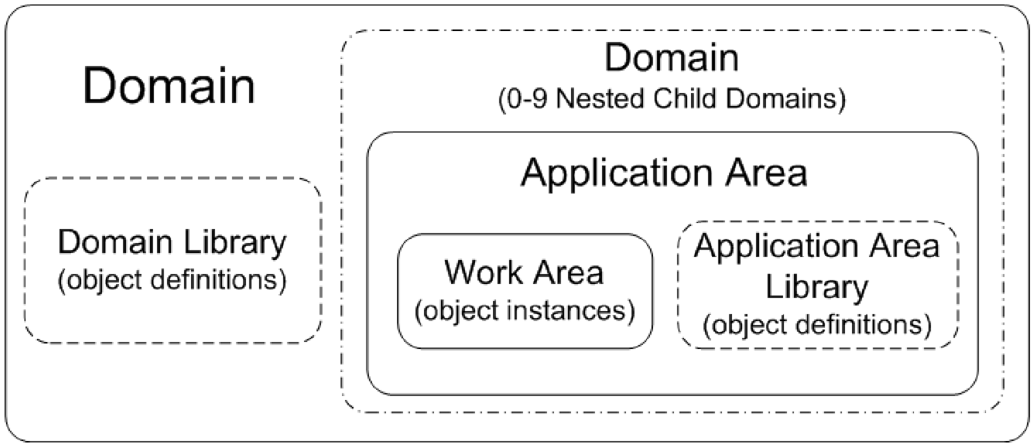
Notes: Oracle LSH can display a maximum of 200 rows at a time by default, so if you define more than 200 Domains within a Domain, or Application Areas within a Domain, or Work Areas within an Application Area, or objects within either an Application Area or Work Area, you get an error. Therefore Oracle recommends that you design your organizational structure to avoid this problem. Alternatively, it is possible to reset an Oracle Applications profile to display more than 200 rows at a time; however this affects all your Oracle Applications.

Also, keep container and Program names short if you are using an integrated development environment (IDE) such as SAS. Oracle LSH identifies objects in the IDE by their full pathname and the maximum length allowed for the full pathname is 256 characters. For further information see "Naming Objects" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

The relationship among the containers is described in the following sections and shown in Figure 3–1, "Oracle LSH Organizational Structure" below. The libraries in both Domains and Application Areas are shown with dotted lines because a Library is not a container that must be defined. A library simply consists of all the object definitions contained directly in either the Domain or the Application Area.

The Domain container inside the Domain container is shown in a different type of dotted line because you can choose to have no child Domains or up to nine (9) levels of nested Domains. See "Domains" on page 3-3 for further information.

Figure 3–1 Oracle LSH Organizational Structure



For ideas on how to use this structure, see "Examples of Organization Design" on page 3-9.

Many of the concepts used in this chapter were introduced in Chapter 1, "Overview". Appendix B, "Object Ownership" shows the container relationships among Domains, Application Areas, Work Areas, and all types of object definitions and instances.

Domains

A Domain is the top-level container in Oracle LSH. Domains have two purposes, and any one Domain can be used for one or both of these purposes:

- **Organizer**. A Domain can contain child Domains and/or Application Areas. You can use a Domain to organize these other containers—child Domains and Application Areas—into logical groups. For example, you might create a Domain for a particular drug project and one Application Area within it for each clinical trial in the project. Alternatively you could create a Domain for a project, create a child Domain within the project Domain for each trial in the project, and create an Application Area for each report set. See "Examples of Organization Design" on page 3-9 for other examples.
- **Library**. A Domain can contain a library of object definitions. See "Domain Libraries" on page 3-4.

  Even if the Domain's only purpose is to store library definitions, you may want to include an Application Area containing a Work Area in order to install and test the definitions within the Domain before promoting them to the Domain library.

You can define any number of Domains and any number of Application Areas within a Domain.

Nested Domains Whether or not you can define Domains within Domains, and if so, how many levels of Domains, depends on your setting for the Domain Nest Value profile; see "Setting Profile Values" in the *Oracle Life Sciences Data Hub System Administrator's Guide* for further information. The allowed values range from zero (no nested Domains; a single level of Domains that can contain Application Areas and object definitions only) and nine (9 levels of nested Domains). The default value is two (2).

The Domain Nest Value setting does not affect the number of Domains you can define at the same level within a parent Domain. You can define any number of sibling Domains at a given level.

In the Oracle LSH user interface, you can browse and search for objects in only one Domain at a time.

Note: When you load Oracle Clinical Global library definitions into Oracle LSH, the system automatically creates a Domain in Oracle LSH for Oracle Clinical Global Library objects and an Application Area for each Oracle Clinical Global Library Domain loaded into Oracle LSH. Each Application Area has the same name as the Oracle Clinical Domain whose Global library it contains. The Oracle LSH Tables, Variables, and Parameters created from the Oracle Clinical Question Sets, Questions, and DVGs become the library of that Domain.

Domain Libraries A library is not a defined object; object definitions belong to a Domain's library when they are stored directly in the Domain. Therefore you cannot have more than one library in a Domain.

Domains are intended to store only tested, validated, production-quality object definitions. Some Domains may contain production-quality objects that are suitable for reuse in many applications. Other Domains may contain object definitions currently used in production for a particular trial or project. Oracle LSH does not enforce this usage in any way; you must develop a validation policy to determine which object definitions are eligible for inclusion in a library and, if you choose, set up Oracle LSH security so that only a few people can move definitions into a Domain library and modify them there. For an example, see "Project 123 Domain User Group" on page 4-17.

Maintaining object definitions in a Domain library is not required. When a Definer creates an object in a Work Area, Oracle LSH automatically stores the definition in the Application Area that contains the Work Area. You can simply leave all object definitions in their Application Area library and still allow users with the necessary privileges to create instances of them in any Work Area or copy the definitions into any other Application Area library.

However, storing validated object definitions in a particular Domain library facilitates reusing these definitions by making it clear that these are intended for reuse. You can develop standard operating procedures requiring Definers to search first in Domain Libraries to find an existing definition that serves their purpose before searching in Application Area Libraries or creating a new definition. Reusing valid object definitions reduces your definition workload and ensures the consistency and quality of object definitions (see "Reusing Object Definitions" on page 1-10).

In general, if a Domain contains logically related Application Areas, it makes sense to use that Domain library for storing validated object definitions that have been developed or adapted specifically for that logical group of applications. If you have developed definitions that are suitable for use in many projects, trials, or applications, such as Enrollment or Demography Tables, Programs or Report Sets, you may want to create a Domain specifically for the purpose of maintaining a central library of standard reusable definitions, and call it the Standards Domain, for example.

You can make it easier to find object definitions in a library by labeling them with a classification. See Chapter 5, "Designing a Classification System for Searching and Browsing".

Application Areas

This section contains information on the standard and alternative usage of Application Areas.

Standard Usage of Application Areas

An Application Area is designed to contain a single business application that you want to validate, release, and maintain as a unit. For example, an Application Area's business purpose might be to manage a single study, including all the programs for transforming and reporting data in that study. Or an Application Area might be used to generate a set of projectwide reports, or to merge data for all studies in a project. An Application Area supports a business application in two ways:

- **Library**. An Application Area library contains the object definitions—Tables, Load Sets, Programs, Workflows, Report Sets, Data Marts, Business Areas and their component parts—created (or copied and modified) specifically to accomplish the business purpose of the Application Area. When you create a new object in a Work Area, Oracle LSH automatically creates the new object definition in the library of the owning Application Area. You can validate these definitions and move them to the Domain library if that is your policy. As in a Domain, the library is not a separate container but a collection of object definitions stored directly in the Application Area.
- **Work Area Container**. An Application Area contains all the Work Areas necessary to develop, test, and put into production all the object instances necessary to accomplish the business purpose of the Application Area over time. A Work Area may contain instances of object definitions that are stored in any Domain or Application Area library.

For further information on Work Areas, see "Using Work Areas" on page 3-8.

Alternative Usage of Application Areas or Child Domains

You can create Application Areas or child Domains whose sole purpose is to store library definitions. For example, in a Domain whose sole purpose is to store standard definitions, you could create Application Areas to store different categories of definitions to make the definitions easier to find by browsing. For example, you could create one Application Area for Demography-related Tables, Load Sets, Programs, Report Sets, and Workflows, and another Application Area for Adverse Events-related definitions.

The user interface separates objects within a library by object type, so there is no need to use Application Areas or child Domains to do that. You can use classification to label definitions by categories such as Demography and Adverse Events, but you can use definitions' classifications only during searching, not browsing.

Work Areas

A Work Area is designed to maintain a specific, installable, executable version of a particular life cycle stage of the whole business application supported by the Application Area. That is, a Work Area is intended to contain all the object instances required for the Application Area's business application. These may be instances of object definitions contained in the parent Application Area's library, in a Domain library, or even in another Application Area's library (though this is not recommended). See "Using Work Areas" on page 3-8.

Work Areas are the link between the LSH definitional subsystem, which consists solely of meta-data, and the actual database schemas that comprise the data repository.

Definitional objects' tables, views, and packages are instantiated in the database by installing a Work Area; see "Installing Work Areas" on page 3-7. All data loading, storage and manipulation in Oracle LSH are done through installed Work Areas.

When a Definer creates a new object in a Work Area, he or she can either create an instance of an existing definition or create a new definition and instance at the same time. In this case, Oracle LSH simultaneously creates the new definition in the containing Application Area and an instance of it in the Work Area.

Work Areas have a special operation called cloning that allows you to duplicate a Work Area and all its object instances and install them to a new set of schemas for the next stage in the application's life cycle. Use this functionality to create separate development, test, and production environments. See "Cloning Work Areas" on page 3-7.

The life cycle stages are represented in Work Areas' Usage Intent attribute, whose allowed values are the same as object validation statuses (except Retired): Development, Quality Control, and Production. Oracle LSH enforces certain rules based on the Work Area's Usage Intent value, its validation status, and the validation status of all its object instances. For a detailed explanation of how to use Work Areas for different application life cycle stages, see "Work Area Usage Intent and Validation Status" on page 6-5.

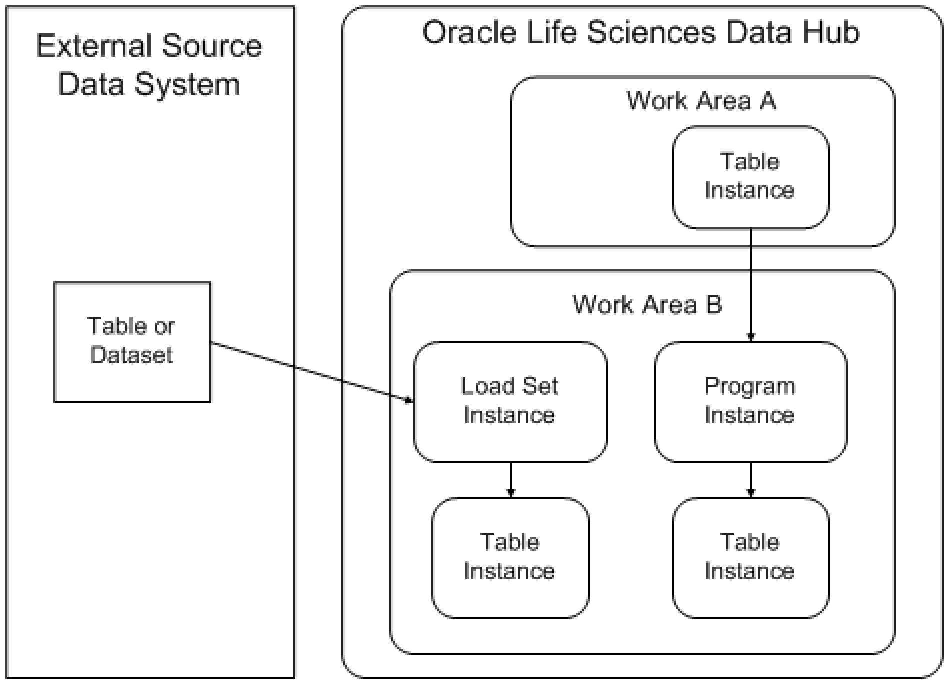
Data Flow

After a Work Area is installed, you can populate its underlying database schema with data in two basic ways:

- To load data from an external system, define, install, and run a Load Set (see "Loading Data" on page 1-4) in the Work Area.
- To operate on data already stored in Oracle LSH, define a Program in the Work Area and map each of its source Table Descriptors to a Table instance in any Work Area in Oracle LSH (see "Operating on Data" on page 1-7). Then install and run the Program.

Therefore it is not necessary to load data into the same Work Area where it will be processed. You can load a set of data to one Work Area and read it from a Program in any Work Area afterward. Figure 3–2 represents data flow options graphically.

Figure 3–2 Data Flow



Installing Work Areas

When you install a Work Area for the first time, the system creates a new set of database schemas (called an Oracle LSH Schema). The installation process also instantiates object definitions in the database through the Work Area's defined object instances. There are three installation modes:

- **Full installation** drops and replaces all objects. Any data already loaded into the schema is lost. It may be useful during development and testing to replace all objects and delete data that may be corrupted.
- **Upgrade installation** changes only objects whose version numbers have incremented since the previous installation, and preserves data by upgrading Tables instead of dropping and replacing. You can explicitly omit objects from an upgrade installation. Upgrade is the only installation mode allowed in Work Areas with a Usage Intent of Production.
- **Partial installation** enables multiple Definers to work in the same Work Area. Partial installation includes only objects that are not explicitly omitted and whose version number has incremented since the previous installation. The person who runs the installation can choose whether to upgrade or drop and replace Tables.

Newly created objects are automatically excluded from installation until their Omitted flag is explicitly unchecked. When a Definer is ready to install and test an object, he or she unchecks the flag. The object is included in the next Work Area installation, which may be started by the same Definer or someone else with the necessary privileges.

For further information, see "Using, Installing, and Cloning Work Areas" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Cloning Work Areas

You can use the cloning operation together with the Usage Intent Work Area attribute and object validation statuses (Development, Quality Control, Production, and

Retired) to create clean, controlled, distinct environments for application development, testing, and production. See "Work Area Usage Intent and Validation Status" on page 6-5.

The cloning operation is similar to the copy operation except that it is possible to clone a Work Area onto an existing Work Area, so that the clone overwrites the existing Work Area. For example, if you already have a Quality Control usage intent Work Area, and several objects fail quality control testing, you can fix the objects in the Development Work Area and then clone the whole Development Work Area onto the QC Work Area, creating a new version of the QC Work Area.

When you clone a Work Area, you add the same label to the source and target Work Area. The label is proof that the two Work Areas are identical. In addition, all the object instances in the Work Area are duplicated exactly in the target Work Area. Their validation statuses remain the same.

Using Work Areas

This section contains information on the standard and alternative ways to use Work Areas.

Standard Usage of Work Areas

Work Areas are designed to contain the entire application required for the business purpose of the Application Area, at a particular stage of its development. Oracle LSH uses an attribute (Usage Intent) and an operation (cloning) unique to Work Areas, to support controlled, validated passage from one life cycle stage to another: Development, Quality Control, and Production.

When you set up an organizational structure, you create one Work Area for each Application Area. Its Usage Intent and validation status are both set to Development. In standard usage, your team creates and tests objects in the development Work Area. When the objects in the Development Work Area are ready for formal testing, you clone the Work Area and all its object instances to create a duplicate Work Area with a Usage Intent of Quality Control (QC). You then install the QC Work Area, load fresh data, and test. When all objects are tested and validated, you can clone the Quality Control Work Area to create a duplicate Work Area with a Usage Intent of Production.

For details, see "Work Area Usage Intent and Validation Status" on page 6-5.

Alternative Usage of Work Areas

Oracle LSH was designed to allow multiple Definers to work in a single Work Area that contains an entire logical application, using the Partial installation mode to install their own work.

However, it is possible to use different Work Areas for different portions of the business application. For example, you can create a separate Work Area for each Definer. If you create multiple Work Areas for different portions of the application, we recommend that you also maintain a primary Work Area to contain the entire application to validate and clone as a whole. Definers can then develop new object instances and definitions in their own Work Areas and move the instances into the main Work Area when they are ready.

This approach has the following advantages:

- You can develop and validate different parts of the business application at different times. For example, if the business purpose of the Application Area is to analyze all the data for a particular study, you might need to develop and validate adverse events reports before efficacy reports for the study.

- Definers can install their Work Area at any time, never having to wait for another Definer's partial installation to complete. However, installation should not be a lengthy process. This issue is primarily of concern for large groups of Definers working on the same business application.

> **Note:** You can remove the individual Definers' Work Areas after you have copied their object instances into the main Work Area. Unused object instances are eliminated, making more space available. However, object instances do not require very much space.
>
> You can remove only Work Areas whose Usage Intent is Development, or Work Areas whose Usage Intent is Quality Control or Production but which do not contain any installed objects.

The approach has the following disadvantages:

- When Definers move objects from one Work Area to another they may need to remap Table instances to executables. If they copy and paste mapped executable instances and Table instance together, Oracle LSH maintains the mapping.

  Because only one executable can write to any given Table instance, normally target mappings would be in the Definer's Work Area and Oracle LSH would maintain the mapping if the Definer moves the two object instances together. However, source Tables would often not be located in an individual Definer's Work Area, so the Definer would need to remap to source Table instances and test the new mappings. Also, if a Definer copies the object instances instead of moving them, he or she must redo all the mapping.
- Definers should retest object instances after pasting them to the main Work Area.
- If Definers have used the same name for the same type of object instance in two separate Work Areas, one of them must be renamed when they are pasted into the same Work Area. Oracle LSH renames them automatically by appending an underscore and the number one (`_1`), or the next higher number, to the name.

Examples of Organization Design

Following are several examples of ways to organize Domains, child Domains, Application Areas and libraries. You can use a combination of these approaches or design your own approach.

Examples 1 and 2 are alternative ways to organize data handling for a project that contains multiple studies and for which you need to report on each study separately as well as the project as a whole. The main advantage of Example 2: Project Domain with Reporting on Pooled Data is that you can report on the same data snapshot for all studies and for the project. The main advantage of Example 1: Project Domain with Reporting on Separately Stored Study Data is that you do not need a separate container for unblinded data at the end of the trial, saving space in the database.

> **Note:** You could follow the same procedures without using child Domains. The child Domains serve to organize the Application Areas, making it easier to browse.

Example 1: Project Domain with Reporting on Separately Stored Study Data

Alternatively, you can keep each study's data separate throughout the process except to report on the project as a whole. Within each project Domain, create the following child Domains:

- **A Domain for Each Study**. Load data for each study in the project into an Application Area in its own Domain. Create a Load Set from the source tables or data sets. Oracle LSH stores the Load Set definition in the Application Area library.

  If you collect data for different studies using slightly different formats, define a Program to copy raw data into Table instances with a standard data type and length for the project. Create an instance of the Program for each study, mapping its source table descriptors to the appropriate raw data table instances.

  Create Report Sets and run each of them on the standardized study data. If necessary, create a different version of the Report Sets for each study.

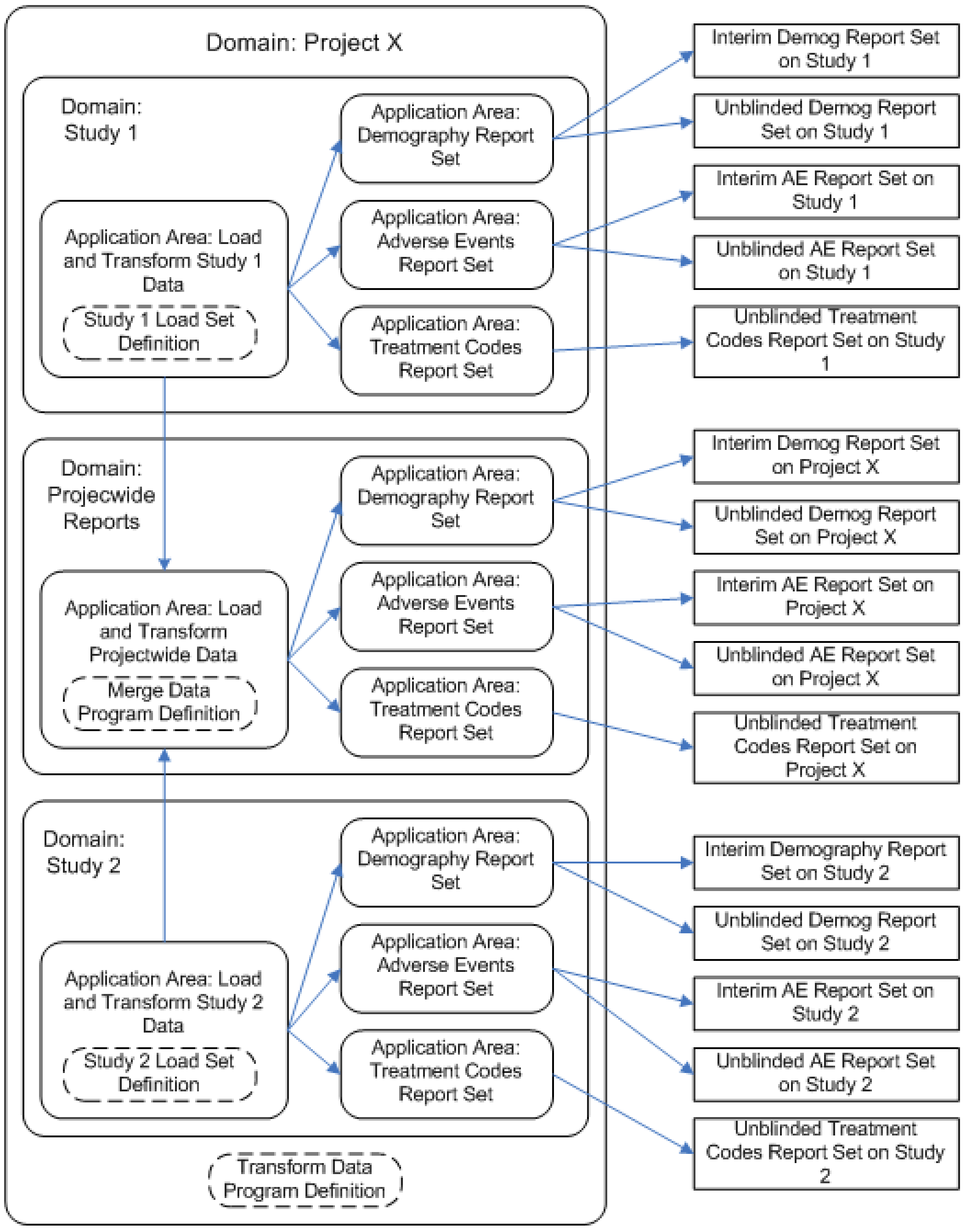
  When you are ready to unblind study data, run the Report Sets in Unblind mode or unblind each Table instance.

- **A Domain for Projectwide Data and Reports**. Define a Program to copy data from the standardized table instances for each study into a single set of table instances for the project.

  Run the Report Sets for the project as whole. If necessary, create a new version of each Report Set for the projectwide data.

  When all studies are unblinded, run the Report Sets in unblinded mode or unblind each Table instance and then run the Report Sets.

