tranSMART

ETL Analyst’s Guide

Johnson & Johnson

BIOTECHNOLOGY | IMMUNOLOGY | ONCOLOGY

January 25, 2011 v.1