Figure 3–3 Organizational Structure for Example 2, with Data Flow


Domain: Project X
Domain: Study 1
Application Area: Demography Report Set
Application Area: Load and Transform Study 1 Data
Study 1 Load Set Definition
Application Area: Adverse Events Report Set
Application Area: Treatment Codes Report Set
Interim Demog Report Set on Study 1
Unblinded Demog Report Set on Study 1
Interim AE Report Set on Study 1
Unblinded AE Report Set on Study 1
Unblinded Treatment Codes Report Set on Study 1
Domain: Projecwide Reports
Application Area: Load and Transform Projectwide Data
Merge Data Program Definition
Application Area: Demography Report Set
Application Area: Adverse Events Report Set
Application Area: Treatment Codes Report Set
Interim Demog Report Set on Project X
Unblinded Demog Report Set on Project X
Interim AE Report Set on Project X
Unblinded AE Report Set on Project X
Unblinded Treatment Codes Report Set on Project X
Domain: Study 2
Application Area: Load and Transform Study 2 Data
Study 2 Load Set Definition
Application Area: Demography Report Set
Application Area: Adverse Events Report Set
Application Area: Treatment Codes Report Set
Interim Demography Report Set on Study 2
Unblinded Demog Report Set on Study 2
Interim AE Report Set on Study 2
Unblinded AE Report Set on Study 2
Unblinded Treatment Codes Report Set on Study 2
Transform Data Program Definition

Example 2: Project Domain with Reporting on Pooled Data

In this example, you create a Domain for every project. In this Domain, you load the data from all of the project's studies into Oracle LSH, transform data to a standard format for the project, merge the standard data from all studies into a single set of table instances for the project, and report on the merged, standardized data.

Within each project Domain, you create the following child Domains:

- **Load and Transform Data**. Load data for each study in the project into its own Application Area. Create a Load Set from the source tables or data sets. Oracle LSH stores the Load Set definition in the Application Area library.

If you collect data for different studies using slightly different formats, define a Program to copy raw data into Table instances with a standard data type and length for the project. Create an instance of the Program for each study, mapping its source table descriptors to the appropriate raw data table instances.

- **Merge and Report Data**. Define a Program to copy data from the standardized table instances for a study into a single set of table instances for the project, adding a column for Study ID to each table if it is not already there.

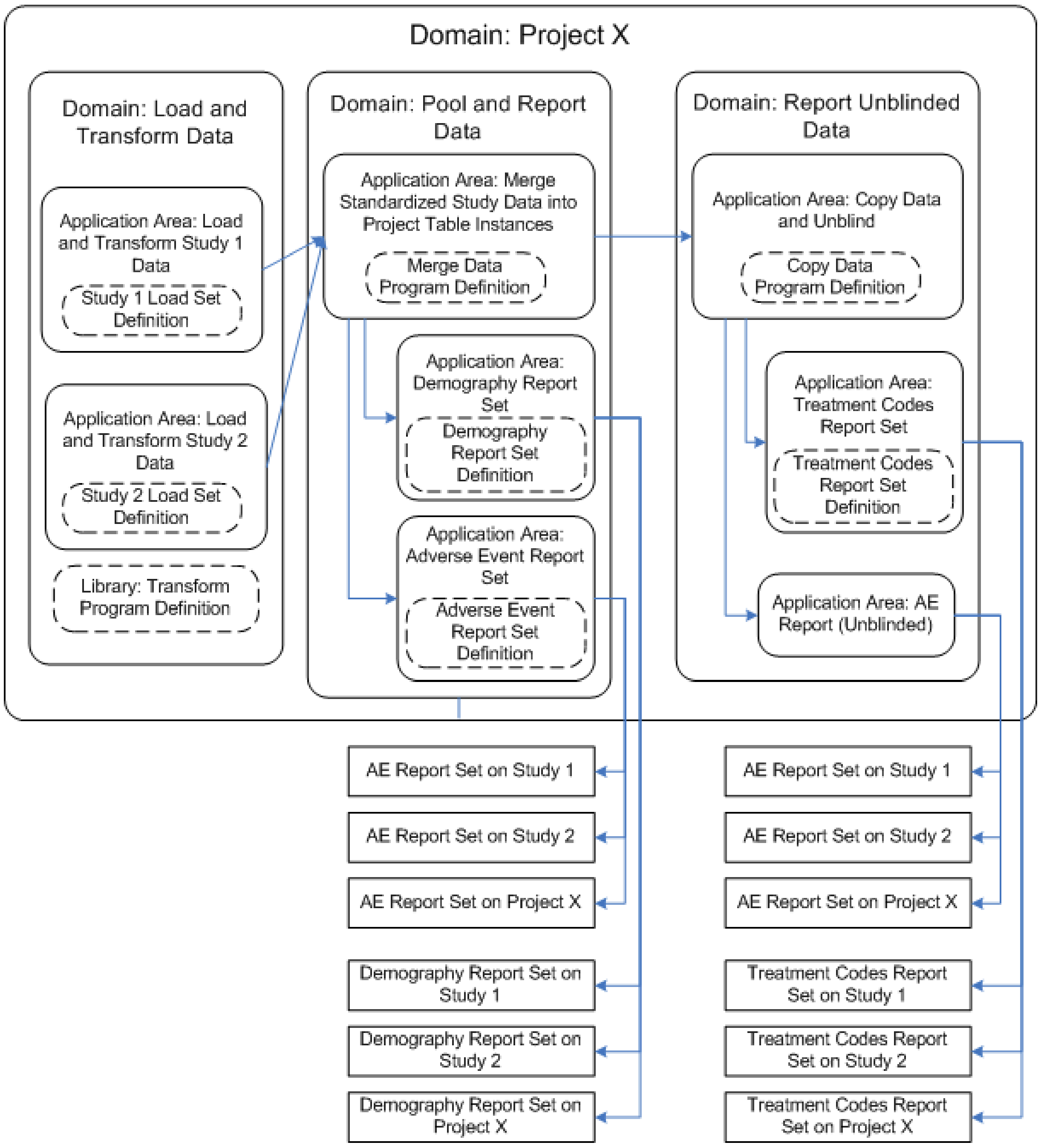
  Create Report Sets and run each of them on the pooled standardized data. Use Parameters to specify which study to report on and create a different Execution Setup for each study. By accepting all values for the Study ID Parameter you can create a Report Set on the project as a whole.

  You can create data snapshots and run each Report Set for each study and the project as a whole on the same data snapshot.

- **Unblind and Report Data**. If you need to unblind data for each study at a different time, create another Domain to hold the unblinded data. Define a Program to copy data with a particular study ID from the Merge and Pool Data Domain to the Report Unblinded Data Domain. To unblind a study, run this Program setting the Study ID Parameter to the correct value for the study, and run it in Unblind mode.

  As each study becomes ready for unblinding, repeat the procedure. You can run the same or different report sets on the unblinded study data for each study and, when all studies in the project are unblinded, run the same Report Sets on projectwide data.

Figure 3–4 Organizational Structure for Example 1, with Data Flow


Domain: Project X
Domain: Load and Transform Data
Application Area: Load and Transform Study 1 Data
Study 1 Load Set Definition
Application Area: Load and Transform Study 2 Data
Study 2 Load Set Definition
Library: Transform Program Definition
Domain: Pool and Report Data
Application Area: Merge Standardized Study Data into Project Table Instances
Merge Data Program Definition
Application Area: Demography Report Set
Demography Report Set Definition
Application Area: Adverse Event Report Set
Adverse Event Report Set Definition
Domain: Report Unblinded Data
Application Area: Copy Data and Unblind
Copy Data Program Definition
Application Area: Treatment Codes Report Set
Treatment Codes Report Set Definition
Application Area: AE Report (Unblinded)
AE Report Set on Study 1
AE Report Set on Study 2
AE Report Set on Project X
Demography Report Set on Study 1
Demography Report Set on Study 2
Demography Report Set on Project X
AE Report Set on Study 1
AE Report Set on Study 2
AE Report Set on Project X
Treatment Codes Report Set on Study 1
Treatment Codes Report Set on Study 2
Treatment Codes Report Set on Project X

Example 3: Domain for Companywide Reports

The companywide Domain holds Application Areas for companywide reports such as a list of all investigators updated monthly for submission to the FDA, and patient enrollment charts.

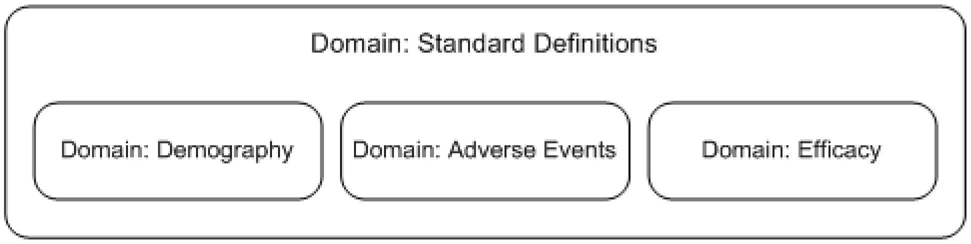
Example 4: Domain for Standard Definitions

Use standard definitions of Tables, Programs, Data Marts, and so on to make it easier to reuse definitions and reduce the amount of work required for validation. Put standard definitions in a Domain only when you have thoroughly tested and validated them so that they are approved for reuse.

You may want to use CDISC data formats for standard Table definitions and create Programs that read from and write to those tables, for example.

You can use child Domains to organize definitions by type. For example, create a child Demography Domain to store demography-related Tables, Programs, Report Sets, and Data Marts. Create an Adverse Events Domain and an Efficacy Domain, and so on.

Figure 3–5 Standard Definitions Domain



Note: You can create these definitions as needed in other Domains and move them to the standard definition Domain when they are ready, or you can create Application Areas and Work Areas inside these Domains for the purpose of creating and testing standard definitions.

Example 5: Oracle Clinical Definitions

When you define and run a Load Set for the Oracle Clinical Global Library, the adapter automatically creates an Oracle LSH Domain into which it puts Oracle LSH Variables, lists of values, and Tables that it converts from user-defined Oracle Clinical Questions, DVGs, and Question Groups. For further information, see "Defining Load Sets" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Example 6: Domain for Cross-Project Analysis

In combination with any of the other examples, you can create one or more Domains to use for cross-project analysis. For example, you could create one Domain for one group of projects, and another Domain for a different set of projects.

Example 7: Study or Project Template Domain

Develop standard definitions for use in all studies, validate them, and put them in one or more Domains of standard definitions. Create a Study Template Domain with at least one Application Area and copy into it the standard definitions that you know need to be modified for each study. Create a Work Area in the Application Area that contains instances of the definitions in the Study Domain and also instances of definitions in the Standard Domain that do not need modification. In the Work Area, map all executables to Table instances as necessary.

When you begin a new study, copy the entire Study Template Domain, including its Application Areas Work Areas. The object instances in the copied Work Area point to the correct definitions, either in the copied Application Area or in the original Domain. All mappings remain intact.

Through the instances in the copied Work Area, modify the definitions in the Application Area as necessary for the new study.

This model works with Example 1: Project Domain with Reporting on Separately Stored Study Data on page 3-10. You could work out a similar process for the entire Domain, or for the Domain in Example 2: Project Domain with Reporting on Pooled Data on page 3-11.

Design Considerations

As you design an organizational structure for your company, take the following points into consideration:

Security System Users are allowed to perform an operation on an object when they:

- belong to a User Group that is assigned to that object, either directly or indirectly by inheritance
- are assigned to a Role within the User Group that allows the requested operation on the requested object subtype

Objects inherit the User Group assignments of their containing objects, unless the User Group is explicitly unassigned from a particular object. If you assign a User Group to a Domain, that User Group also has access to everyDomain and Application Area in the Domain and every object definition in every Domain and Application Area, including every Work Area, and every object instance in every Work Area. However, users can perform only the operations defined by their Role(s) in the User Group.

You can unassign a User Group from an object at any level, in which case any objects it contains are also no longer assigned that User Group.

Outputs—reports, report sets, data marts, and visualizations—inherit their User Group assignments from the object instance that generates them (Programs, Report Sets, Data Marts, and Business Areas, respectively). Object instances inherit their User Group assignments from their containing Work Area. Therefore you can grant Consumers (end users) access to outputs by assigning them to a User Group that is assigned to the Production Work Area of the appropriate Application Area.

See Chapter 4, "Designing a Security System" for further information.

Classification System Classifications are used to label objects and outputs so that they are more easily found both by browsing the user interface and by advanced searches.

Objects can inherit the classifications of their container objects. Therefore, if you set up Domains and Application Areas in the same logical pattern you need for classification, you can effectively assign classifications to all the objects within a Domain or Application Area simply by assigning the classifications to the Domain or Application Area.

For example, if a Domain supports all of Project A and its Application Areas each support one study within Project A, you can assign an appropriate classification to each Application Area, such as Project A/Study 01 or Project A/Study 02. Then all Reports and other outputs produced from within each Application Area are automatically labeled as Project A/Study 01 or Project A/Study 02, and users can search for them in that context.

User Interface In the Oracle LSH user interface you can view only one top-level Domain at a time, with all the Domains and Application Areas it contains. To browse object definitions and Work Areas in other Domains, you must change Domains. Similarly, if you are viewing outputs, you can browse the outputs of only one Domain at a time.

You can search for objects across all Domains.

Business Areas Business Areas are the bridge between the Oracle LSH database and data visualization tools such as Oracle Business Intelligence Enterprise Edition. A Business Area contains Table Descriptors that are mapped to Table instances. A data visualization can include only data mapped to a single Business Area.

When you create a Business Area, you can map to all Table instances in the Work Area at once if you choose to, simplifying Business Area definition. You can also map Business Area Table Descriptors to Table instances in other Work Areas, and/or to selected Table instances in the same Work Area. Therefore your data visualizations are not restricted to data contained in a single Work Area.

Design Decision Summary

This section lists the basic design decisions you must make, with information on restrictions, if any. To work on your design, it may be helpful to sketch a diagram similar to those in the section "Examples of Organization Design" on page 3-9.

Domains

You must make the following decisions about Domains:

- Decide the number and purpose of each Domain.
- Decide whether or not the Domain will contain a library of object definitions. If it will, decide the purpose of the library.
- Decide whether or not the Domain will contain child Domains and if so, whether the child Domains will contain their own child Domains (and so on up to a total of 9 levels of Domain). If there will be child Domains, decide the purpose of each one.
- Decide whether or not the Domain will contain Application Areas (see "Application Areas" on page 3-17).

  Even if the Domain's only purpose is to store library definitions, you may want to include an Application Area containing a Work Area in order to install and test the definitions within the Domain before promoting them to the Domain library.

If the Domain will include a library, you should also plan:

- validation requirements for objects in the library, including a minimum validation status and what supporting information is required, if any (see Chapter 6, "Validating Objects and Outputs")
- security requirements for objects in the library, including one or more roles with privileges allowing moving object definitions into the library and modifying them there (for an example, see "Project 123 Domain User Group" on page 4-17).

Restrictions None.

You can define any number of Domains, and each Domain can contain a library of object definitions of any validation status and/or any number of Application Areas. The number of levels of child Domains allowed is determined by your setting of the Domain Nest Value profile; see "Nested Domains" on page 3-3 and "Setting the

Maximum Number of Nested Domains" in the *Oracle Life Sciences Data Hub System Administrator's Guide* for further information.

Considerations Take the following into consideration:

- Domains are an important part of the security and classification systems; read Chapter 4, "Designing a Security System" and Chapter 5, "Designing a Classification System for Searching and Browsing" before you reach a decision on Domain usage.
- You can browse objects in only one Domain at a time in the Oracle LSH user interface. However, you can search for objects in any Domain at any time.

Application Areas

You must decide the number and purpose of Application Areas in each Domain.

Restrictions None.

You can define any number of Application Areas in a Domain.

Considerations The business purpose of each Application Areas contained in a Domain should be a part of the business purpose of the Domain.

For example, if the purpose of the Domain is to contain all the data for a drug project, the purpose of each Application Area might be to contain all the data for a particular study that is part of that project.

If the purpose of the Domain is to contain standard Report Sets, the purpose of each Application Area might be to contain a particular Report Set.

See also "Alternative Usage of Application Areas or Child Domains" on page 3-5.

Work Areas

For normal usage, you need to create one Work Area in each Application Area.

Restrictions None.

Considerations In standard Application Area usage, as time goes on you will clone this Work Area to create a Quality Control Work Area, and then clone the QC Work Area to create a Production Work Area.

It is possible to use an Application Area for object definition storage without a Work Area. However, even if you are using the Application Area for definition storage, you may want a Work Area to install and test the definitions. See "Alternative Usage of Work Areas" on page 3-8.

4 Designing a Security System

This section includes information on the following topics:

About Security Design in the Oracle Life Sciences Data Hub

The Oracle Life Sciences Data Hub (Oracle LSH) security system determines which users can perform which operations on which defined objects (such as Programs and Tables) and outputs (such as reports). Users are allowed to perform an operation on an object or output when they:

- belong to a user group that is assigned to the object or output either explicitly or by inheritance
- and are assigned to a role within that user group that allows the operation on the object

You must design and set up a security system before you can use Oracle LSH.

To create an Oracle LSH security system you must do the following:

1. Design an **organizational structure**, or set of containers—Domains, Application Areas and Work Areas—to contain definitional objects(see Chapter 3, "Designing an Organizational Structure"). If you grant security access to user groups at the Domain, Application Area, and Work Area levels, the user group assignment automatically cascades to all contained objects. Design your security system in conjunction with your organizational structure to take advantage of this feature and minimize the number of user group assignments you need to make to individual objects (see "User Group Assignment by Inheritance" on page 4-6).
2. (Optional) Define one or more **subtypes** for any of the predefined object types; for example, Clinical Outputs and Financial Outputs. You can then assign different privileges to different roles on different subtypes of the same object type; for example, allow statisticians to view Clinical Outputs but not Financial ones. Subtypes are also used to differentiate classification requirements (see Chapter 5, "Designing a Classification System for Searching and Browsing"). One subtype is predefined for each object type.

3. Design a set of **roles**. A role consists of a set of privileges: operations allowed on object subtypes. Some roles should reflect professional job descriptions at your company. Other roles should cover specific Oracle LSH responsibilities such as breaking data blinds. Users can have multiple roles in the same user group and different roles in different user groups. See "Roles" on page 4-5.

   Predefined **application roles** determine user access to the user interface (UI) but are not used in the security enforcement algorithm for defined objects or outputs. See "User Interface Security" on page 4-8.

4. Design and create a set of **user groups**, each with a set of roles, and assign them to Domains, Application Areas, or Work Areas. See "User Groups" on page 4-6.

To complete the security system you must create a user account for each person who will use Oracle LSH and assign at least one user to be the group administrator for each user group. The group administrator then assigns users to roles within the group. Each group administrator must then add users to his or her group and assign users to roles within the group.

You can add organizational containers, object subtypes, roles, and user groups, as well as users, over time as necessary.

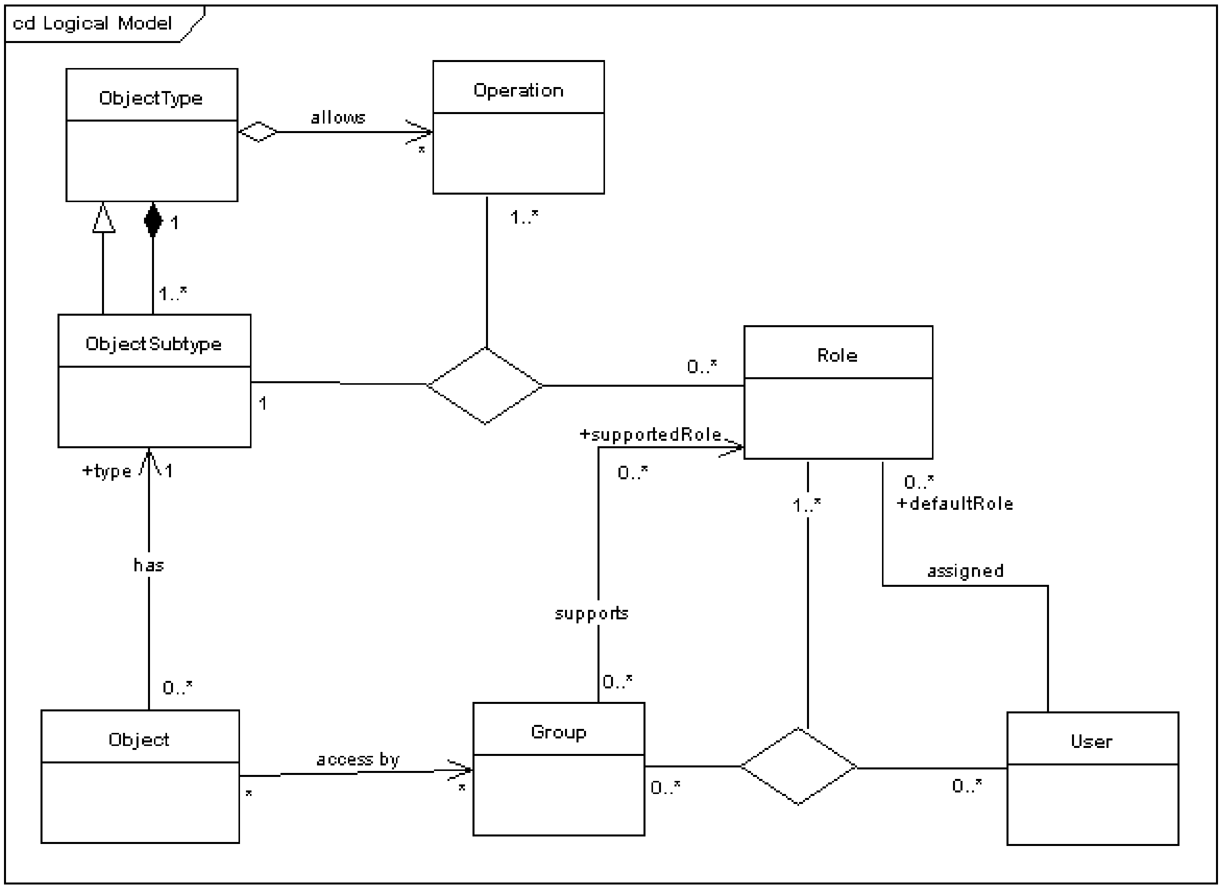
Note: In this chapter the word "object" is used to apply to object definitions and object instances. Users get security access and privileges on both definitions and instances in the same way. However, you can define different security requirements for each because they are distinct object types. For example, you can give a Definer privileges on Program definitions as well as instances, and give a Consumer access to Program instances only.

Object Security Elements

The Oracle LSH security system is based on objects subtypes, roles, user groups, and user accounts. Object types and the "Default" object subtype for each object type are predefined. Defining additional object subtypes is optional. You must define all the other elements. Instructions are included in "Setting Up the Security System" in the *Oracle Life Sciences Data Hub System Administrator's Guide*. The elements are described in the following sections:

- Object Subtypes on page 4-3
- Roles on page 4-5
- User Groups on page 4-6
- Users on page 4-8

Figure 4–1 Logical Representation of the Object Security Elements and Their Relations



As shown in the above diagram, object security elements have the following relations:

- Each object type defines which operations can be performed on objects of that type.
- Each object type must have at least one object subtype.
- Object subtypes inherit characteristics from object types (namely allowed operations).
- Each object is of one and only one subtype.
- A role specifies which operations can be performed on an object of a given subtype by a user with that role (in a user group assigned to the object).
- User groups support roles.
- A user is assigned to a user group by assigning him or her to one or more supported roles in the user group.
- User groups are assigned to objects and users gain access to an object through one or more user groups assigned to the object. The operations they can perform on an object depend on the roles they have in user groups assigned to the object.

Object Subtypes

Oracle LSH ships with predefined object types, including Program definitions, Program instances, Table definitions and instances, Report Set definitions and instances, and so on. Each object type includes a set of operations allowed on objects of that type; for example, Modify or Submit (for a complete list, see Appendix A, "Object Types with Operations"). One subtype (called "Default") is shipped for every object type.

All user-defined objects must be created based on an object subtype. An object's subtype helps determine its security and classification requirements. If you choose to, you can create additional subtypes; for example, Clinical and Financial. You can then differentiate security and classification requirements for different subtypes of the same object type.

For example, you can create two subtypes—Clinical and Financial—for Outputs. You can then assign different roles to the same operations on the two subtypes, so that some people can view financial reports but not clinical ones, and others can view clinical reports but not financial ones.

Subtypes and Security

Object subtypes have the same set of allowed operations as their parent object type. You must create a set of roles and assign each role to a set of operations on object subtypes. Users with a particular role can then perform the operations on the subtypes specified for the role, if the user group in which they have the role is assigned to the object.

If you create additional subtypes, you can assign different roles to their operations. For example, you might want to allow different people to view clinical and financial reports. To do this, you must create the appropriate roles and assign them to the appropriate operations on only the appropriate object subtype. For example:

- Give the Financial Officer role privileges on the Financial Output subtype but not the Clinical Output subtype.
- Give the Statistician role privileges on the Clinical Output subtype but not the Financial Output subtype.
- Give the Trial Administrator role privileges on both Output subtypes.

Subtypes and Classification

Oracle LSH uses classifications to help users find objects through searching and browsing. You assign classificationhierarchy levels to subtypes to determine the classification hierarchy levels to which an object of that subtype can and must be assigned. For example, you can require that Financial Outputs be classified to the Fiscal Year > Quarter > Month hierarchy but Clinical Outputs be classified to the Project > Study > Site hierarchy.

See Chapter 5, "Designing a Classification System for Searching and Browsing" and specifically "Assigning Classification Levels to Object Subtypes" on page 5-6 for further information.

Subtype Inheritance

When a Definer creates an object, the object inherits its parent's subtype by default, if that subtype exists for its object type. The Definer can change the subtype as necessary if there is more than one subtype for the object type. For example, if a Definer creates a Report Set definition in an Application Area whose subtype is Clinical, the Report Set inherits a subtype of Clinical if you have created a Report Set subtype called Clinical. If not, the report Set definition is created with the subtype designated as the default for Report Set definitions—either the shipped subtype called Default or another subtype you designate as the default for Report Set definitions.

Definers create Report Set instances in Work Areas, so Report Set instances inherit their subtype from the Work Area, if the same subtype exists for Report Set instances.

See Appendix B, "Object Ownership" for information about all parent/child object relationships.

Roles

A role consists of a set of privileges: operations allowed on a set of object subtypes. A single user can have any number of roles, and can have different roles in different user groups. You must assign roles to user groups and to users within user groups.

Roles Based on Job Description

Most of the roles you define should correspond to a professional role such as Statistician or Data Manager, and allow the appropriate operations on appropriate objects for that job description. For example, the Statistician role might have View and Submit privileges on Clinical Report Sets only, while a Data Manager or Programmer might have full privileges on all types and subtypes of object definitions and instances.

Consumers need to be able to run and view reports. For this they need the Submit privilege on Execution Setups and the View operation on outputs.

Some Consumers may need to create data visualizations using Oracle Business Intelligence Enterprise Edition. These people need View privileges on Business Area instances and Read Data operations on Table instances to have access to Oracle LSH data.

Programmers (Oracle LSH Definers) need many privileges. They need to be able to run and view reports as Consumers do. In addition, they need at least View privileges on Domains and Application Areas in order to see Domains and Application Areas and browse for objects contained in them. Definers also need View privileges on an object subtype in order to copy a definition of that subtype, and Create privileges for each object subtype on an Application Area in order to create a definition of that subtype in an Application Area.

Oracle LSH-Specific Roles

Oracle LSH-specific roles are useful for sensitive operations that only a few people should be allowed to do in Oracle LSH. You can assign these roles to users who also have more general roles assigned. For example, you may want to define the following roles:

- **Blind Break**. A blind break is the execution of a report on real, sensitive data while it is still blinded—usually done for medical emergency reasons during the course of a trial. Create a Blind Break role specifically to allow creating a one-time blind break on patient data at runtime, and read the resulting report(s). This requires the Blind Break operation on both Table instances and on Outputs.
- **Unblind**. Create an Unblind role to allow permanently unblinding Tables at the end of a study.

Note: To perform a blind break or to permanently unblind data, a user must have a special application role as well as access to the particular object through a user group; see "Application Roles" on page 4-8.

- **Librarian**. Create a Librarian role with all operation privileges on all definitional object types, add the role only to user groups assigned to a Domain, and within those user groups assign the role only to the few Data Managers you want to be able to modify library definitions. (You can allow all definition-related roles, such as Programmers and Data Managers, to view all object definitions in the Domain library so that they can see, copy, and reference library definitions.)

Application Roles

In addition, Oracle LSH ships with a set of predefined roles that serve an entirely different purpose. Most of these **application roles** allow access to parts of the Oracle LSH user interface (UI) such as tabs and subtabs, and allow specific actions in those UI tabs and subtabs. You must assign at least one application role to each user account. See "User Interface Security" on page 4-8 for further information, including a list of Oracle LSH's application roles.

User Groups

A user group definition consists of a name, description and a list of roles available to assign to users within the group. A user group also includes at least one group administrator, who adds users to the group and assigns them to roles within the group. The group administrator can also remove users from the group and remove roles from users within the group.

The security administrator creates users and user groups and assigns users to be group administrators, but cannot add or remove users from groups or assign him or herself as a group administrator.

To make user group creation easier, the security administrator can copy one user group definition, which includes the roles assigned to the group, to create another user group. It is possible to copy the assigned users as well. The group administrator can then add and delete users and change users' role assignments.

User groups determine which users have access to which objects. (The operations they can perform on an object depend on the roles they have within the user group that allows them access to the object.) To have access of any kind to an object, a user must belong to a user group that is assigned to the object. User groups can be assigned to an object either directly or indirectly by inheritance.

User group object assignments are not object version-specific. As an object is upgraded to one new version after another, the same user groups have access to it unless you explicitly usassign them. If you usassign a user group, its users can no longer see any version of the object.

To explicitly assign or unassign a user group to an object, or to revoke an inherited user group assignment, a user must have a role that allows the Manage Security operation on the object.

Explicit User Group Assignment to Objects

If you have the necessary privileges, you can explicitly assign a user group to any object, including containers, object definitions, object instances, Execution Setups, and outputs, at any time.

User Group Assignment by Inheritance

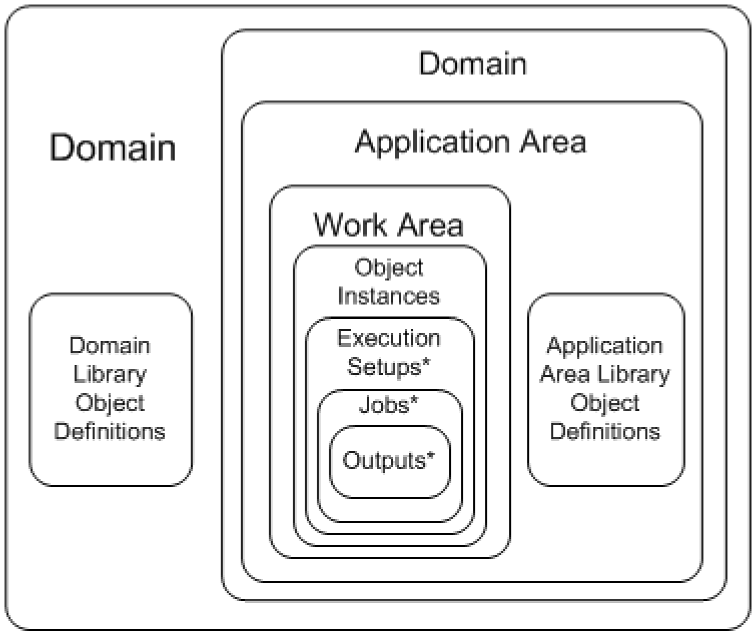
Objects inherit the user group assignments of their container object. You can explicitly revoke an inherited user group assignment for a particular object if necessary.

For example, if you assign a user group to a Domain, the system assigns that user group by default to all child objects as they are created: the child Domains in the Domain (if any), all the Application Areas in the Domain and in all of its child Domains, and all the object definitions and Work Areas in the Application Areas, all the object instances in the Work Areas, and all the execution setups and outputs of all the object instances. See Figure 4–2, "LSH Object Ownership" and Appendix B, "Object Ownership" to see these relationships displayed graphically.

Assignment revocations are also inherited. If you explicitly revoke the inherited user group assignment to a particular Application Area within the Domain, the user group is also no longer assigned to any of the object definitions or Work Areas contained in the Application Area.

Because user group assignments are inherited, in general it is best to create user groups with narrowly defined roles to assign to Domains. Assign users to roles with maximum privileges at the level where they will primarily work: for Definers, the Application Area (to allow creating object definitions through instances in the Work Area). Put Consumers in a user group assigned to the production Work Area so they can see and execute Execution Setups and outputs. See "Example of Security Design" on page 4-16.

Figure 4–2 LSH Object Ownership



Note: *Execution Setups, jobs and outputs are contained in instances of executable objects only—not in Table or Business Area instances.

Figure 4–2 shows the containers that constitute the organizational structure—Domains, Application Areas, and Work Areas—and the object definitions or instances they contain.

Duplicate Assignments

It is possible to assign the same user group to an object both implicitly (by inheritance) and explicitly. In that case, if either one of the assignments is removed, the user group is still assigned to the object because of the other assignment.

You can see all inherited and explicit assignments in the object's security screen.

Users

Enroll each person who will use Oracle LSH in any way by defining a user account for him or her. The user account includes a user ID, encrypted password, email address, and at least one application role.

Application roles allow the user to view a portion of the Oracle LSH user interface (UI) or to perform special operations; see "User Interface Security" on page 4-8. A user's assigned application roles apply across all Oracle LSH Domains and objects; they are not associated with user groups.

Assigning Users to User Groups A person's assignments to user groups and the roles to which he or she is assigned within those groups determine which operations the person can perform on which objects; see "About Security Design in the Oracle Life Sciences Data Hub" on page 4-1. A user can belong to any number of user groups, and can have any number of object security roles within a group. A user can have different roles in different user groups. A user must have at least one role within each group. The group administrator assigns users to groups and to roles within groups.

User Interface Security

Oracle LSH ships with predefined application roles that allow access to portions of the Oracle LSH user interface. Application roles apply to the user across the Oracle LSH application; they are not user group-specific. When you create a user account, you must assign it at least one application role so that the user can see at least part of the Oracle LSH user interface.

There are two types of user interface roles:

- Application Roles allow users to work with Oracle LSH on an ongoing basis.
- Administrator Roles are required for the initial setup of Oracle LSH. Some administration roles are also required for the ongoing maintenance of Oracle LSH.

Application Roles

Most Oracle LSH users need one or more of the predefined application roles to perform their work in Oracle LSH. The predefined Oracle LSH application roles are:

LSH Consumer Users assigned the LSH Consumer role have access to the Home and Reports tabs in the Oracle LSH user interface. Assign this role to users who need to retrieve information from Oracle LSH and/or run Oracle LSH applications to load and transform data. Oracle LSH object security determines which operations these users are allowed to perform on particular applications and reports.

LSH Definer Users assigned the LSH Definer role have access to the same tabs as the LSH Consumer plus the Applications tab. Assign this role to users who must build, test, or validate Oracle LSH applications. Oracle LSH object security determines which operations these users are allowed to perform on particular defined objects and outputs.

LSH Data Blind Break User Only the LSH Data Blind Admin can assign this role to users. This role does not provide access to any tabs in the Oracle LSH user interface. Users assigned to this role typically also have the LSH Consumer role.

Users with this application role can do the following, if they have normal Oracle LSH object security access and the blinding-related object security privilege noted:

- Run a job on Tables with a Blinding Status of Blinded that displays the real data, not the dummy data (also requires an object security role with the Blind Break operation on Table instances)
- View the output(s) generated by a job run on real, blinded data (also requires an object security role with the Blind Break operation on outputs)

LSH Data Unblind User Only the LSH Data Blind Admin can assign this role to users. This role does not provide access to any tabs in the Oracle LSH user interface. Users assigned to this role typically also have the LSH Consumer role.

Users with this application role can do the following, if they have normal Oracle LSH object security access and the blinding-related object security privilege noted:

- Permanently unblind data in Table instances (also requires an object security role with the Unblind operation on Table instances)
- Run a job on real data in Table instances whose status is Unblinded (also requires an object security role with the Read Unblind operation on Table instances)
- Change the status of an output from Blinded to Unblinded (also requires an object security role with the Unblind operation on outputs)
- Read outputs with a status of Unblinded (also requires an object security role with the Read Unblind operation on outputs)

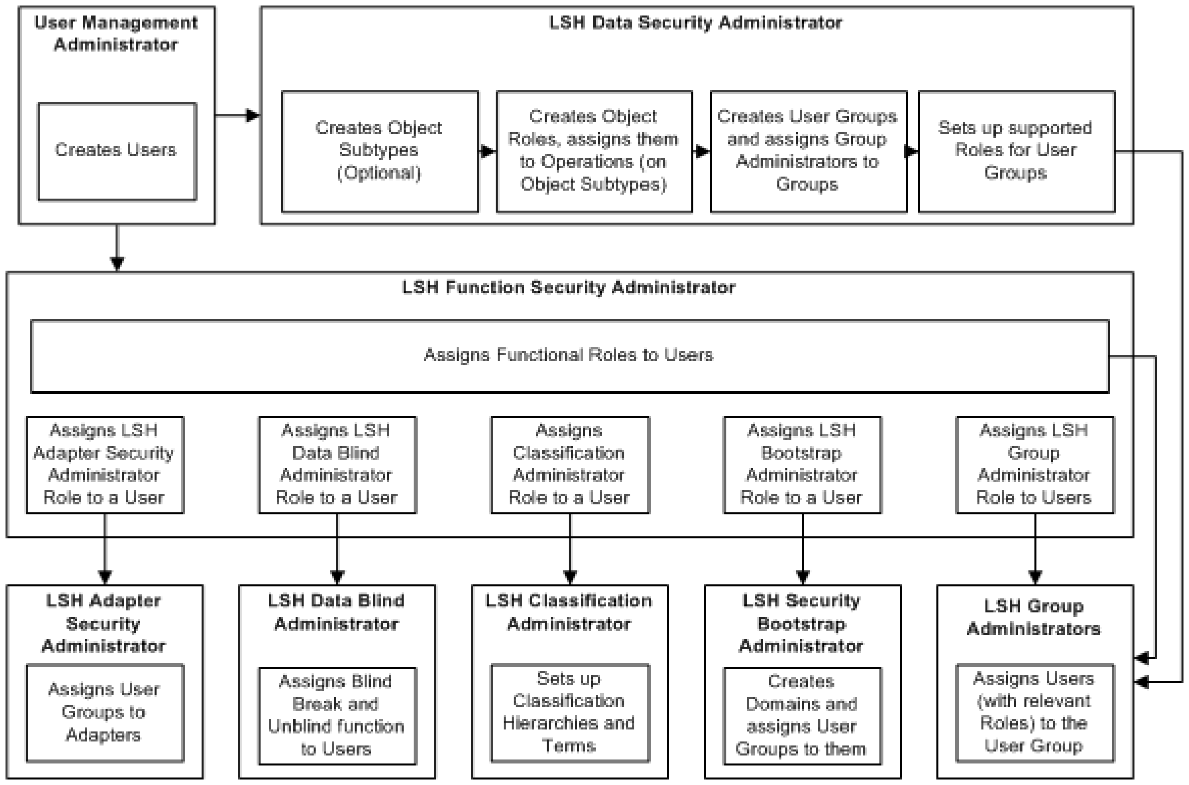
Administrator Roles

To provide the most secure environment, the tasks required to set up Oracle LSH are logically grouped, and each group of tasks must be performed by a user with a different administrative application role.

The System Administrator role is automatically created during Oracle LSH installation. Your designated system administrator must log on using this account, create at least one user account and assign the LSH Setup Admin role to the account (or create at least two user accounts and assign the two Setup Admin component accounts—LSH Data Security Admin and LSH Function Security Admin—separately).

Figure 4–4 shows the flow of Oracle LSH security tasks.

Figure 4–3 Oracle LSH Security Adminstrator Tasks


User Management Administrator
Creates Users
LSH Data Security Administrator
Creates Object Subtypes (Optional)
Creates Object Roles, assigns them to Operations (on Object Subtypes)
Creates User Groups and assigns Group Administrators to Groups
Sets up supported Roles for User Groups
LSH Function Security Administrator
Assigns Functional Roles to Users
Assigns LSH Adapter Security Administrator Role to a User
Assigns LSH Data Blind Administrator Role to a User
Assigns Classification Administrator Role to a User
Assigns LSH Bootstrap Administrator Role to a User
Assigns LSH Group Administrator Role to Users
LSH Adapter Security Administrator
Assigns User Groups to Adapters
LSH Data Blind Administrator
Assigns Blind Break and Unblind function to Users
LSH Classification Administrator
Sets up Classification Hierarchies and Terms
LSH Security Bootstrap Administrator
Creates Domains and assigns User Groups to them
LSH Group Administrators
Assigns Users (with relevant Roles) to the User Group

The Oracle LSH Administrator roles are:

LSH Data Security Admin The LSH Data Security Administrator has access to the Security tab in the Oracle LSH user interface and works with user groups, subtypes, and roles to set up object-related security.

LSH Function Security Admin The LSH Function Security Administrator works in the Oracle Applications UMX user interface to register users (create user accounts) and assign application roles to them.

LSH Security Admin LSH Security Administrator is a predefined composite role that combines the functions of the LSH Data Security Admin and LSH Function Security Admin roles. To allow the same person to perform both functions, assign just the LSH Security Admin role.

LSH Classification Admin The LSH Classification Administrator has access to the Classifications tab in the Oracle LSH user interface and can create classification hierarchies and add terms to them.

LSH Data Blind Admin The LSH Data Blind Administrator can assign the LSH Data Blind Break User role and the LSH Data Unblind User role to users.

LSH Group Admin Every Oracle LSH user group must have at least one LSH Group Administrator. The Group Administrator can add and remove users from the user group and change users' role assignments within the group.

LSH Bootstrap Admin The LSH Bootstrap Administrator can create Domains and assign user groups to the Domains. This is the first step in creating an organizational structure.

LSH Super User The LSH Super User has access to all user interface tabs in LSH: Home, Applications, Reports, Classification, Security, and Administration, as well as the Oracle Applications user and role screens. The Super User role does not include the Bootstrap role or the blinding-related roles.

Security for Blinded Data

This section contains the following topics:

- About Data Blinding, Blind Breaks, and Unblinding on page 4-11
- Blinding-Related Security Privileges on page 4-12
- Interaction of Blinding Statuses and User Privileges on Executing Jobs and Viewing Data on page 4-14

About Data Blinding, Blind Breaks, and Unblinding

Some sensitive clinical data must be protected from viewing during the course of a clinical trial; for example, treatment codes that would reveal which patients were receiving which drugs.

Oracle LSH supports data blinding in Oracle LSH Table instances and in all reports generated using one or more blinded Table instances as a source. Users with special privileges can unblind data (for example, at the end of a trial) or break the blind temporarily as required during the trial. Separate privileges are required to run and view reports generated on blinded and unblinded data.

All Table instances have a Blinding flag. You must set this flag to Yes to indicate that the Table instance may contain blinded data. If the Blinding flag is set to Yes, Oracle LSH maintains two sets of rows: one set for the real data and another set for dummy data, effectively partitioning the table in the database.

When a user loads data or runs a Program that reads from or writes to a Table instance whose Blinding flag is set to Yes, he or she must indicate in the submission form whether the job is running on real, blinded data or on dummy data. The system reads from and writes to the partition the user indicates. The options for the particular user submitting the job are limited by his or her own blinding-related security privileges; see "Blinding-Related Security Privileges" on page 4-12.

Note: Oracle LSH cannot ascertain whether data in an external system requires blinding or not. You must set up your security system so that only people who understand the issues and the source data can run Load Sets that may load sensitive data.

Table instances also have a Blinding Status attribute. Table instances whose Blinding flag is set to Yes can have a Blinding Status of either Blinded or Unblinded to indicate the current state of the data. Table instances whose Blinding flag is set to No can have a status of either Not Applicable (the default) or Authorized, which makes it possible for a nonblinded Table instance to be the target of a Program that reads from one or more blinded Table instances; see "Reading from Blinded Table Instances and Writing to Nonblinded Table Instances" on page 4-15.

Oracle LSH allows you to keep data securely blinded and also allows the following:

- **Blind Break**. A blind break is access to real, sensitive data that remains blinded to other users; it is a single execution of a job that reads from or writes to one or more blinded or unblinded Table instances. The Program may produce a report that displays the real, sensitive data. Even the log file of the job may contain sensitive information. One privilege is required to execute the job, and another privilege is required to view any resulting outputs. All blinded Table instances involved remain blinded and the same privilege is required to run the same executable again on the real data.

  The Blind Break privilege also allows access to real, blinded data through the Browse Data feature, through an integrated development environment (IDE), and through data visualizations; see "Blinding-Related Security Privileges" below.

- **Unblind**. Users with unblind privileges can permanently unblind data in a Table instance. They can also reblind the Table instance if necessary. A separate privilege is required to view reports generated on unblinded data.

For further details on how Oracle LSH handles blinded data, see "Execution and Data Handling" in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Blinding-Related Security Privileges

This section contains the following topics:

- Blinding-Related Object Security Operations on page 4-12
- Blinding-Related Application Roles on page 4-14

To perform most operations on sensitive data, a user must have normal security access to the particular Table instances or outputs that contain the sensitive data, a specific blinding-related operation on those objects, and a specific blinding-related application role.

Blinding-Related Object Security Operations

This section contains the following topics:

- Operations on Table Instances on page 4-12
- Operations on Outputs on page 4-13

To perform a blinding-related operation on a Table instance or output, a user must be assigned a role that includes the relevant object security operation on Table instances or outputs of the relevant subtype within a user group assigned to the particular Table instance or output.

Operations on Table Instances Oracle LSH has the following blinding-related object security operations on Table instances:

- **Blind Break** allows access to real data in a Table instance whose Blinding Status is currently either Blinded or Unblinded as follows:
  - If a user has Blind Break privileges on all blinded and unblinded Table instances that serve as a data source or target for a job, the user can run the job on real data to produce an output and/or write data to one or more target Table instances.
  - If a user has Blind Break privileges on all blinded and unblinded Table instances in the underlying Business Area, the user can view real data when

he or she launches a data visualization tool such as Oracle Business Intelligence Enterprise Edition.

- If a user has Blind Break privileges on a blinded or unblinded Table instance, the user can choose to view real data in the Table instance using the Browse Data feature.
- If a user has Blind Break privileges on all blinded and unblinded Table instances mapped to a Program instance, the user can choose to view real data when he or she launches an integrated development environment (IDE) such as SAS to work on the Program instance.

Note: The functionality described in the last two points is intended for Definers, and users must also have the LSH Definer application role to see the Applications tab in the user interface and use these features.

- **Read Unblind** allows access to real data in the Table instances in all the same situations as Blind Break above, but only if all the relevant Table instances have been unblinded.
- **Read Data** allows access only to dummy data in the blinded and unblinded Table instances in all the same situations as Blind Break above.
- **Unblind** allows the user to unblind and reblind Table instances.

Note: Unblinding Table instances also requires the LSH Definer application role to see the Applications tab in the user interface.

Operations on Outputs Oracle LSH has the following blinding-related operations on outputs:

- **Blind Break** allows viewing outputs with a status of Blind Break (whose data was generated from, or written to, one or more Table instances whose Blinding Status was Blinded or Unblinded at runtime).
- **Read Unblind** allows viewing outputs with a status of Unblinded (whose data was generated from, or written to, one or more Table instances whose Blinding Status was Unblinded and no Table instances whose Blinding Status was Blinded at runtime).
- **Unblind** is designed to allow unblinding outputs but has not been implemented in Oracle LSH Release 2.4.8.

Note: In the Example of Security Design on page 4-16, roles allowing blinding-related operations are included only in the Production Work Area user group. Users assigned these roles can perform blinding-related operations on all Table instances and outputs in the Production Work Area.

If you want to restrict a user's unblinding privileges to a single Table instance, you must create a user group specifically for that Table instance with the necessary role and assign it to the few users who should have unblinding privileges on that Table instance.

Blinding-Related Application Roles

To perform blinding-related operations on Table instances or outputs, a user must have the relevant blinding-related application role in addition to the required blinding-related object security privilege.

While a user group administrator assigns blinding-related object security privileges to a user in the context of a particular user group, blinding-related application roles are assigned to users by a system administrator and apply across Oracle LSH; they are not limited to a particular user group, object subtype, or Domain, for example.

LSH Data Blind Break User A user must have the LSH Data Blind Break User application role in order to perform a Blind Break object security operation on Table instances or outputs; see "Blinding-Related Object Security Operations" on page 4-12 .

LSH Data Unblind User A user must have the LSH Data Unblind User application role to either unblind or reblind Table instances. (No application role is required for the Read Unblind operation on either Table instances or outputs.)

Interaction of Blinding Statuses and User Privileges on Executing Jobs and Viewing Data

The interaction of Table instance and output Blinding Statuses and blinding-related user privileges is summarized in Table 4–1 and described in detail below. The information in the column Data Available to User applies to users who are running jobs, using the Browse Data feature on a single Table instance, launching a visualization, or working on a Program in an IDE.

The information in the rightmost columns applies only to outputs generated by running jobs on the source Table instances (not to Browse Data, visualizations, or IDEs).

Table 4–1 Summary of the Interaction of Blinding Statuses and User Privileges on Viewing Data

Most Restrictive Blinding Status of Any Source Table Instance	Most Powerful User Privilege on All Relevant Table Instances	Data Available to User	Resulting Output Blinding Status	User Privilege Required on Output to View Output
Blinded	Blind Break (and the LSH Data Blind Break User application role)	Dummy or Real	If real data processed: Blinded If dummy data processed: Dummy	If real data processed: Blind Break (and the LSH Data Blind Break User application role) If dummy data processed: View
Blinded	Read Data or Read Unblind	Dummy	Dummy	View
Unblinded	Blind Break (and the LSH Data Blind Break User application role) or Read Unblind	Dummy or Real	If real data processed: Unblinded If dummy data processed: Dummy	If real data processed: Read Unblind If dummy data processed: View
Unblinded	Read Data	Dummy	Dummy	View
Not Applicable	Read Data, Blind Break, or Read Unblind	Not Applicable	Not Applicable	View

The following sections describe the information summarized in the above table.

Blinded Table Instances If the user has the Blind Break operation on all blinded and unblinded source and target Table instances and the LSH Data Blind Break User application role, he or she can choose to run a job on—or view—either dummy or real data and:

- Any output produced on real data has a Blinding Status of Blinded and the Blind Break operation on the output and the LSH Data Blind Break User application role are required to view the output.
- Any output produced on dummy data has a Blinding Status of Dummy and the View operation on the output is required to view the output.

If any of the source or target Table instances whose Blinding flag is set to Yes has a Blinding Status of Blinded, and the user has only the Read Data or the Read Unblind operation on any of these Table instances, the user can run a job on—or view in an IDE— only dummy data. Any outputs produced have a Blinding Status of Dummy. The View operation on outputs is required to view them.

Unblinded Table Instances If all of the applicable Table instances whose Blinding flag is set to Yes have a Blinding Status of Unblinded, and the user has the Read Unblind operation on all the unblinded Table instances or the Blind Break operation on all the unblinded Table instances (and the LSH Data Blind Break User application role), the user can choose to run a job on—or view—real or dummy data.

- Any output produced on real data has a Blinding Status of Unblinded and the Read Unblind operation on the output is required to view the output.
- Any output produced on dummy data has a Blinding Status of Dummy and the View operation on the output is required to view the output.

If any of the applicable Table instances whose Blinding flag is set to Yes has a Blinding Status of Unlinded, and the user has only the Read Data operation on any of these blinded Table instances, the user can run a job on—or view in an IDE—only dummy data. Any outputs produced have a Blinding Status of Dummy. The View operation on outputs is required to view them.

Nonblinded Table Instances If all the applicable Table instances have their Blinding flag set to No and their Blinding Status is Not Applicable, the Table instances contain real data that is not sensitive and does not required blinding. The user requires only the Read Data operation on the Table instances to run a job on—or view—the data. Any output produced has a Blinding Status of Not Applicable, and the View operation on the output is required to view the output.

Reading from Blinded Table Instances and Writing to Nonblinded Table Instances

In rare cases you may need to run a Program that reads from one or more blinded or unblinded Table instances and writes to one or more nonblinded Table instances (whose Blinding flag is set to No). Executing such a job fails unless:

- Every target Table instance whose Blinding flag is set to No has a Blinding Status of Authorized.
- The user running the Program has Blind Break privileges on all blinded source and target Table instances and either Blind Break or Unblind privileges on all unblinded source and target Table instances.
- The user running the Program explicitly authorizes running the Program after a warning that the Program reads from one or more blinded or unblinded Table instances and writes to one or more nonblinded Table instances.

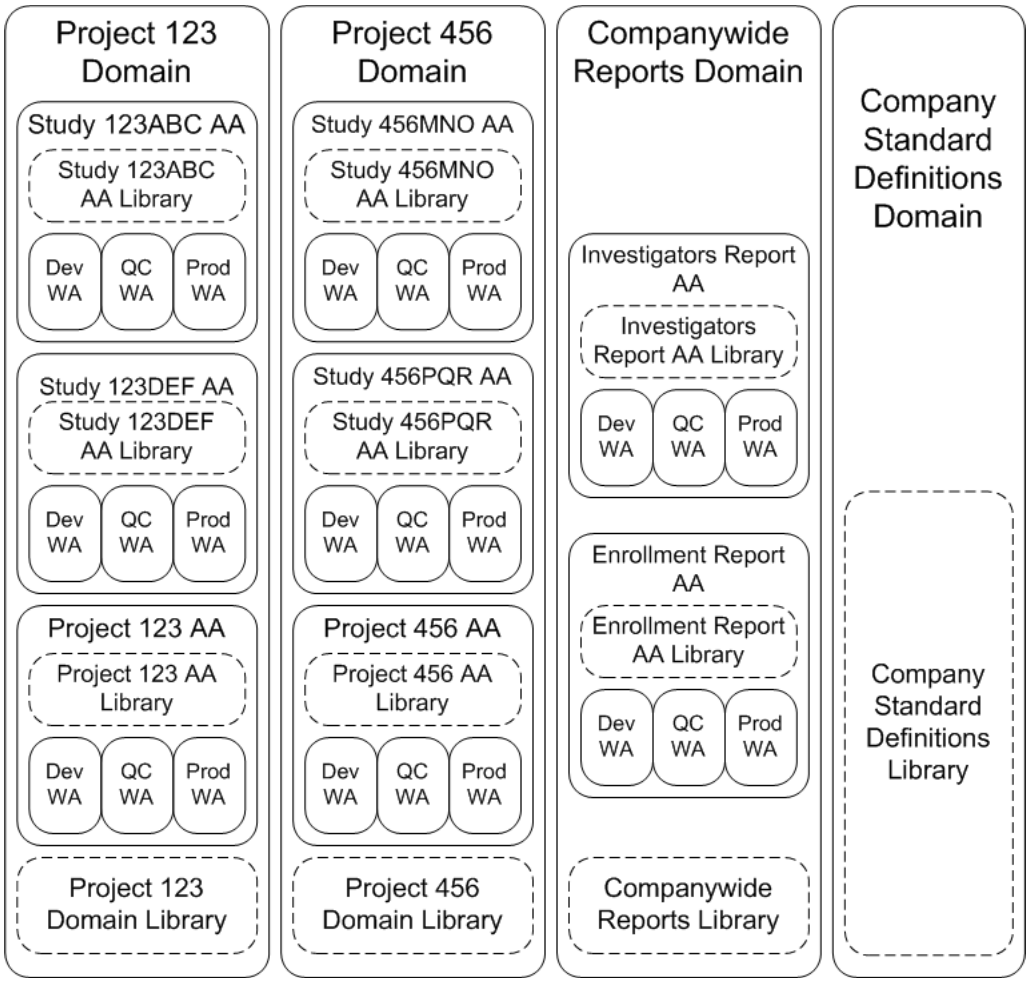
Example of Security Design

In this example, pictured below, are the following Domains:

- One Domain for each drug project. Inside each project Domain is a library of standard definitions for use within that project, an Application Area for each study, plus another Application Area for projectwide analysis. Within each Application Area are three Work Areas, one each for development, quality control, and production.
- One Domain for companywide reports with two Application Areas: one Application Area for the report listing all investigators and another for companywide enrollment reports. A Domain library contains the standard companywide report definitions
- One Domain for a library of standard company definitions.

In general, there is a user group defined for each organizational structure (Domain, Application Area, and Work Area). See "Example User Groups" on page 4-16.

Figure 4–4 Organizational Structure for Security Design Example



Example User Groups

Following is information on each user group used in this example, including a list of the roles assigned to each.

Project 123 Domain User Group

The Project 123 user group is assigned to the Domain that includes all of Project 123.

The privileges granted to a user through his or her role(s) in a user group assigned to a Domain apply by default to the Application Areas and Work Areas in the Domain, unless the user group is explicitly unassigned from a particular container.

Therefore, the roles assigned to the Domain user group should allow only the operations required to allow the proper use of the object definitions in the Domain library. See "Domain Libraries" on page 3-4 for a discussion of the recommended usage of Domain Libraries.

The Project 123 user group includes the following two roles:

- **Librarian/Project Programmer**. The Librarian (or Project Programmer) role allows maintenance—including creation, modification, and deletion— of the object definitions of all types in the Domain library. The Create operation is required to move an object into the library.

  We recommend that these definitions be developed in the Domain's Application Areas and moved into the library only after they are stable, validated, and approved for reuse. They should then need very little maintenance. Assign this role to only a few people.

- **View All Definitions**. All Programmers and Data Managers working on Project 123 need the View operation on all object definitions in the Project 123 Domain library so that they can reuse those definitions. Therefore, all users who have the Programmer or Data Manager role for any study in the Project 123 Domain also belong to this user group with the role View All Definitions.

Study 123ABC Development User Group

The Study 123ABC user group is assigned to the Study 123ABC Application Area.

Programmers work on object instances in a Work Area, but when they create a new object, Oracle LSH automatically simultaneously puts the new object definition in the Application Area that contains the Work Area, and an instance of it in the Work Area. Therefore, to be able to develop new object definitions for a study, Programmers need Create and Modify privileges on object definitions in the Application Area as well as on object instances in the Work Area. They also need View privileges on each object subtype.

Note: If you develop a set of standard modular object definitions, you may be able to develop applications entirely by reusing those definitions. In that case, some users may require only Create, Modify, and Delete privileges on object instances in the Development Work Area and View privileges in the library of standard definitions.

The privileges granted to a user through his or her role(s) in a user group assigned to an Application Area apply by default to all the Work Areas in the Application Area, unless the user group is explicitly unassigned from a Work Area.

In this example this user group is explicitly unassigned from the Quality Control and Production Work Areas so that Definers cannot modify on objects in those Work Areas. Definers work in the Development Work Area only. See "Work Area Usage Intent and Validation Status" on page 6-5.

The Study 123ABC user group includes the following roles:

Programmer The Programmer role allows all the creation of all object definition types in an Application Area and modification operations on object definitions and instances of all types, including Modify Validation Status QC. The Programmer role has these same privileges on object instances types in a Work Area. The Programmer can therefore create new object definitions and instances or create new instances of existing definitions, test them, and promote them to Quality Control after successful testing.

Data Manager The Data Manager role allows the same operations as the Programmer role, but has Modify Validation Status Production (instead of QC), which allows promotion to both QC and Production. In addition, a Data Manager can clone Work Areas. For this, the Data Manager must have the Modify rights on Work Areas as well as Clone.

Study 123ABC Quality Control User Group

The Study 123ABC Quality Control user group is assigned to the Quality Control Work Area in the Study 123ABC Application Area. Its purpose is to allow Quality Control engineers to test applications but not to modify objects.

Quality Control Engineer The Quality Control role allows the View operation on object instances of all types, and the Submit operation on object instances of all executable object types. Therefore Quality Control engineers can run all executable objects in any Work Area in the Application Area and view the resulting outputs, but they cannot modify any objects.

Data Manager The Data Manager role allows all operations on all object types. Because this user group is assigned to a Work Area, only the operations on object instances are relevant. The Data Manager can modify object instances so that they point to the new version of a fixed object. The Data Manager has the IQ_Submit role needed to run executables to test them before promoting object instances to the Quality Control validation status. The Data Manager can also clone the Work Area.

See "Work Area Usage Intent and Validation Status" on page 6-5 for information on the intended usage of Works Areas and the interaction of objects' validation status and the Work Area Usage Intent attribute.

Note: In this example, the Study 123ABC user group is explicitly unassigned from the Quality Control Work Area so that Programmers cannot change object instances in the QC environment. Programmers should make any necessary changes in the Development Work Area, and then the Data Manager should clone the modified Work Area onto the Quality Control Work Area for testing.

Alternatively, if your operating procedure is for your Definers to test their own applications, you do not need the Quality Control user group. To allow programmers and data managers full privileges on the Quality Control Work Area, simply do not unassign the Development user group from it.

Study 123ABC Production User Group

The Study 123ABC Productionuser group is assigned to the Study 123ABC Production Work Area. The Production Work Area controls security access to the Oracle LSH schema that contains production data and outputs on production data. Therefore all personnel who need to see reports on production data need access to the Production Work Area.

This is the first user group in this example to differentiate between privileges on one subtype and another. In this security system, Programmers, Data Managers, and Quality Control Engineers can work on objects of any subtype; for example, Financial or Clinical Report Sets. However, the Production Work Area user group includes users who can see only Clinical object subtypes, users who can see only Financial object subtypes, and some users who can see both subtypes.

Note: In this example, the Study 123ABC Development user group is explicitly unassigned from the Production Work Area so that Programmers cannot change object instances in the production environment.

Some of the roles listed here have the same privileges on the same object types. From the point of view of Oracle LSH, a single role could be substituted for these. However, it may be easier to administer user accounts if you create roles for actual job titles in use in your company and assign them to the people who hold those jobs.

The following roles have Submit and View privileges on instances of Programs, Report Sets, Workflows, and Business Areas (for data visualizations). They also have the Read Data operation on Table instances so they can run executables on Table instances, and the View operation on outputs.

Most roles should include

- **Investigator**. Has operations on subtype Clinical Programs, Report Sets, Workflows, and Business Areas only.
- **Statistician**. Has operations on subtype Clinical Programs, Report Sets, Workflows, and Business Areas only.
- **Project Manager**. Has operations on both subtypes: Clinical and Financial Programs, Report Sets, Workflows, and Business Areas.
- **Blind Break Manager**. Has operations on subtype Clinical Programs, Report Sets, Workflows, and Business Areas only.

The following roles have Submit and View privileges on Execution Setups of Load Sets and Data Marts in addition to Programs, Report Sets, Workflows, and Business Areas:

- **Trial Manager**. Has operations on both subtypes, Clinical and Financial, for each object type.
- **Submission Officer**. Has operations on only the Clinical subtype of each object type.

Three additional roles allow certain operations on blinded data. By creating a separate role for this purpose, you can limit this sensitive privilege to the minimum number of people required to perform it in your organization. You can assign these roles to users who also have other roles. See "Security for Blinded Data" on page 4-11.

Note: These roles are different from the LSH Data Blink Break User application role and the LSH Data Unblind User application role. One of those application roles is required in addition to these object-specific roles to create a blind break or unblind data. See Table 4–1, " Summary of the Interaction of Blinding Statuses and User Privileges on Viewing Data" on page 4-14.

In addition, the user must be assigned to one of these roles in a user group with access to the object.

- **Blind Break**. The Blind Break role includes the Blind Break operation on Table instances and on outputs. The Blind Break role allows a user to run a Program or other executable on a Table instance with a Blinding Status of Blinded and view the resulting outputs (see "Security for Blinded Data" on page 4-11).
- **Permanently Unblind**. The Permanently Unblind role includes the Unblind operation on both Table instances and outputs. The Unblind operation allows the user to change the Blinding Status of the Table instance or output from Blinded to Unblinded.
- **View Unblinded Data**. The View Unblinded Data role includes the Read Unblind operation on both Table instances and outputs. Even after a Table or output has been permanently unblinded, this role is required to see the unblinded data.

User Groups for the Other Domains, Application Areas, and Work Areas

You may be able to use the user groups described in this example as standard user groups for all Domains, Application Areas, Quality Control Work Areas and Production Work Areas. Even the users in the user groups may be the same in some cases—especially in groups intended for Programmers and Data Managers.

You can copy user group definitions in two ways:

- Copy just the definition, including the name, description, and assigned roles.
- Copy all of the above plus all the users assigned to the user group, with their role assignments.

For example, the same simple, two-role Domain user group structure may be appropriate for all your Domains. In this case, you have the following choices:

- If exactly the same people should have access to two Domains, with the same roles in each Domain, you can assign the same user group to both Domains.
- If most of the same people should have access to two Domains, with the same roles in each Domain, you can copy the first Domain user group you create, including the users assigned to the group. You assign it to the second Domain and the Group Administrator can then make any necessary changes to users and users' role assignments.
- If different people should have access to two Domains, you can copy the first Domain user group you create, using the option to not include any of the users. You can assign it to the second Domain. The Group Administrator then assigns users to roles within the group.

If the user group definition itself needs to be different from other Domains, you can define an entirely new user group or copy an existing one and make any necessary changes.

Example Users

In this example, you might have the following users:

- Karl Martin. Karl is a **programmer** who works on both studies in Project 123. He is assigned to the following user groups: Project 123 Domain user group, Study 123ABC Development user group, and Study 123DEF Development user group.

  In the Domain user group he has the View All Definitions role. In the two study user groups he has the Programmer role.

- Sylvia Green. Sylvia is the **data manager** for Project 123. She is assigned to the following user groups: Project 123 Domain user group, Study 123ABC Development user group, Study 123ABC Quality Control user group, Study 123ABC Production user group, Study 123DEF Development user group, Study 123DEF Quality Control user group, Study 123DEF Production user group.

  In the Project 123 Domain user group she has the Librarian role. In the other user groups she has the Data Manager role.

- Sanjay Rao. Sanjay is a **statistician** working on both Project 123 and Project 456. He is assigned to the Production Work Area user group for all studies in Project 123 and all studies in Project 456.

  In each user group he has the Statistician role, allowing him to see and execute Clinical Programs, Report Sets, Workflows, and Business Areas. He also has the Permanently Unblind role and the LSH Data Unblind User application role, allowing him to permanently unblind study data and view the results.

- Sining Chen. Sining is an **investigator** for Study 123ABC. He is assigned to the Study 123ABC Production Work Area user group with the Investigator and Data Blind Break roles.

  He can see reports and create data visualizations for Study 123ABC and create a Blind Break during the trial if necessary.

Design Decision Summary

To design a security system, you must complete the following tasks:

- Design an Organizational Structure on page 4-21
- Design a Set of User Groups on page 4-22
- (Optional) Design One or More Object Subtypes on page 4-22
- Design a Set of Roles and their User Group Assignments on page 4-23

Design an Organizational Structure

Take advantage of the fact that user group assignments are inherited by objects from their container. Design an organizational structure of Domains, Application Areas, and Work Areas where you can use this feature and avoid excessive manual assignments of user groups to containers and objects.

Restrictions None.

You can define any organizational structure you choose; see Chapter 3, "Designing an Organizational Structure".

Considerations Take the following into consideration:

- You will assign user groups to the container objects in your organizational structure, so design an organization that reflects the logical units that the same group of users need to access.
- You can explicitly unassign and reassign user groups to containers and other objects at any time.

Design a Set of User Groups

Design a set of user groups in conjunction with your organizational structure design.

See "User Groups" on page 4-6 for further information.

Restrictions None.

You can define any number of user groups. You can assign any number of roles to user groups.

Considerations Take the following into consideration:

- By default, all objects are automatically assigned the same user groups as their containing object.
- You can explicitly unassign and reassign user groups to objects at any time.

(Optional) Design One or More Object Subtypes

Oracle LSH ships with a set of predefined primary object types, including definitions and instances of Tables, Programs, Load Sets, Report Sets, Data Marts, Workflows, and Business Areas. Oracle LSH also ships with one predefined subtype for each of these object types.

A role consists of a set of allowed operations on object subtypes. If you define additional subtypes for an object type, you can allow different roles to have operations allowed on different subtypes. For example, you might want different people to be able to view clinical reports and financial reports, in which case you can create Clinical and Financial Report Set Output subtypes.

See "Object Subtypes" on page 4-3 for further information.

Restrictions None.

You can define any number of subtypes for any primary object type.

Considerations Take the following into consideration:

- Use subtypes for broad categories of information separate from the logical categories you use for your organizational structure and user groups, but which also differentiate which users should have access to which objects. For example, if your organizational structure and user groups use Project and Study as their primary logical units, you could define subtypes of Clinical, Financial, and Administrative, which apply to all projects and studies.
- If you are using Oracle LSH only for clinical data, you probably do not need additional subtypes. If you decide to use Oracle LSH for other types of data in the future, you can add subtypes then.
- Objects inherit the subtype of their container object, if a subtype with the same name is defined for their own object type. Objects can also inherit classifications from their container objects, if you choose. To make object creation simpler, create a subtype with the same name for all object types, or at least all container and

primary objects (Domains, Application Areas, Work Areas, Programs, Tables, Load Sets, Report Sets, Data Marts, and Business Areas). See "Design Decision Summary" on page 5-14 in Chapter 5, "Designing a Classification System for Searching and Browsing" for additional considerations.

Design a Set of Roles and their User Group Assignments

A role definition consists of a name, description, and a set of allowed operations on one or more object subtypes. You assign roles to user groups and users to roles within user groups. A single user can have any number of roles, and can have different roles in different user groups.

See "Roles" on page 4-5 for further information.

Restrictions None.

You can define any number of roles with any number of operations on object subtypes assigned to each role.

Considerations Take the following into consideration:

- To simplify assigning users to roles, create roles that correspond to the actual jobs in your company.
- You may want to create a few roles specifically to do certain Oracle LSH tasks, especially blinding-related tasks. You can then assign just a few users to those roles to keep blinded data as secure as possible.

5 Designing a Classification System for Searching and Browsing

This section includes information on the following topics:

About Classification

A typical clinical trial requires the development of many Programs and other defined objects, and repeated executions of each one, so that by the end of the trial there are large numbers of objects and outputs such as reports.

The Oracle Life Sciences Data Hub (Oracle LSH) classification system allows you to organize these objects and outputs by classifying, or labeling, them in ways that reflect your business or clinical organization, or that identify any characteristic you choose. Oracle LSH uses classifications to help users find outputs and defined objects in two ways: Browsing and Searching.

Browsing Oracle LSH uses the classification system you set up to create a customized user interface in the Reports tab for both the Outputs and Visualizations subtabs.

You can create hierarchies of categories that are logically related to one another, so that Oracle LSH displays outputs within that hierarchical structure, like the folder structure on a PC.

For example, if you have multiple projects, multiple studies in each project, and multiple sites in each study, you can create a hierarchy with three levels: Project, Study, and Site. Enter all your actual project names in the Project level, all study names in the Study level, and all site names in the Site level, noting which sites are part of which studies and which studies are part of which projects.

In the Reports tab, Oracle LSH displays a folder for each project, labeled with the project name. Each project folder contains a folder for each of the studies in that project as well as all reports classified to that project as a whole. Each study folder contains a folder for each of the sites in that study, as well as all reports classified to that study as a whole. Each site folder contains all reports classified to that site.

Alternatively, within each study you could have folders for reports on demography and adverse events reports, for example.

Note: Oracle LSH executable objects can produce outputs as follows: Programs can produce reports; Report Sets generate sets of reports; Data Marts produce files in text, SAS data sets or transport files, or Oracle Export format containing large amounts of data; and SAS and text Load Sets may produce the text or SAS file loaded as an output. All executables can also produce log files, but these are not displayed in the Reports screen directly; they are available as a link from the main screen for each output.

Searching Oracle LSH allows users to search for outputs and defined objects based on classifications, as well as other criteria. Definers need to search for existing object definitions they can reuse, either exactly as they are or with modifications. Consumers need to find reports and other outputs that contain information they need. They may also need to find submission forms to generate outputs on new data, or to load data into Oracle LSH.

The Simple Search feature allows using a single classification value as a search criterion. Advanced Search allow using multiple classification values.

Classification Setup To implement Oracle LSH classification, you must do the following:

- **Define at least one classification hierarchy** that contains information relevant to your business or clinical organization—any information that characterizes Oracle LSH objects in a way that might help a user find an object in Oracle LSH (see "Defining Classification Hierarchies" on page 5-3).
- **Define classification values** at each level of each hierarchy, with relations between values; for example, in the Project > Study > Site hierarchy, put all project names in the Project level, all study names in the Study level, each with a relation to its project, and all site names on the Site level, each with a relation to its study or studies (see "Defining Classification Values" on page 5-5).
- **Assign classification hierarchy levels to object subtypes**, specifying whether the classification to each level is mandatory for objects of that subtype and whether classification by inheritance from the owning object is allowed (see "Assigning Classification Levels to Object Subtypes" on page 5-6).
- **Assign classification values to objects**, starting with Domains, Application Areas and Work Areas. These classifications can be automatically inherited by objects within these containers if you allow classification by inheritance for their object subtypes. Definers can change inherited classifications for individual objects as necessary. The new classification applies to all versions of the object and carries forward to new versions unless a new version is explicitly reclassified.

Instructions are included in "Setting Up the Classification System" in the *Oracle Life Sciences Data Hub System Administrator's Guide*.

Designing a Classification System

You should design your classification system in conjunction with designing your organizational structure and security system:

- **Organizational Structure**. When you choose to use classification by inheritance, objects automatically receive classification value(s) for a particular hierarchy level from their container. Take advantage of this functionality by designing an organizational structure that reflects the way you want outputs to be displayed on screen.

  You may want to make classifications inherited for all object types except the organizational objects (Domains, Application Areas, and Work Areas.)

  Outputs can inherit classifications from the executable object instance that generates them, which in turn can inherit classifications from the Work Area, which in turn can inherit classifications from the Application Area (see Appendix B, "Object Ownership"). Therefore, if you use the Project > Study > Site hierarchy, classify an Application Area to Study ABC123, and allow inheritance for the Study level for all object subtypes, all reports generated within the Application Area will be classified to Study ABC123 and displayed in the Study ABC123 folder on the Reports screen.

- **Subtypes and the Security System**. Each object type has one predefined subtype called Default. You can create additional subtypes if you choose to, but this is optional. For example, if you plan to store and analyze financial information in Oracle LSH, you could create a subtype called Financial. When you set up your classification system, you can allow or require that objects of a particular subtype have be classified at a particular hierarchy level. Financial object subtypes might be classified to the same or different hierarchy levels as Default or Clinical object subtypes; for example, you might have a financial hierarchy with levels called Fiscal Year > Quarter.

  When you set up your security system, you assign roles to operations on object subtypes. If you plan to store and report on financial information in Oracle LSH, you probably want different people to see financial reports and clinical reports. If you create a Financial subtype, you can assign different roles to the View operation on Financial outputs than to the View operation on Clinical outputs. Therefore, take both your classification and security systems into consideration when deciding whether to use subtypes; see "Object Subtypes" on page 4-3.

As you begin the design process, focus first on the hierarchies you will use to display outputs in the Reports window. These hierarchies are also available for use in searching. Next, if you wish, develop additional hierarchies for use in searching only (see "Keyword Hierarchies" on page 5-4).

See "Classification Example" on page 5-9 and "Classifying Outputs" on page 5-9.

Defining Classification Hierarchies

You must define at least one hierarchy to create a user interface for browsing for outputs. You can define any number of additional multi- and single-level hierarchies.

Multilevel Hierarchies

Multilevel classification hierarchies consist of ranked, logically related levels with a strictly ordered parent/child relationship, represented in a linear sequence such as Project > Study > Site > Patient. A parent level can have only one child level, and a child level can have only one parent level.

Oracle LSH uses multilevel hierarchies to display outputs on the Reports screen. You can also use multilevel hierarchies to search for outputs and defined objects.

Single-Level "Hierarchies"

Some types of information may not be hierarchical by nature. In this case, you can still define a "hierarchy," but it can consist of a single level. In searches, single-level hierarchies function as a list of values.

Keyword Hierarchies You can designate a single-level hierarchy as a keyword hierarchy. Oracle LSH does not use keyword hierarchies to display outputs or submission forms for browsing, but does allow using them to search for outputs and defined objects. Using Advanced Search, users can search on more than one hierarchy value at a time. Use keyword hierarchies to help users add search criteria to narrow their search results and find the object definitions and outputs they need. You can create keyword hierarchies especially for a particular kind of object or output.

For example, create several keyword hierarchies to use for Program definitions, such as technology types (SAS, PL/SQL, and Oracle Reports), the basic function of the Program (transforming, reporting, or both), and the tables the Program reads from (Demography, Adverse Events, Concomitant Medications, and so on). When a Definer searches for a Program definition, he or she can use any number of these keywords as search criteria. See "Classification Hierarchies for Use in Searching Only" on page 5-13 for more examples.

Nonkeyword Single-Level Hierarchies You may want to put outputs and submission forms into several categories that are not hierarchical and use them for browsing on screen. In this case, define a hierarchy with a single level and do not designate it a keyword hierarchy.

Defining Hierarchy Levels

You must define at least one hierarchy level for each classification hierarchy. If the hierarchy has more than one level, each level must be logically related so that the topmost level contains the most general values and the bottommost level contains the most specific ones.

You must make the following design decisions for each hierarchy level:

- **Set the Allow Classification Flag**. If set to Yes, this level can be assigned to object subtypes, which allows objects of that subtype to be classified to values contained in this hierarchy level. (When you define object subtypes, you specify whether the hierarchy level is allowed, not allowed, or required for each particular object subtype; see "Assign Hierarchy Levels to Object Subtypes" on page 5-14.) The default setting is Yes.

  If set to No, the hierarchy level still exists and serves to organize classification values in the levels below. For example, you might want to create a Project > Study > Site > Patient hierarchy even if you do not generate reports that are specific to a single site. In this case, you can set the Allow Classification Flag to No for the Site level. In the Reports screen, folders for each site are displayed within each Study folder. Each site folder contains a folder for each patient. Any patient-specific reports, such as for a serious adverse event, are contained in a patient's' folder.

- **Set Term Uniqueness**. This setting determines the way Oracle LSH enforces the uniqueness requirement for classification values in the level. This attribute has these possible settings:

- **Unique Within Level**. Each classification value contained in this level must be unique relative to all other classification values in the level.

  For example, there can be only one Site called General Hospital in the Site level. Because each child classification value can have only one parent value, General Hospital can be associated with only one study.

- **Unique Within Parent**. Each classification value contained in this level must be unique relative to the other classification values in the level that are related to the same parent classification value.

  In this case, if the site General Hospital is participating in more than one study, the classification value General Hospital can appear multiple times in the Site level, if it is associated with a different study each time (see Figure 5–1, "Example of Related Classification Values within a Multilevel Hierarchy Structure" on page 5-5).

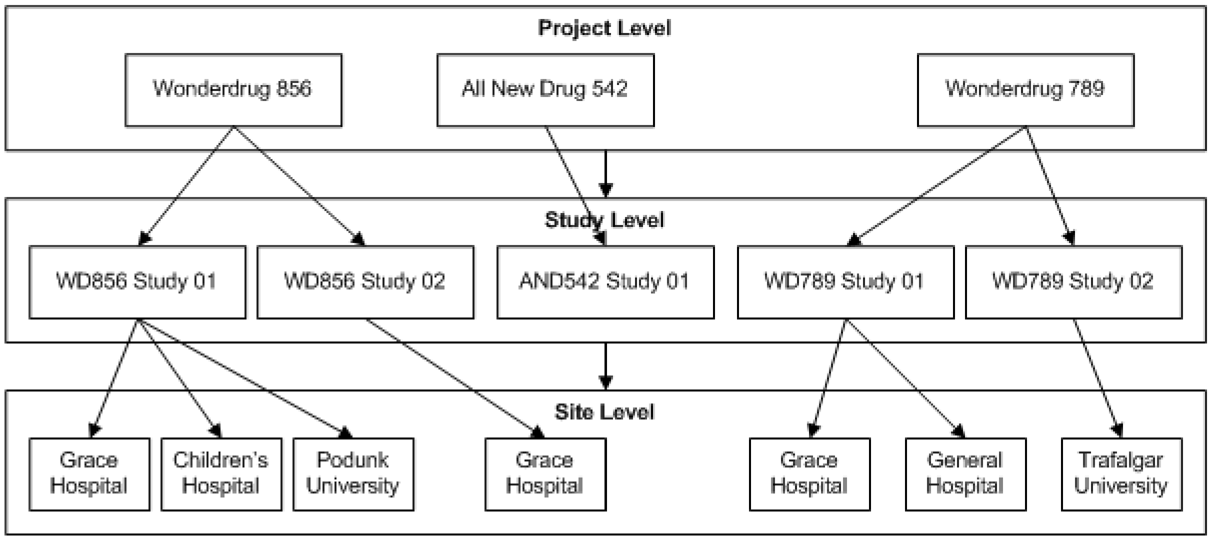
Defining Classification Values

A classification value is the actual name used in your organization for a particular item in the category represented by the classification level; for example, the name of a particular project in the Project level.

In a multilevel hierarchy, classification values have relations to logically related classification values in the levels above and below their level. Each classification value must have exactly one related term in the level above and may have zero or more related terms in the level below.

In a single-level hierarchy, classification values do not have defined relations with any other classification values. They may or may not be logically related to each other.

Figure 5–1 Example of Related Classification Values within a Multilevel Hierarchy Structure



In the example above, the Term Uniqueness attribute of the Site level is set to Within Parent so that the same site can appear in the Site level more than once, related to more than one study. In the example, both General Hospital and Grace Hospital appear twice, for two different studies.

Assigning Classification Levels to Object Subtypes

This section includes the following topics:

About Subtypes and Classification

Oracle LSH ships with predefined object types, including Program definitions, Program instances, Table definitions and instances, Report Set definitions and instances, and so on (for a complete list, see Appendix A, "Object Types with Operations").

Each time a Definer creates an object, he or she must select a subtype for it. One subtype (called "Default") is predefined for every object type. If you choose to, you can create additional subtypes for any object type. You must assign at least one classification level to each object subtype, including the default subtype.

You assign classification hierarchy levels—for example, Study in the Project > Study > Site hierarchy—to object subtypes. Definers can then classify objects of that subtype to that hierarchy level and find objects of that subtype by searching on that level's classification values. In the case of subtypes of Outputs and of Execution Setups, which become submission forms, the system displays them in the Reports screen by their classification values for that hierarchy level.

You may want to define subtypes specifically for the purpose of assigning different classification hierarchy levels to objects of the same type. For example, you could create a classification hierarchy Fiscal Year > Quarter and assign its levels to Outputs of subtype Financial but not Clinical. See "Object Subtypes" on page 4-3.

Making a Classification Level Mandatory

For each classification hierarchy level you assign to an object subtype, you must set the Mandatory flag to Yes or No.

Mandatory If you set it to Yes, then all objects created using that subtype must have at least one classification value for the hierarchy level. For example, if you assign the level Project > Study to the Clinical Output subtype as Mandatory, all Clinical outputs must be classified to at least one study name.

The Oracle LSH Reports tab display is arranged hierarchically by classification values. If an output is not classified at all, it does not appear in the Reports tab. Therefore you may want to make at least one classification level mandatory for outputs you want to see in the Reports screen. If you have set up an organizational structure and classification system to work together, you can set a value for Study, for example, at the Work Area level and make the Study level value inherited by all object instances and outputs, so that outputs automatically receive the Study value set for their Work Area (see "Allowing Classification by Inheritance" on page 5-7).

Note: If an output is not classified, you can still find it by searching on its name or other attributes.

Optional If you set the Mandatory flag to No, classifications to that hierarchy level are optional for objects of that subtype. If you assign classification values of that hierarchy level to an object of that subtype, the system allows using them in searching and, if the object is an output or submission form, displays it on screen in a folder named for the classification value. For example, if a report is classified to Wonderdrug 856 > WD856 Study 01, in the Outputs window the report is contained in the WD856 Study 01 folder, which is under the Wonderdrug 856 folder.

If you do not assign any classification values of that hierarchy level to an object of that subtype, the only effect is that those values are not available for use in searching and browsing.

In general, assigning hierarchy levels as optional gives you flexibility, does not cause problems, and may be necessary. For example, some outputs may pertain to a particular study, and others may pertain to a project as a whole. If you assign both the Project and Study levels to Output subtypes, the Definer can select the level that is appropriate for a particular output.

However, if you create a hierarchy that applies only to a limited number of object types, do not assign its levels to other object types. For example, you could create a single-level hierarchy called Technology Type for Programs with SAS, PL/SQL, and Oracle Reports as values, so that you could search for Programs of a particular object type. This hierarchy has no relevance to Tables or Outputs, for example.

When you assign a hierarchy level to a subtype and make it optional, the level appears as an option in a drop-down list when you are classifying a particular object of that subtype and when you are searching for one. It makes sense not to add an item to this list that does not apply to the object type. For example, it is better not to assign the Technology Type hierarchy level to Tables or Outputs.

Allowing Classification by Inheritance

For each classification hierarchy level you assign to an object subtype, you can set the Ref Allowed flag to either Yes or No.

Reference Allowed If you set the flag to Yes, objects of that subtype can inherit their classification values for that hierarchy level from their container: the Domain, Application Area, Work Area, Report Set, or Workflow where they are located.

To make this happen automatically when each object of this subtype is created, you must also set the default classification for the hierarchy level to Inherited (see "Setting a Default Classification" on page 5-8).

You must also assign the same hierarchy level to the parent object subtype.

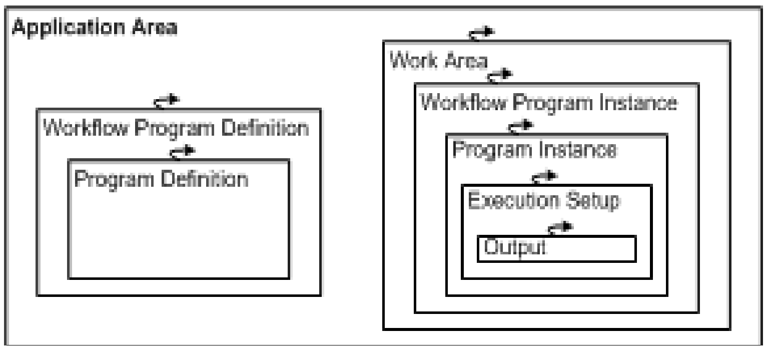
Allowing classification by inheritance can save time during object definition, especially if your organizational structure and classification systems are compatible. For example, if your Application Areas contain applications for a single study, you can assign the hierarchy level Project > Study to Application Areas and to Programs, Tables, Report Sets, and the other primary objects that are contained directly in Application Areas. Then when you classify a particular Application Area to a particular study, such as Project01 > Study01, all the objects contained within it are automatically classified to Project01 > Study01 as well.

In addition, if you change the classification of the Application Area, all the objects it contains for which inheritance classification is allowed are automatically reclassified to the new value for the same hierarchy level. For example, if you copy the Application Area classified to Project01 > Study01 (with all the objects it contains) to serve as the basis for the Application Area for Study02, and change the classification of the copy to

Project01 > Study02, the objects it contains are automatically reclassified to Project01 > Study 02.

You can define several levels of classification by reference. For example,

Figure 5–2 Example of Classification by Inheritance



Classification by Inheritance Not Allowed If you set the Ref Allowed flag to No, Definers can still manually classify objects of that subtype to that hierarchy level. If you define the level as Mandatory for the subtype, Definers must classify them manually to the hierarchy level.

Overriding Inherited Classifications Even if you define an object subtype as having classification by inheritance for a particular hierarchy level, Definers can override that classification for any object of that subtype.

Definers can also change the classification back to a classification by inheritance. The object then is automatically classified to its container's classification value for the hierarchy level.

Note: Top-level Domains cannot have inherited classifications. To enforce this at the subtype level, set Reference Allowed to No for Domains. If you are using nested Domains and you want child Domains to be able to inherit classification values from their parent Domain, you may want to create an additional subtype for Domains. For example, create subtype Top-Level Domain where no referencing is allowed, and subtype Child Domain where referencing is allowed for all classification levels.

Setting a Default Classification

When you assign a classification hierarchy level to an object subtype, you can set one or more default classification values for objects of that subtype. Whatever value you set as the default classification is applied to every object created of this subtype. You can override the default classification for an object at any time.

Default classifications for subtypes are not required. You can set two types of default classifications:

- Default classification by inheritance
- Default classification to a specific value

Setting the Default to Inherited If you set the default classification to Inherited, objects of the subtype automatically receive the same classification for this hierarchy level as their container object. See "Allowing Classification by Inheritance" on page 5-7.

Setting the Default to a Specific Value You can enter one or more specific values for each hierarchy level assigned to a subtype. Every object created as that subtype then is automatically classified the same way.

For example, if the classification level Project > Study is assigned to the subtype, you could set values such as Project01 > Study01 and Project01 > Study02.

This approach is not recommended because the point of classifying objects is to differentiate them in an appropriate way.

Leaving the Default Setting Blank If you do not assign a default value for a hierarchy level, objects of the subtype will be created without a value at that level. You can add a classification to a particular object at any time.

If the hierarchy level is defined as Mandatory for the subtype but has no default value, you must enter a value manually when you create an object of the subtype.

Classifying Outputs

Outputs such as reports receive their classification from the Planned Output in the definition of the Program or other executable that generates them.

If the Planned Output's classification value for a particular level is explicitly assigned, to a particular value, the actual output receives that value for that level. However, since the Planned Output is part of the Program (or other executable) definition, and you may want to reuse the definition by creating instances of it in multiple Work Areas where that value may not be appropriate, this may not be a good way to classify the output.

Alternatively, you can set the Planned Output definition's classification value for a particular level to Inherited. The actual output then receives its value for that level from the Execution Setup used to submit the job that produces the output.

Therefore, when you set up classification levels for object subtypes, you must be sure to include the same levels for corresponding subtypes of both Outputs and Execution Setups.

Execution Setups inherit classification values from their executable instance (for example, the Program instance that owns the Execution Setup). Executable instances inherit their classification values from their Work Area.

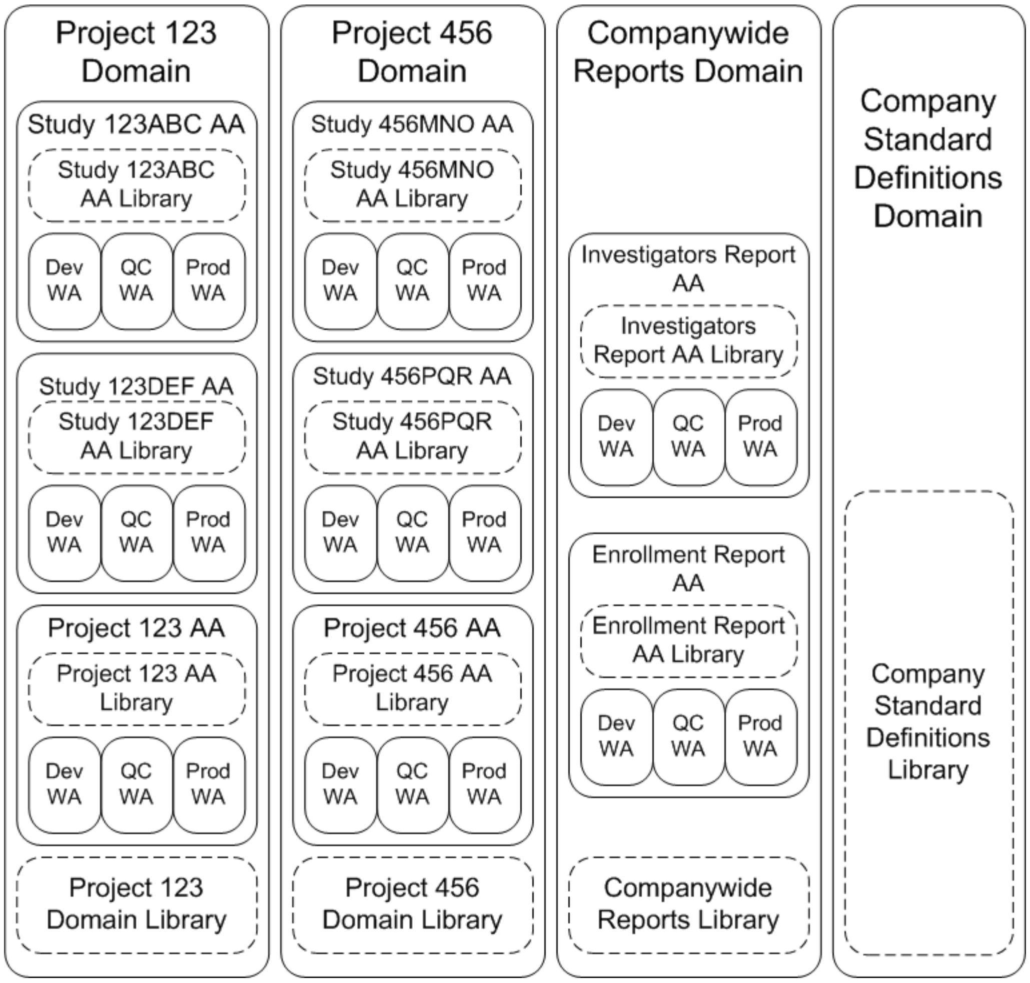
Outputs can also be classified by a parameter value entered when the job is submitted or generated by the job. You can define a Parameter with a list of values that includes the term values in a classification hierarchy level. Further information is available in the "Classifying Outputs" section in the chapter on common development tasks of the *Oracle Life Sciences Data Hub Application Developer's Guide*.

In addition, users with the necessary privileges can reclassify an output after it has been generated.

Classification Example

If you set up your organization and security systems as described in the previous chapters, you could set up classification as described in this section.

Figure 5–3 Example Organizational Structure



In this example, there is a Domain for every project, as well as a Domain for companywide reports and a Domain for company standard definitions. Within each project Domain there is an Application Area for every study in that project and another Application Area for projectwide reports. In every Application Area there are three Work Areas, one each for Development, Quality Control, and Production.

Classification Hierarchies for Use in Browsing and Searching

In this example, there are two classification hierarchies used to create the user interface for displaying outputs in the Reports window: Project > Study > Site > Patient and Division > Department > Group.

In this example, each report output is classified to a single level of the Project > Study > Site > Patient hierarchy and to the Group level in the Division > Department > Group hierarchy. Therefore a link to each report is displayed twice in the Reports window, and users can browse for a report either by its group classification or by its project, study, site or even patient classification.

Project > Study > Site > Patient

This classification hierarchy works well with the organizational structure shown in "Example 1: Project Domain with Reporting on Separately Stored Study Data" on page 3-10.

The Reports window display is based on the logical hierarchical structure of projects and their studies, sites and patients. The Project level contains all the company's project names as values, and the Study and Site levels contain all the company's study and site names, respectively, each linked to their project or study. The Patient level contains the Patient IDs of all patients participating in the project, each linked to their site. Most reports will apply to a whole project, study, or site, but from time to time there may be reports run on a single patient for medical reasons, and these reports are classified to the ID of the particular patient.

Subtype Hierarchy Level Assignments All four levels of this hierarchy are assigned to every subtype of every object type, including organizational objects, object definitions, and object instances. None of the hierarchy levels is defined as Mandatory for any subtype. The default classification value for all is set to Inherited.

For Domains, a subtype called Top-Level Domain has all its classification levels set to Explicit. A subtype called Child Domains has all its classification levels set to Inherited.

This is the most flexible design and the simplest to set up.

Assignments to Specific Objects Individual objects are classified as follows:

- **Project**. The Project 123 and Project 456 **Domains** are each assigned to the Project level with a value of Project 123 and Project 456 respectively. They do not have values assigned for the Study or Site level. The Companywide Reports and Company Standard Definitions Domains do not have any values assigned for any levels in the Project > Study > Site hierarchy, and neither do any of the objects they contain.

  Note: These Domains contain only definitions and definitions are not displayed in the Reports window, and they are not specific to any particular project, study, site, or patient so it is not necessary to classify this Domain to the Project > Study > Site > Patient hierarchy. These classifications would not be useful in either searching or browsing. However, Program, Report Set, and Workflow definitions within the Domain must be classified to a Group in the Division > Department > Group hierarchy; see "Division > Department > Group" on page 5-12.

  The project-level **Application Areas** are assigned to the Project level with a value of Inherited so that they are automatically classified to the same project as their Domain.

  The **Work Areas** within the project-level Application Areas are also assigned to the Project level with a value of Inherited so that they are automatically classified to the same project as their Application Area.

  All the **object definitions** in the project-level Application Area and in the Domain library, and all the **object instances** in their Work Areas, are also assigned to the Project level with a value of Inherited so that they are automatically classified to the same project as their container.

- **Study**. The study-level **Application Areas** are each assigned to the Study level with the value corresponding to their study: Study 123ABC, Study 123DEF, Study 456MNO, and Study 456PQR. The Project value they originally inherited has been explicitly removed.

Each **Work Area** within each study-level Application Area is assigned to the Study level with a value of Inherited so that they are automatically classified to the same study as their Application Area.

Most **object definitions** contained in the study-level Application Areas and **object instances** contained in their Work Areas are also assigned to the Study level with a value of Inherited so that they are automatically classified to the same study as their Application Area.

- **Site**. Some **object definitions** and **instances** within a study-level Application Area or Work Areas generate or contain information that is specific to a site rather than to the whole study. Those particular objects are classified to the Site level, with the value of the particular site or sites whose data they contain, transform, or report. The study-level classification value they inherited has been explicitly removed.
- **Patient**. Some **object definitions** and **instances** within the study-level Application Areas or Work Areas generate or contain information that is specific to a particular patient; for example, a serious adverse event report. Those objects are classified to the Patient level. When the Definer creates Program to generate a report on information about a single patient, the Definer does not know in advance which patient will require the report. Therefore the Definer must make the Patient ID an input/output Parameter of the report. The Definer can also use this Parameter to classify the report to the appropriate Patient ID.

Division > Department > Group

The company organization hierarchy allows people to browse for reports that pertain to their own group in the corporate structure. It has three levels: Division > Department > Group.

The Division level is the highest and includes as values the names of all corporate divisions that use Oracle LSH; for example, Development or Manufacturing. The Department level is the middle level and includes all departments that use Oracle LSH, related to their division. For example, the Development division value is linked to the Department values Biometrics, Clinical Science, and Project Management. The Group level is lowest and includes groups within the departments that use Oracle LSH; for example, the Biometrics department value is linked to Group values Data Management and Statistics, and the Clinical Science department is linked to Group values Clinical Pharmacology and Cardiology.

In this example, reports are always classified to the Group level. The Division and Department levels serve only to organize the groups into the correct structure to make the reports easier to find in the Reports window.

Subtype Hierarchy Level Assignments In this example, Oracle LSH is used only for clinical data, not financial or administrative or data originating in any division other than the Development Division. Therefore the Companywide Reports Domain and the Company Standard Definitions Domain are both classified to the Development division. The enrollment report is classified to the Data Management group and the investigators report is classified to the Project Management department.

Definitions and instances of Programs, Report Sets, and Workflows in all Domains are all assigned to the Division > Department > Group level. In all cases, the level is defined as Mandatory and the default value is blank, so that the Definer must enter a specific value for each object of those subtypes.

Planned Outputs and Execution Setups are also assigned to the Division > Department > Group as a mandatory level. The default value is Inherited. Planned Outputs automatically receive the classification value of their Program definition, and Execution Setups receive the classification value of their Program instance.

Assignments to Specific Objects Definers must enter a specific classification value in the Group level for each definition and instance of Programs, Report Sets, and Workflows when they create those objects. Unlike the Project > Study > Site classification hierarchy, where a particular Program might be used for different projects, studies, or sites, a Statistics Program will probably always be a Statistics Program, and a Cardiology Program will always be a Cardiology Program.

By default, outputs receive their Group classification from the Planned Output, which receives the classification value of the Program definition. Because a Statistics Program will probably always be a Statistics Program, and so on, this default behavior is almost always correct. However, it is possible for the Definer to override the default classification; See "Classifying Outputs" on page 5-9.

Classification Hierarchies for Use in Searching Only

In this example, there are additional keyword hierarchies defined for use in searching for object definitions, instances, and outputs. Most are used only with objects of a particular type or subtype, as described in this section. Most are defined as Mandatory and have a blank default value, so that the Definer must select a classification value for each individual object. Keyword hierarchies do the following:

Label Programs by Technology Type A keyword hierarchy called Program Technology Types has classification values SAS, PLSQL, and Oracle Reports. Definers use the classifications to search for Program definitions to reuse.

Label Load Sets by Type A keyword hierarchy called Load Set Type has classification values for all the different Load Set types, including SAS, Text, Oracle Tables and Views, and all the Oracle Clinical Load Set types (see "Defining Load Sets" in the *Oracle Life Sciences Data Hub Application Developer's Guide*). Definers use the classifications to search for Load Set definitions to reuse.

Label Data Marts by Format A keyword hierarchy called Data Mart Formats has classification values ASCII Fixed, ASCII Delimited, SAS CPort, SAS XPort, SAS Dataset, and Oracle Export. Definers use the classifications to search for Data Mart definitions to reuse.

Label Programs by Purpose A keyword hierarchy called Program Purpose has classification values that categorize the Program by its general purpose, such as: Combine Data and Report Data. You can include more specific values (such as Combine Demographic Data) or include different types of categories in the same list of values (such as Demography and Concommittant Meds) and assign multiple values to each Program, or use a separate hierarchy for labeling data (see below). Definers use the classifications to search for Program definitions to reuse.

Label All Primary Objects with a Data Category A keyword hierarchy called Data Category has classification values that correspond to the data content of commonly used Tables; for example, Demography, Concommittant Medications, and Adverse Events. This hierarchy level is assigned to all definitions and instances of Programs, Workflows, Report Sets, Load Sets, Data Marts, and Tables. Tables are classified by the category of data they contain. Executable objects are classified by the type of data they operate on. Some objects are classified to multiple values.

Label Outputs and Data Marts for Submission to Regulatory Agencies A keyword hierarchy called Submission has values Not for Submission and For Submission. Report Sets, Data Marts, and their outputs that are intended for submission are labeled as such. This value, with the appropriate study or project value and other criteria, helps in assembling all the necessary data.

Label Data Marts to be Sent to Research or Business Partners A keyword hierarchy called Partners contains the names of organizations with which the Oracle LSH customer needs to share data; for example, a pharma company creates a list of CROs, or a CRO creates a list of pharma companies. Data Marts and their outputs are labeled with the name of the organization they must be sent to.

Design Decision Summary

To create a classification system for use in searching and browsing, you must do the following:

- Design a Set of Classification Hierarchies with Levels and Values on page 5-14
- Assign Hierarchy Levels to Object Subtypes on page 5-14
- Assign Hierarchy Levels to Domains, Application Areas, and Work Areas on page 5-16

Design a Set of Classification Hierarchies with Levels and Values

Design a set of classification hierarchies, including at least one with multiple levels, and define values in each level. If a hierarchy has multiple levels, they must be arranged in a strict linear hierarchy, and each value in the lower levels must be related to a value in the next higher level. See "Defining Classification Hierarchies" on page 5-3 and "Classification Example" on page 5-9 for further information.

Restrictions You must define at least one nonkeyword classification hierarchy for use in browsing, or nothing will be displayed in the Reports window of the Oracle LSH user interface.

Considerations We recommend the following approach:

- Design one or more classification hierarchies to serve as the basis for the Reports window user interface. Design these hierarchies first, and in conjuction with the organizational structure (see Chapter 3, "Designing an Organizational Structure").
- Design as many additional hierarchies as you need to help users find object definitions and reports. For examples, see "Classification Hierarchies for Use in Searching Only" on page 5-13.
- If you plan to use Oracle LSH to process financial or administrative data as well as clinical data, define additional subtypes and hierarchies. You can add subtypes and hierarchies at any time.

Assign Hierarchy Levels to Object Subtypes

If you want to be able to use values in a hierarchy level to label an object, you must assign the hierarchy level to the subtype of that object. For each hierarchy level assignment to an object subtype, you also define:

- Whether a classification to that hierarchy level is mandatory for objects of that subtype

- Whether Inherited classification is allowed for objects of that subtype
- A default classification for objects of that subtype (optional, and may be Inherited or, in the case of outputs, a classification parameter value)

Restrictions If an output is not classified it does not appear in the Reports tab.

Considerations Take the following into consideration:

- **For a child object to inherit a classification value, its parent must have the same classification level assigned**. If you define a classification level assigned to an object subtype as Inherited, be sure to assign the same classification level to the parent's object subtype. Otherwise Definers will receive an error if they try to use the inherited classification.

  For example, a Definer creates a Program definition using the Default subtype, which has the classification level Project > Study assigned and set to Inherited. If the parent Application Area does not also have the Project > Study level assigned, the Program definition cannot inherit a value from the parent Application Area. Therefore the system gives an error. The Definer must explicitly set the value for Study for the Program.

- **The system tries to create new objects with the same subtype as their parent object**. When a Definer creates an object the system checks the subtype of the parent object. If the new object has the a subtype with the exact same name, the system creates the new object using that subtype by default. If not, the system uses the predefined Default subtype.

  For example, when a Definer creates a Program instance and definition at the same time, the system tries to give the instance the same subtype as the Work Area and the definition the same subtype as the Application Area. If Program instances do not have a subtype with the same name as the parent Work Area, the system uses the predefined Default subtype. If Program definitions do not have a subtype with the same name as the Application Area, the system uses the predefined Default subtype.

  Therefore, if you define a Financial and a Clinical subtype for all object types, and define one top-level Domain as Financial and another as Clinical, by default the system creates all objects in the Financial Domain as Financial and in the Clinical Domain as Clinical.

  The Definer can change the subtype as necessary.

- **Coordinate your classification level assignments for Execution Setup subtypes and for Output subtypes**. Subtypes defined for Outputs are used by Planned Outputs.

  If you want an Output to inherit its classification values from its Execution Setup, which belongs to the executable instance in a Work Area, rather than its Planned Output, which belongs to the executable definition in an Application Area or Domain, set the Planned Output's classifications to Inherited. However, if the Execution Setup does not have the same classification levels assigned as the Planned Output, the actual output will inherit values for the missing levels from the Planned Output's parent; namely, the executable object definition.

- **For maximum flexibility, make all classification levels optional and inherited** for all objects except organizational objects—Domains, Application Areas, and Work Areas—so that all primary definitional objects inherit their classifications from the Domain, Application Area, or Work Area where they are located. Do not define default classifications.

- **Mandatory Classifications**. If you define a level as mandatory for Output subtypes and the Definer does not provide a default value, Oracle LSH will not execute the job to generate the output unless the person who submits the job provides a value.
- **Multiple Displays**. If you assign multiple hierarchy levels to an Output subtype and an actual output is classified to them, it appears in multiple locations in the Reports screen.

Assign Hierarchy Levels to Domains, Application Areas, and Work Areas

As you set up your Domains, Application Areas, and Work Areas, assign classification values to them according to your design. If you are using Inherited for any object subtypes, objects of those subtypes will automatically be assigned the same values as they are created. You may need to remove some of these inherited classifications. See "Allowing Classification by Inheritance" on page 5-7 and "Classification Example" on page 5-9.

6 Validating Objects and Outputs

This section contains information on the following topics:

About Validation

Regulatory agencies require that you validate the data you submit as part of your drug approval application. The Oracle Life Sciences Data Hub (Oracle LSH) includes tools to help you to do this, including validation statuses, validation supporting information, and coversheets that include validation information for every report you create.

Oracle LSH cannot certify the quality of the data that you load into Oracle LSH. It is your responsibility to validate data in your source data system(s). However, Oracle LSH can help you certify that Oracle LSH handled data in a valid manner. If you can demonstrate that your source data is valid, and that Oracle LSH manipulated your source data in a valid manner, then data reported or sent out of Oracle LSH must be valid. Each time you generate a report on Oracle LSH data, you can generate a report coversheet that includes the validation status of every defined object that affected the source data for the report, including Tables, Programs, and Load Sets, since the data entered Oracle LSH.

In addition, Oracle LSH allows you to manually validate Program outputs using the same methods you do now. In the context of a Report Set, where different sections may be at different stages of validation at any one time, the system reuses existing, validated outputs as long as any requested outputs would be duplicates of the validated one. Therefore you can continue to develop some sections of a Report Set, changing Parameter values and using new Program versions, and reuse the outputs of other sections as long as your changes do not affect those sections.

Oracle LSH keeps all defined objects under version control. Each version of each object has a validation status. Oracle LSH ships with predefined validation statuses that represent different stages in an object's life cycle: Development, Quality Control, Production, and Retired.

By default, all new objects and all new versions of objects have a validation status of Development. Oracle LSH does not enforce any conditions for promotion of an object to Quality Control or Production. You should develop the standards you require for promotion. Oracle LSH does enforce rules based on objects' validation status and on the interaction of a Work Area's validation status and its Usage Intent attribute value (see "Work Area Usage Intent and Validation Status" on page 6-5). These rules promote using validation status to create separate, clean, Quality Control and Production environments.

Oracle LSH allows you to link documents and outputs with objects to demonstrate that they meet a validation requirement. For example, you can store documents such as Functional Requirements, Test Requirements, or Test Cases with a Program or Report Set, or an output such as a log file or generated report to demonstrate that the Program or Report Set was successfully executed. See "Validation Supporting Information" on page 6-4 for further information.

You can begin application development in Oracle LSH before arriving at your standards for definitional object validation, and it is possible to use Oracle LSH without ever using another validation status beyond Development. However, if you plan to submit data to a regulatory agency, you should not go into production until you have developed standards and validated all objects that operate on or store production clinical data.

All objects that are contained directly in a Domain, Application Area, or Work Area have a validation status.

Validation Statuses

Oracle LSH allows you to apply the following validation statuses fo all objects that require validation: Development, Quality Control, Production, and Retired.

Development

Oracle LSH automatically gives all new objects, and new object versions, a validation status of Development.

Organizational objects (Domains and Application Areas) have a permanent validation status of Development, which is never visible in the user interface.

Quality Control

This validation status is intended to be used for objects that have already undergone informal testing by a programmer and are currently being formally tested, perhaps by a separate group of people, such as Quality Control engineers.

It is your responsibility to develop standards for the use of the Quality Control (QC) validation status.

Production

This status is intended for objects that have been thoroughly tested and certified as being ready for use with production clinical data suitable for preparatory to submission to a regulatory agency.

Oracle LSH protects data in a Work Area with a validation status of Production by preventing the use of Work Area installation modes that would delete data in the Work Area's tables.

You must develop standards for the use of the Production validation status.

Retired

This validation status is intended for objects you no longer wish to use.

Retired Object Rules Oracle LSH enforces the following rules on objects with a validation status of Retired:

- You cannot create an instance of a Retired object definition. (However, existing instances of a definition whose status is changed to Retired can still have an active validation status and function as before.)
- You cannot modify an existing object instance to point to an object definition whose validation status is Retired.
- Executable object instances whose validation status is set to Retired cannot be run.

If you decide that you want to use a retired object, there are three ways to do so:

- If you have the necessary privileges, you can reactivate the object, setting its validation status back to whatever it was immediately before retirement.
- Copy the retired object. The system sets the validation status of the new object to Development.
- Create a new version of an object by checking it out (object definitions only).

Retired Work Area Rules Retire the Production Work Area for a trial when you close the trial. Oracle LSH enforces the following behavior for Work Areas whose validation status is set to Retired:

- No object contained in a Retired Work Area can be run.
- Objects contained in a Retired Work Area retain their current validation status when the Work Area is set to Retired.
- When you clone a Retired Work Area, the validation status of the clone is also set to Retired. The object instances contained in the original Retired Work Area retain the same validation status in the clone.
- When you clone a Retired Work Area, any Retired object instances contained in the original Work Area are cloned along with all the other object instances.

Validation-Related Security

Oracle LSH includes the following validation-related operations on object types:

- **Modify Validation Status QC**. This privilege is required to set an object's validation status to Quality Control. Users with this privilege can also set the status to Retired.
- **Modify Validation Production**. This privilege is required to set an object's validation status to Production. Users with this privilege can also set the status to Retired.
- **Manage Supporting Information**. This privilege is required to link documents and outputs to an object as Validation Supporting Information.

In addition, Work Areas have a validation-related operation called **Install Qualify Submit** . No executables can be run in a Work Area until the Work Area's validation status is equal to or greater than its usage intent value, except by users with the special Install Qualify Submit privilege on the Work Area. This is to allow testing of the Work Area before making it generally available for use. See "Work Area Usage Intent and Validation Status" on page 6-5 for further information.

Validation Supporting Information

Oracle LSH allows you to link an object to two types of supporting information for the purpose of documenting that one of your internal validation requirements has been met: Supporting Documents and Supporting Outputs.

Oracle LSH does not enforce the use of supporting information. It is possible to promote an object to a higher validation status even if the object is not linked to the supporting information you require in your standard operating procedure (SOP).

Supporting Documents

You can upload documents to Oracle LSH and create a link from an object to the document. Oracle LSH keeps each document under version control, creating a new version number each time the document is uploaded. Oracle LSH never deletes a supporting document.

Users with the Manage Supporting Information privilege can link objects to supporting documents and outputs.

For example, you could store the following types of documents as validation supporting information for an object:

- a requirements document stating the purpose of the object and specifying functional requirements
- a design document that specifies the way the functional requirements will be achieved technically
- a test requirements document listing the functionality that must be formally tested
- actual test cases to be used in testing the object
- test results

The examples given above might all be appropriate requirements for promotion from Development to Quality Control, and require the last for promotion from QC to Production.

Supporting Outputs

You can also create a link between an object and a particular output. For example, to demonstrate that you successfully ran a Program, you could create a link to one or both of the following outputs of the successful job:

- a report generated by the Program
- the log file of the successful Program execution

Supporting outputs may be appropriate requirements for promotion to Quality Control and Production status.

Output Coversheets

You can generate a summary or detailed coversheet for any output in Oracle LSH that lists the validation status of the defined objects that handle the data in the output.

- The summary coversheet gives the validation status of the executable object that generated the output and of each Table instance from which the executable read data.
- The detailed coversheet gives the validation status of each executable object and Table instance that handled the data from the time it was loaded into Oracle LSH to the executable that produced the output. The system examines the immediate source Table instances, then the executables that wrote data to those Table instances, then the Table instances that those executables read data from, and so on to the Load Sets that loaded data into Oracle LSH.

If you have developed good standard operating procedures using the tools provided by Oracle LSH for testing and validating defined objects to ensure that they do not corrupt data, and followed those procedures, then data that was valid when loaded into Oracle LSH should remain valid in Oracle LSH.

Cascading Validation

Whenever you change an object's validation status to a higher status, the system attempts to make the same validation upgrade to any objects contained by the object you are explicitly upgrading. In addition, if you upgrade an object instance to a status higher than the status of its underlying definition, the system attempts to upgrade the validation status of the definition as well. For example:

- If you promote a Workflow definition from Development to QC, the system attempts to upgrade all the object instances owned by the Workflow, such as Programs and Load Sets. The system also attempts to upgrade the underlying definition of each instance.
- If you promote a Work Area from QC to Production, the system attempts to upgrade all the object instances contained in the Work Area, and all the definitions on which they are based.

If the system is unable to upgrade the status of all affected objects, the validation fails. The system is unable to upgrade objects in the following circumstances:

- The user performing the upgrade does not have the Modify Validation Status privilege on one of the other objects that must be upgraded as part of the operation.
- One of the objects that must be upgraded is checked out by a different user.

An upgrade is changing validation status from Development to either Quality Control or Production, or from Quality Control to Production.

Downgrades include changing validation status from Production to any other status, changing from Quality Control to Development, and changing from any other status to Retired. There is no cascading of validation downgrades.

Work Area Usage Intent and Validation Status

Work Areas have a special attribute called Usage Intent whose allowed values are Development, Quality Control, and Production. Oracle LSH enforces the following rules based on a Work Area's usage intent value and the validation status of the Work Area and all its object instances:

- To be installed in a Work Area, object instances must have a validation status equal to or greater than the usage intent of the Work Area.
- The Work Area cannot be promoted to a validation status higher than the validation status of any of its object instances.
- No executables can be run in a Work Area until the Work Area's validation status is equal to or greater than its usage intent value, except by users with the special Install Qualify Submit privilege on the Work Area. This is to allow testing of the Work Area before making it generally available for use.
- Full installation and Table instance replacement during partial installation are not allowed in Work Areas with a usage intent of Production. This is to protect production data from deletion.

These rules support the following intended Work Area usage:

1. **Development**. When you create a Work Area, the system sets both its Usage Intent attribute and its validation status to Development. When you create an object definition and/or instance, the system sets its validation status to Development. When a Definer has successfully conducted informal testing on an object, according to your organization's standards, he or she sets the object's validation status to Quality Control (or requests a privileged user to change the status, depending on your security design).

   When all the object instances in a Work Area have a validation status of Quality Control, a privileged user clones the Work Area, creating duplicates of all object instances with pointers to the same object definitions and the same version number and validation statuses as the originals. The system creates a label for both Work Areas indicating that they are identical. The privileged user enters text for the label and creates the new Work Area with a usage intent of Quality Control. However, its validation status remains at Development.

2. **Quality Control**. Install the new Quality Control Work Area. While its validation status (Development) is lower than its usage intent (Quality Control), only a user with the Install Qualify Submit privilege can run executables. That privileged user loads fresh data and tests the installation. The privileged user then promotes the Work Area validation status to Quality Control. Users with normal access to the Quality Control Work Area can then test the objects.

   If testers find bugs or other problems, developers should fix them in the Development Work Area. To do this, you can clone the QC Work Area as a new Development Work Area overwriting the old one, perform a full installation, and load fresh data. When all objects have been fixed and tested, and their validation status upgraded, you can clone the Work Area onto the QC Work Area. The cloning operation updates only objects that have been modified.

   Promote each object definition and instance to Production when it meets your production standards. Clone the QC Work Area to create a Production Work Area.

3. **Production**. Install the new Production Work Area. While its validation status is lower than its usage intent, only a user with the Install Qualify Submit privilege can run executables. That privileged user loads fresh data and tests the installation. The privileged user then promotes the Work Area validation status to Production. Users with normal access to the Production Work Area can then run the application.

   Run the application as necessary for your business purposes. Using the same cloning procedures described above, modify the objects as necessary over time. Make modifications through the Development Work Area, test each modified object definition in the QC Work Area, and clone the QC Work Area to the Production Work Area.

Note: The cloning operation replaces only objects that have been modified. If the cloning operation would be destructive to Tables or data (for example, the cloned Work Area did not include an existing Table or Table column, or a Column length was shorter) you receive a warning, but are allowed to continue.

---

4. **Retired**. When you close a trial, you can retire the Production Work Area that holds its data, preserving the data in a frozen state, recreatable as it was at any point in time. You can still write Programs that read the data. For example, you can combine and analyze the data with data from other closed or current trials.

Validation Rules

The system enforces the following rules based on objects' validation status:

- You can change an object's validation status only when it is checked in. If the object is not checked in, the system tries to checks it in. If another user has checked it out, you cannot change the validation status.

  Changing the validation status of a checked-in object does not change the version number of the object.

- You cannot promote an object instance to a validation status higher than its underlying object definition.

  For example, if you create an instance of a Program definition whose validation status is Quality Control, you can promote the instance to a validation status of Quality Control, but you cannot promote it to Production until the definition has been promoted to Production. However, the system attempts to promote the definition to Production when you promote the instance, and if you have the necessary privileges on the definition, and if the definition is not checked out by a different user, the promotion succeeds.

- Validation status is version-specific. Each time you create a new version of an object by checking it out (including checking out a definition through an instance) the system automatically gives the new version a validation status of Development. The system does not change the validation status of any existing versions.

- When you copy an object, the validation status of the copy is set to Development no matter what the validation status of the original is.

- You cannot run an Execution Setup whose validation status is less than the validation status of its owning object instance unless you have the IQ Submit security privilege on the Execution Setup.

- To be installed in a Work Area, object instances must have a validation status equal to or greater than the usage intent of the Work Area.

- The Work Area cannot be promoted to a validation status higher than the validation status of any of its object instances. However, if you try to promote a Work Area to a status above that of any of its objects, you have the opportunity to promote all the object instances and their underlying definition versions to the same status in a cascade operation.

The cascade validation fails if a different user has checked out any of the object definitions in need of promotion.

- No executables can be run in a Work Area until the Work Area's validation status is equal to or greater than its usage intent value, except by users with the special Install Qualify Submit privilege on the Work Area. This is to allow testing of the Work Area before making generally available for use.
- Full installation and the Replace operation on Table instances in partial installation are not allowed in Work Areas with a usage intent of Production. This is to protect production data from deletion.
- You cannot remove a Table instance whose validation status is Production from a Work Area of any usage intent.

Design Decision Summary

You should set standards for promotion to the Quality Control and Production validation statuses for definitions and instances of Tables, Programs, Load Sets, Report Sets, Workflows, Data Marts, and Business Areas. Oracle LSH automatically promotes component objects such as Parameters, Notifications, and Overlay Templates when you promote their containing object. However, you may also want to explicitly validate these object definitions and move them to a Domain for reuse.

If you are using object subtypes, you can set different validation requirements for different subtypes of the same object type.

Oracle LSH provides two types of requirements to use: Supporting Information and Security. You can create additional required operating procedures if you so choose.

Supporting Information

Specify documents and/or outputs that you require for promotion to each validation status. Write and enforce your policy as you would any other company standard operating procedure (SOP). Oracle LSH stores supporting documents and links both documents and outputs to objects as Validation Supporting Information.

Restrictions None.

Considerations Oracle LSH does not enforce any system behavior in relation to Validation Supporting Information. However, the Data Validation Report lists all documents and outputs linked to an object as Validation Supporting Information.

Security

Oracle LSH supplies the operations Modify Validation Status QC, Modify Validation Status Production, and Manage Supporting Information operation on all object types and subtypes. You must define one or more roles with each of these operations on object subtypes, assign the roles to user groups, and assign users to the roles within user groups.

Restrictions Users with a role that has the Modify Validation Status QC operation can promote an object from Development to Quality Control. They can also downgrade any validation status except Production, provided that Validation Rules are not violated.

Users with a role that has the Modify Validation Status Production operation can promote an object from QC to Production as well as from Development to QC. They can also downgrade any validation status, provided that Validation Rules are not violated.

Considerations You can assign both operations to the same role or to different roles. If you assign them to different roles, you can still assign both roles to the same user. However, if you always assign different users to each role, you can demonstrate that more than one person was involved in validating an object.

You can use the same roles for all object types and subtypes or differentiate them. For example, you might want to use a different role for promoting Load Sets than for promoting Report Sets.

A
Object Types with Operations

The Oracle Life Sciences Data Hub (Oracle LSH) ships with many predefined object types, most of which have predefined operations. Object types that are not organizational object types themselves andare not owned by organizational object do not have operations.

This sections lists all object types with operations defined for them, and lists the predefined operations for each object types. The object types are divided into the following categories:

- Organizational Object Types on page A-1
- Primary Object Types on page A-4
- Secondary Object Types on page A-5
- Instance Object Types on page A-8
- Outputs on page A-9

Organizational Object Types

Organizational objects contain and logically organize other objects. Here they are divided into two groups:

- Organizational Objects Used in Definition on page A-1
- Organizational Objects Used in Administration on page A-3

Organizational Objects Used in Definition

These organizational objects are visible in the Oracle LSH user interface (UI), in the Applications tab. In the UI and in Oracle LSH documentation, a Library Domain is called a Domain. Internally in Oracle LSH it is called a Library Domain to distinguish it from an Adapter Domain.

Library Domain A Library Domain's operations are:

View
Modify
Delete
Manage Security
Create Application Area
Create Business Area
Create Data Mart
Create Load Set
Create Notification

Create Parameter
Create Parameter Set
Create Program
Create Report Set
Create Overlay Template
Create Table
Create Variable
Create Workflow
Create Workflow Structure
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Application Area An Application Area's operations are:

View
Modify
Delete
Manage Security
Create Work Area
Create Business Area
Create Data Mart
Create Load Set
Create Notifications
Create Parameter
Create Parameter Set
Create Program
Create Report Set
Create Overlay Template
Create Table
Create Variable
Create Workflow
Create Workflow Structures
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Work Area A Work Area's operations are:

View
Modify
Delete
Manage Security
Create Business Area Instance
Create Data Mart Instance
Create Load Set Instance
Create Program Instance
Create Report Set Instance
Create Table Instance
Create Workflow Instance
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Install
Install Qualify Submit
Clone

Organizational Objects Used in Administration

These organizational objects are not visible through the Oracle LSH user interface. The Instance Domain is the outermost container in Oracle LSH; it contains all Domains. Adapter Domains and Adapter Areas are used to integrate external systems with Oracle LSH.

Instance Domain An Instance Domain's operations are:

View
Modify
Manage Security
Create Library Domain
Create Adapter Domain

Adapter Domain An Adapter Domain's operations are:

View
Modify
Delete
Manage Security
Create Adapter Area
Create Business Area
Create Data Mart
Create Load Set
Create Overlay Template
Create Parameter
Create Parameter Set
Create Program
Create Report Set
Create Table
Create Variable
Create Workflow
Create Workflow Structure
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Adapter Area An Adapter Area's operations are:

View
Modify
Delete
Manage Security
Create Work Area
Create Business Area
Create Data Mart
Create Load Set
Create Notification
Create Overlay Template
Create Parameter
Create Parameter Set

Create Program
Create Remote Location
Create Report Set
Create OTD
Create Table
Create Variable
Create Workflow
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info
Create Remote Area
Create Remote Location

Primary Object Types

Oracle LSH uses seven predefined primary object types. Object definitions of these types can be contained in either a Domain or an Application Area. There is a chapter on each of these object types in the *Oracle Life Sciences Data Hub Application Developer's Guide*.

Business Area A Business Area's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Data Mart A Data Mart's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Load Set A Load Set's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Program A Programs's operations are:

Delete

Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Report Set A Report Set's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Table A Table's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info
Read
Read Unblind
Blind Break

Workflow A Workflow's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Secondary Object Types

Secondary objects cannot be installed and used independently; they serve as components of primary objects. However, some of them are contained directly in organizational objects so that they are available for reuse. Here they are divided according to their container:

- Secondary Objects Contained in Organizational Objects on page A-6
- Secondary Objects Contained in Instance Objects on page A-7
- Secondary Object Types Contained in Administrative Organizational Objects on page A-7

Secondary Objects Contained in Organizational Objects

Only secondary objects that are contained directly in organizational objects or instance objects require security operations. Security for operations on secondary objects contained in primary object definitions, such as Source Code, Source Code instances, and Table Descriptors, is controlled by the Modify operation on the containing primary object (such as a Program definition).

Notification A Notification's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Overlay Template An Overlay Templates's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Parameter A Parameter's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Parameter Set A Parameter Set's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Variable A Variable's operations are:

Delete
Modify
Manage Security
View

Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Workflow Structure A Workflow Structure's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info

Secondary Objects Contained in Instance Objects

Only one secondary object belongs to object instances. All Oracle LSH executable primary object instance types contain Execution Setups.

Execution Setup An Execution Setup's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info
Submit
IQ Submit

Secondary Object Types Contained in Administrative Organizational Objects

Remote Locations are contained in Adapter Areas and are accessible only through the Admin tab in the Oracle LSH user inteface. Connections are contained in Remote Locations.

Remote Location A Remote Location's operations are:

Delete
Modify
View
Manage Security
Classify
Reports
Create Connection

Connection A Connection's operations are:

Delete
Modify
View
Use

Connections of type Shared have the following additional operations:

Manage Security
Classify

Instance Object Types

Instance objects are contained in a Work Area.

Business Area Instance A Business Area Instance's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info
Read Business Area Instance
Read (Unblind) Business Area Instance.

Data Mart Instance A Data Mart Instance's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info
Create Execution Setup

Load Set Instance A Load Set Instance's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info
Create Execution Setup

Program Instance A Program Instance's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info
Create Execution Setup

Report Set Instance A Report Set Instance's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info
Create Execution Setup

Table Instance A Table Instance's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info
Read Data
Blind Break
Unblind
Read Unblind

Workflow Instance A Workflow Instance's operations are:

Delete
Modify
Manage Security
View
Classify
Modify Validation Status QC
Modify Validation Status Production
Modify Supporting Info
Create Execution Setup

Outputs

Outputs are in a category by themselves. These are the actual outputs generated by executable primary object instances. An Output Instance's operations are:

Annotate
Blind Break
Classify
Delete
Manage Security
Modify
Modify Supporting Info
Modify Validation Status Production
Modify Validation Status QC
Read Unblind
Unblind
View

B
Object Ownership

In the Oracle Life Sciences Data Hub (Oracle LSH), object definitions and instances are "owned" by their immediate container object. The following diagrams show the Oracle LSH data model for object ownership:

- Figure B–1, "Object Ownership within a Domain"
- Figure B–2, "Object Ownership within an Application Area"
- Figure B–3, "Object Ownership within a Work Area"

Object ownership has ramifications in many areas of Oracle LSH, including security (user group assignments) and classification; it is possible for objects to inherit the user group assignments and classifications of their owning objects. In addition, Oracle LSH enforces unique names for objects of the same type in the same container object.

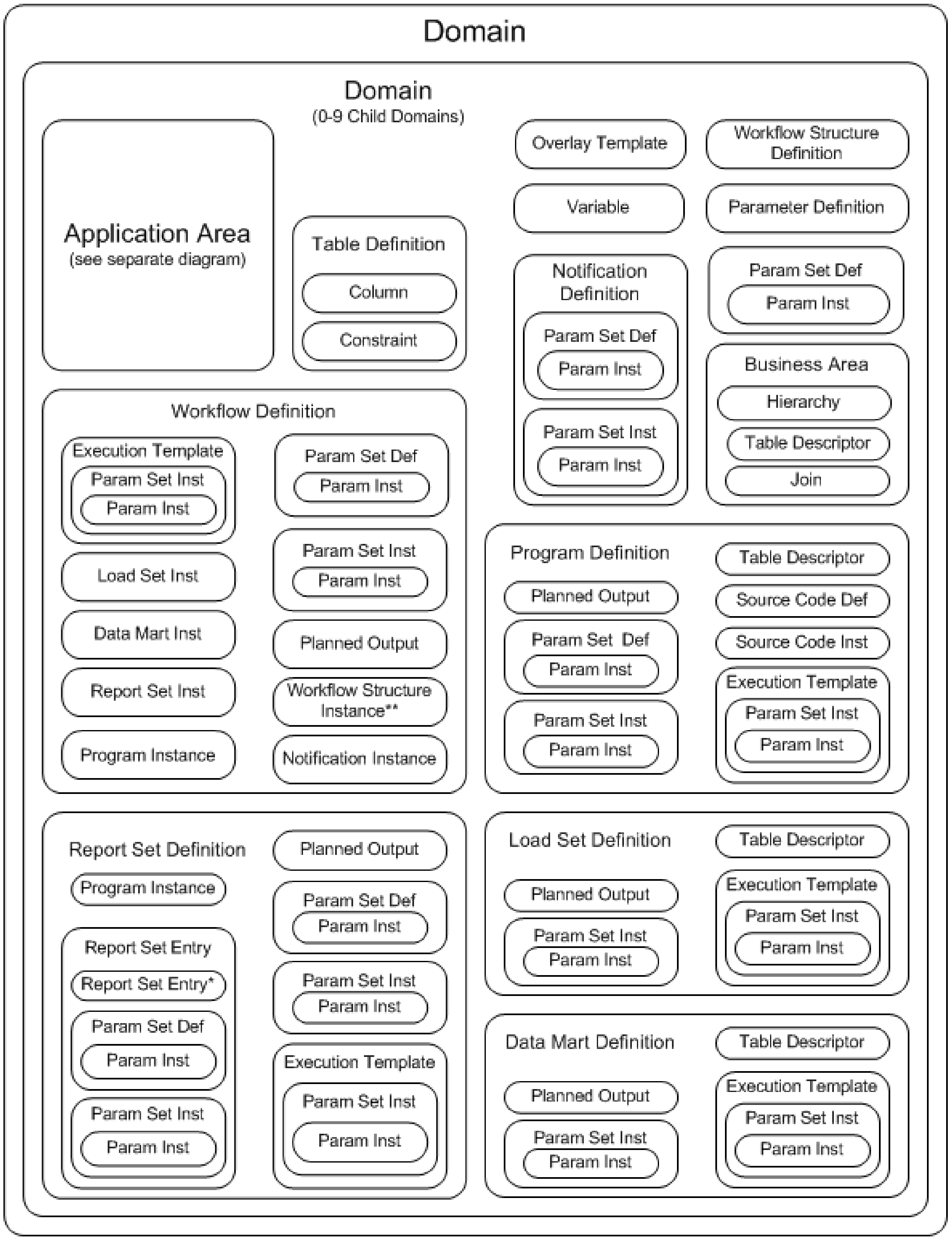
Domains

Domains contain Application Areas and object definitions. Object definitions contained directly in a Domain constitute the Domain library.

Figure B–1 shows all layers of object ownership within a Domain, except for objects contained in the Application Area. See Figure B–2 for object ownership within an Application Area.

Note: Report Set Entries inside Report Set Definitions can be nested indefinitely.

Figure B–1 Object Ownership within a Domain

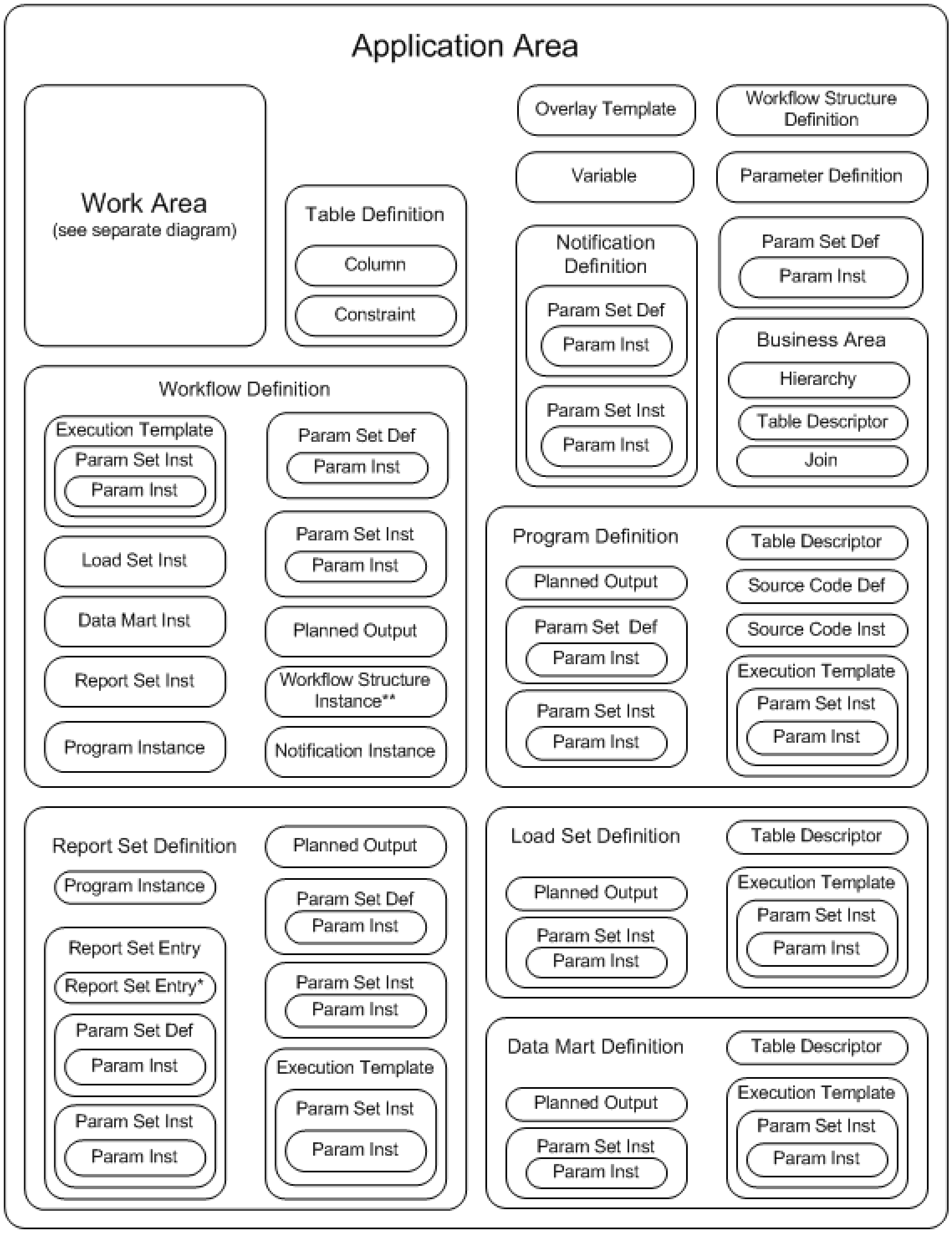


Application Areas

Application Areas contain exactly the same objects as Domains, except that Domains contain Application Areas and Application Areas contain Work Areas. Both Domains and Application Areas can include all object definition types. Figure B–2 shows all possible layers of object ownership within an Application Area.

Note: Report Set Entries inside Report Set Definitions can be nested indefinitely.

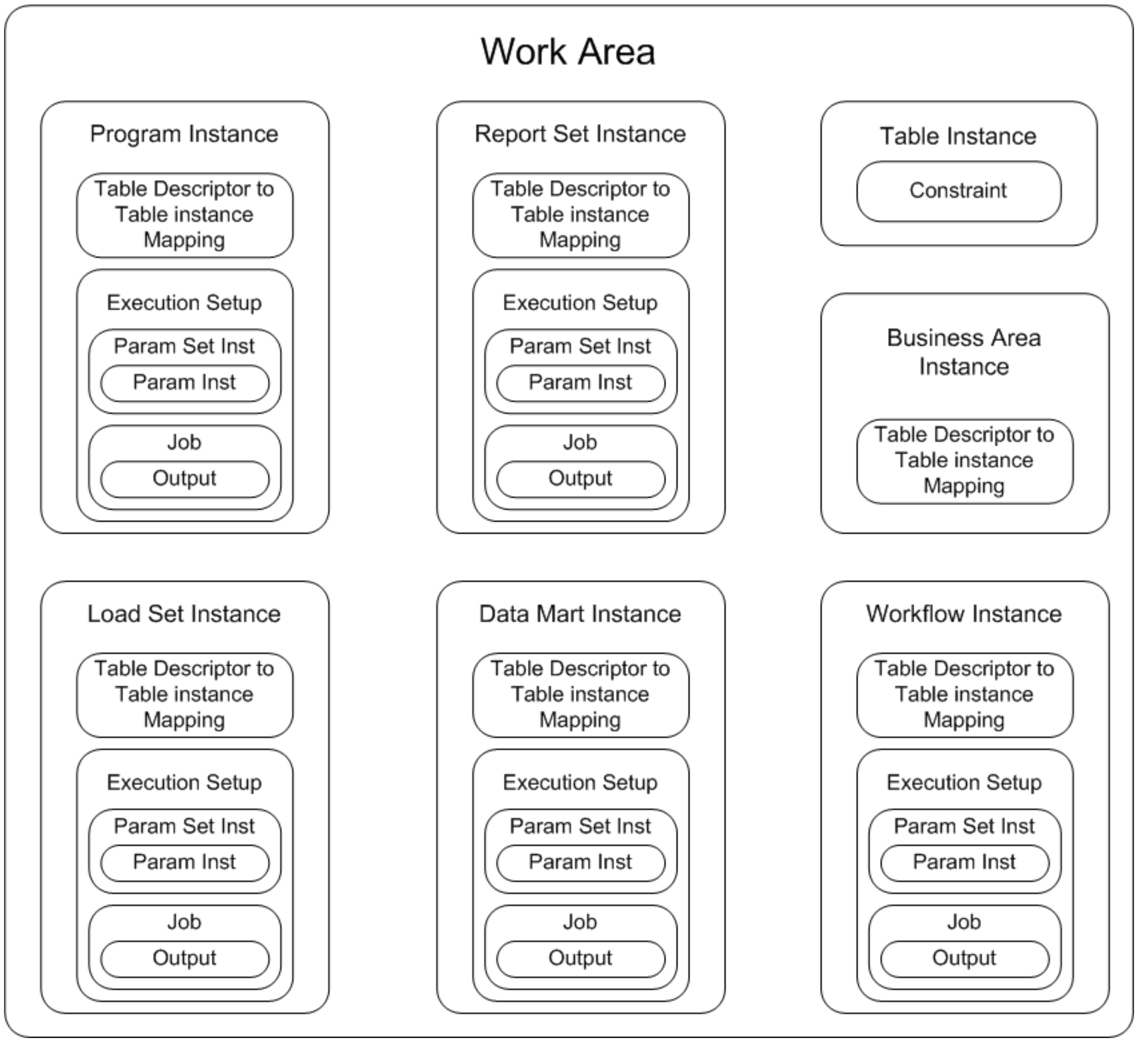
Figure B–2 Object Ownership within an Application Area



Work Areas

Work Areas contain object instances, each of which points to an object definition located either in an Application Area or in a Library. This diagram shows only object ownership, so you cannot see the pointers to the definitions. Most object instances point to an object definition with the same type name; for example, Program instances point to Program definitions. There are two exceptions: both a Parameter and a Table Column point to a Variable as their definition source. Figure B–3 shows all possible layers of object ownership within an Application Area.

Figure B–3 Object Ownership within a Work Area



Glossary

activity

An element of a Workflow: either an executable object such as a Program, a structural object such as a Fork, or a Notification.

adapter

An interface between Oracle LSH and another system. Oracle LSH includes adapters that allow you to load data into Oracle LSH from other systems, to transform data by creating Programs in integrated development enviroments, and to create visualizations of Oracle LSH data for display in other systems.

application

The set of all the defined objects required to perform a particular business function such as loading and analyzing data for a study or producing a particular set of reports.

Application Area

A container that is used to develop and manage the Oracle LSH objects required for a single business application (such as a study, project, or set of reports); contains one or more Work Areas where the set of object instances (such as Table and Program instances) necessary for the business application are installed.

Application Area library

A set of object definitions contained directly in an Application Area.

audit

A record of each change made to each record in a Table instance defined with a processing type that includes auditing, with the timestamp of the change and the user ID of the person who made the change. You can recreate the state of data in an audited table as it was at any point in time.

blind break

A single execution of a job on data defined as blinded in Oracle LSH, so that the sensitive blinded data is displayed in the job output, but is not permanently unblinded. Blind breaks may be required for patient medical emergencies or for reporting purposes during the course of a clinical trial.

blinded data

Data that must be hidden in Oracle LSH because its display would reveal patient treatment patterns in a double-blind clinical trial. If you specify that a table supports blinding, Oracle LSH creates a partition, allowing you to load the blinded data in the "real data" partition of the table and, optionally, to load dummy data in the "dummy data" partition. Oracle LSH requires standard object security privileges to run a Program on dummy data but requires additional security privileges to run the same Program on real (blinded) data.

browse

Navigate through the user interface looking for an output or definitional object.

Business Area

Oracle LSH definitional object used by a visualization tool such as Oracle Business Intelligence Enterprise Edition to create ad hoc, onscreen visualizations of Oracle LSH data.

check constraint

A check constraint applies to a particular Table Column and specifies allowed data values for that Column.

child object

An object contained in, or owned by, another object.

classification

(1) The act of associating an Oracle LSH object definition or object instance with one or more values in one or more classification hierarchies, for the purpose of categorizing the object so that it can be found during searching and browsing. (2) The actual classification hierarchy value or set of values associated with an Oracle LSH object definition or object instance. (3) The Oracle LSH subsystem of single- and multi-level hierarchies and their values used in searching and browsing.

classification hierarchy

Hierarchical or flat structure consisting of one or more levels, with one or more values in each level. If a hierarchy contains multiple levels, they must be logically related and the values in each level must be logically related to values in adjacent levels (each value must be related to one and only one value in the next higher hierarchy level, but can be related to many values in the next lower level). Used in searching and browsing.

cloning

A special copy operation for Work Areas that results in an identical Work Area, including all its contained object instances and mappings. All object instances in the clone have the same version number and validation status as their counterparts in the original Work Area, and the two Work Areas share a user-specified label.

constraint

A constraint on an Oracle LSH Table or Column similar to an Oracle column or table constraint. See also **check constraint**, **primary key constraint**, and **unique key constraint**.

Consumer

An Oracle LSH user who retrieves (consumes) information from the Oracle LSH database by running and viewing reports and creating onscreen data visualizations.

container object

An object that contains, or owns, another object; also called a parent object or organizational object.

Data Mart

(1) A large quantity of data extracted from one or more user-defined Oracle LSH Table instances in one of several formats. (2) The definitional object that generates the Data Mart output.

data source

An Oracle LSH Table instance or **source data system** table, view, or data set from which an Oracle LSH Program reads data.

data transformation

See **Program**.

date data type

The Oracle date data type is defined as the number of days since the beginning of the Julian Calendar in a floating point numeric. Oracle LSH stores all date values in this format but displays dates and accepts entry of dates in one of several European, US or Standard formats (such as DD-MON-YYYY) based on user preference.

Definer

An Oracle LSH user who develops applications by defining objects in Oracle LSH, typically a programmer.

definition

See **object definition**.

definitional object

Objects defined by the user in Oracle LSH, including Oracle LSH Tables, executables, and their components; includes both object definitions and object instances. See also object definition and **object instance**.

Definer

A LSH user who develops LSH business applications by defining objects and instantiating them in the database.

development environment

An installed Work Area whose usage intent is set to Development and whose database schemas are used for the purpose of developing Oracle LSH object definitions and instances. Oracle LSH supports maintaining separate Work Areas and schemas as development, quality control, and production environments so that data from a development or quality control environment is never mixed with production data.

Domain

A container that is used to group Application Areas and/or to store and control a library of object definitions.

Domain library

A set of object definitions contained directly in a Domain; intended as a collection of valid, production-quality object definitions suitable for reuse.

dummy data

Alternative data with blinded information obfuscated, processed instead of the real (sensitive, blinded) data for the purpose of testing Programs and producing reports prior to unblinding the real data.

Entry

See **Report Set Entry**.

execution

(1) The running of a particular Oracle LSH job; for example, running a Program to generate an output. (2) The Oracle LSH runtime subsystem that manages the execution of jobs.

Execution Setup

A defined object that is a component of each Oracle LSH executable object instance (Programs, Report Sets, Data Marts, Load Sets, Workflows) whose purpose is to control the execution of the executable object. It includes input Parameters whose values are either bound or settable at submission. The Execution Setup serves as the basis for the submission form of an executable object.

Execution Template

An Execution Setup that has been explicitly made available for use as a template to other instances of the same executable object definition version. Definers can use Execution Templates as the basis for an Execution Setup in another instance of the same executable object definition.

full reload processing

Processing mode for executables writing to Reload Tables in which all rows in a data source are reinserted in their current state by a Program or other executable object, and rows that are not reloaded are soft-deleted. Oracle LSH automatically detects which rows are new inserts, which are updates, which are unchanged and implicitly deletes any rows that are not reloaded.

IDE

See **Integrated Development Environment (IDE)**.

incremental reload processing

Processing mode for executables writing to Reload Tables in which only those rows that have been added or modified since the last time the same Program or other executable object was run are processed. Unchanged rows can also be reloaded and are ignored by the processing. Incremental reload processing does not delete records and therefore can be used to selectively load different subsets of data without impacting other data in the target table.

index

A non-unique index provides a way to optimize access to data in a table via columns included in the index. See **unique key constraint** and **primary key constraint**.

instance

See **object instance**.

Integrated Development Environment (IDE)

In Oracle LSH, an external source code development software application that can be launched directly from the Oracle LSH user interface for the purpose of writing or editing Program source code, integrated with Oracle LSH by a system that includes: a syntax-directed editor, graphical tools for program entry, integrated support for compiling and running the program, and a mechanism for relating compilation errors back to the source. Oracle LSH Release 2.4.8 supports IDEs for Oracle PL/SQL, SAS, and Oracle Reports.

installation

The process of converting selected Oracle LSH object instances in a single Work Area into installable components such as PL/SQL packages and DDL scripts and instantiating those components in an Oracle LSH schema.

library

See **Domain library** and **Application Area library**.

Load Set

A LSH executable whose purpose is to move data and meta-data into LSH from an external source data system that is connected to LSH by an adapter customized for the source data system. The adapter handles constraints on the structure of the target Table Descriptors, provides the Parameters, and supplies the source code for the installed Load Set.

mapping

(1) (noun) A mapping defines the relationship between a Table Descriptor in a Program or other executable object and a Table instance in a Work Area that the executable object reads from or writes to. Each column of the Table Descriptor and the Table instance must be mapped to a column of a compatible data type and length in the other, or to a constant. (2) (verb) Creating a mapping.

master job

A job launched by the submission of an Execution Setup. When a user submits a Load Set, Program, Data Mart, Report Set, or Workflow, the resulting job is a master job. Some master jobs have subjobs: Report Set executions include Program executions as subjobs. Workflow executions may include Program, Load Set, Data Mart, and/or Report Set executions as subjobs.

narrative

A block of text that annotates a generated report or report set. May precede the body of the report (pre-narrative) or follow it (post-narrative).

Notification

Message routed within Oracle LSH or (optionally) by email; contains a subject name, text, and one or more hyperlinks to Oracle LSH outputs or other objects; can be used for collecting informal approvals or comments or for broadcasting text.

object definition

The definitional meta-data details of an Oracle LSH object, stored in an Application Area or Domain library and created (normally) through an instance of the definition that is contained in a Work Area. The meta-data varies depending on the definition's object type, but includes a name, subtype, validation status and classification(s). For example, Table definitions also contain Columns and Constraints; Program definitions contain Source Code, Parameters, Planned Outputs, and Table Descriptors.

object instance

An instance of Oracle LSH object definition; contains a pointer to an object definition, which contains most of the definitional details, and a few attributes including a name, subtype, validation status, classification(s), and (for executable instances only) mappings. Object instances are located in a Work Area and installed in the database.

object subtype

Subcategory of a predefined object type that serves to differentiate classification and security requirements among objects of the same object type; for example, Financial Report Set and Clinical Report Set. Object subtypes have the same operations allowed as their object type. One subtype for each object type is shipped with Oracle LSH; an Administrator may define additional object subtypes.

object type

A predefined category of LSH definitional objects shipped with LSH; for example, Report Sets or Tables. Object instances constitute a separate object type; that is, a Report Set instance is a different object type from a Report Set definition and can have different classification, security, and validation requirements defined at the subtype level.

operation

An action that can be performed by a user on an object; for example, view or modify. Operations are predefined for object types. The LSH Administrator defines roles and associates them with operations on object subtypes as part of the LSH security system.

Oracle LSH schema

A set of Oracle database schemas (one primary schema and one or more auxiliary schemas) that owns all permanent objects that are populated by the installation of a single Oracle LSH Work Area, including tables, views, packages, or schema stores. An Oracle LSH schema also contains all data loaded into its tables.

organizational object type

Category of Oracle LSH object types used solely as containers of other objects; includes Domains and Application Areas.

output

In Oracle LSH, a file generated by the execution of a Program, Workflow, Report Set, or Data Mart; either the primary output reporting on or containing data, or a secondary output such as a log file or an error file. Also called an actual output to distinguish it from a Planned Output definition.

Overlay Template Definition (OTD)

A layout template for PDF outputs of Oracle LSH Report Sets; can include boilerplate text, graphical elements such as logos and lines, and placeholders for Parameter values for data (such as Study) and publishing specifications (such as font style and size).

Parameter

A defined object that acts as a simple scalar variable and is based on an Oracle LSH Variable definition. In a Program or other Oracle LSH executable object, an input Parameter supplies a runtime value to the executable object and an output Parameter holds a value generated during execution. In addition to a pointer to an Oracle LSH Variable, a Parameter contains a list of allowable values, rules for validating the supplied value, a mandatory/not mandatory setting, and classification(s). Parameter values can be propagated within the context of a Report Set or a Workflow.

Parameter Set

A definitional object created automatically by Oracle LSH to contain the Parameters contained in a single primary definitional object.

parent object

An object that contains, or owns, another object; also called a container object.

Planned Output

An object definition contained in an LSH executable object definition that serves as a placeholder for an actual output to be generated during execution and specifies the classification(s) of the actual output. Primary Planned Outputs report on or contain data; secondary Planned Outputs include log files and error files.

primary key constraint

A column or set of columns whose value(s) identify a row in a Table as unique; often a unique ID. The system enforces the following constraints on the data contained in primary key columns: values cannot be null, and the value or combination of values in primary key column(s) must be unique for each row.

privilege

In Oracle LSH, a privilege is an operation on an object subtype assigned to a user through a role in a user group. A user has the privilege necessary to perform an operation on an object when he or she is assigned to a user group that has access to the object, and within that user group is assigned to a role that allows the operation on the object's subtype.

production environment

An installed Work Area whose usage intent is set to Production and whose database schemas contain only valid Oracle LSH objects interacting with real clinical or other data in an audited and regulatory-compliant manner.

Program

An Oracle LSH defined executable object that functions as a computer program to transform data and/or generate one or more reports. Contains Source Code and one or more Table Descriptor(s); may also contain one or more Planned Outputs and Parameters.

report

(1) The primary output generated by a Program; contains a set of clinical or other data for in-house review or submission to regulatory agencies. Reports may be displayed as listings, figures, tables, or text and may be generated in one of several file formats. (2) A predefined output generated by the system on demand containing information about an object, set of objects, or process in LSH.

quality control (QC) environment

An installed Work Area whose usage intent is set to Quality Control and whose database schemas are used for the purpose of testing Oracle LSH object definitions, adapters, and loads of data and meta-data before using them in a production environment. Oracle LSH supports maintaining separate Work Areas and schemas as development, quality control, and production environments so that data from a development or quality control environment is never mixed with production data.

report

(1) The primary output generated by a Program; contains a set of clinical or other data for in-house review or submission to regulatory agencies. Reports may be displayed as listings, figures, tables, or text and may be generated in one of several file formats. (2) A predefined output generated by the system on demand; for example, a coversheet for a report.

Report Set

(1) A collection of reports generated by a single execution process and integrated with a table of contents. May be generated in PDF format using custom templates with graphics and including bookmarks and hyperlinks. (2) The primary Oracle LSH definitional object that produces the Report Set output.

Report Set Entry

One of the items in the table of contents of a Report Set; may contain other Report Set Entries, narratives, and/or one or more reports generated by a single Program; a Report Set chapter or subchapter.

role

(1) User-defined component of the Oracle LSH security system corresponding to a professional title or function; assigned to operations on object subtypes, to user groups, and to individual users within those groups. To perform a particular operation on an Oracle LSH object, a user must have a role assigned to that operation for the object's subtype in a user group with access to the object. (2) Predefined application roles shipped with Oracle LSH that control access to portions of the Oracle LSH user interface.

runtime

The point in time when a user submits an executable object for execution.

runtime subsystem

The Oracle LSH subsystem that manages the execution of installed executable object instances as well as the tracking of, and access to, the progress and results of the executions.

search

Facility for finding existing defined objects and outputs in Oracle LSH, using criteria including object type, name, and classifications (including keywords). Oracle LSH includes simple and advanced search options.

snapshot

A set of data, or data and objects, at a particular point in time.

Source Code

(uppercase) A secondary definitional object owned by an Oracle LSH Program containing a file of user-written computer instructions (**source code**) whose purpose is to perform a task.

source code

(lowercase) The computer instructions used by a Program to perform a task. Source code may be written in an Integrated Development Environment (IDE) in the context of an Oracle LSH Program or written outside Oracle LSH and then uploaded and stored in Oracle LSH; in Oracle LSH, source code is stored in the definitional object called **Source Code**.

source data

Data collected in another system but loaded into or accessible from LSH.

source data system

An application external to, but integrated with, Oracle LSH that collects data loaded into or accessed by Oracle LSH through an adapter.

Source Table Descriptors

A **Table Descriptor** used to link a Program to a Table instance containing data used as input to the Program.

staging table processing

Data processing type used for Programs that write to Tables defined as Staging. If the Table is defined as Not Audited, each time the Program that writes to it is executed, the system hard-deletes the data resulting from the previous execution. If the Table is defined as Audited, the system saves a copy of the complete set of data resulting from the previous execution, but operates only on current data.

Structure

See **Workflow Structure** and

Table

an Oracle LSH meta-data representation of a table-like object (Oracle view or SAS dataset); includes a description of the object as a whole, its columns, and constraints that apply to one or more of its columns. Appears as an Oracle view to all installed Programs except those based on SAS technology, where it appears as a SAS dataset or SAS view.

Table constraint

See **constraint**.

Table Descriptor

An instance of an Oracle LSH Table defined within an Oracle LSH executable object—Load Set, Program, Workflow, Report Set, or Data Mart—or in a Business Area, for the purpose of linking the executable object or Business Area to an installed Table and its data by mapping.

Table structure

The number of Columns a Table definition contains and the data type and length of each one.

Target Table Descriptors

A **Table Descriptor** used to link a Program to a Table instance to which the Program writes data.

test environment

See **quality control (QC) environment**.

transform, transforming Program

See **Program**.

transactional processing

Data processing type used to write to Table instances defined as transactional. Programs writing to transactional Tables must use explicit Insert, Update, and Delete commands in their source code.

transition

An element of a Workflow that defines one workflow activity as sequential to another and specifies the condition, if any, for the execution of the second.

unblind

Change a Table instance's blinding status from Blinded to Unblinded.

unique key constraint

A column or combination of columns that identifies a row in a Table as unique; similar to a primary key except that null values are allowed in columns that are part of the unique key. You can explicitly disallow null values at the column level.

usage intent

An attribute of Work Areas indicating the Work Area's purpose; either Development, Quality Control, or Production. Oracle LSH enforces rules based on the interaction of a Work Area's usage intent attribute and the validation status of the Work Area and the object instances contained in it.

user group

Unit defined by an Oracle LSH Administrator for use in security that includes one or more users and one or more roles, and is assigned to one or more Oracle LSH objects. To perform an operation on an Oracle LSH object or output, a user must belong to a user group assigned to that object and be assigned a role within that user group that permits the operation on the object's subtype.

validation

(1)Process through which object definitions and instances reach the stage where they can be declared stable and valid according to an Oracle LSH customer's policies. (2) Process required by regulatory agencies to certify that a computer system used in clinical trials is stable and functions properly.

validation status

An object attribute that indicates the life cycle stage of the object: Development, Quality Control, Production, or Retired. Oracle LSH enforces rules based on the interaction of a Work Area's usage intent attribute and the validation status of the Work Area and the object instances contained in it.

Variable

An Oracle LSH defined object equivalent to a SAS variable or Oracle table column that serves as a source definition for Oracle LSH Parameters and Table Columns. Oracle LSH creates Variables from Oracle table columns and SAS dataset variables the first time a particular Oracle table or SAS dataset is loaded into Oracle LSH. The user can define Variables during Parameter or Table Column definition.

version

Restorable, saved state of an object definition or instance.

visualization

An ad hoc graphical and/or text presentation of data created in an interactive onscreen environment.

Work Area

A container in an Application Area where Definers create the object instances required to support the business purpose of the Application Area. Work Areas have two special operations: **installation**, during which some or all of the object instances contained in the Work Area are instantiated or upgraded in an Oracle LSH schema and the Oracle LSH schema is created if necessary; and **cloning**, which creates a duplicate Work Area that shares a label with the original Work Area.

Workflow

An executable Oracle LSH defined object that includes other Oracle LSH executable defined objects such as Load Sets, Programs, Data Marts, and Report Sets, as well as Notifications; executed as a whole in a defined sequence that can include conditional branching.

Workflow Structure

Predefined object that controls the execution of the Workflow; a Workflow Structure can detect the completion status of the previous activity and fire the next one or more activities as specified for their type: Start, End, And, Or, Fork.

XML Publisher

Oracle product integrated with Oracle LSH for use in generating PDF-format Report Sets and enabling the use of custom overlay templates. Also used to produce internal PDF-format reports.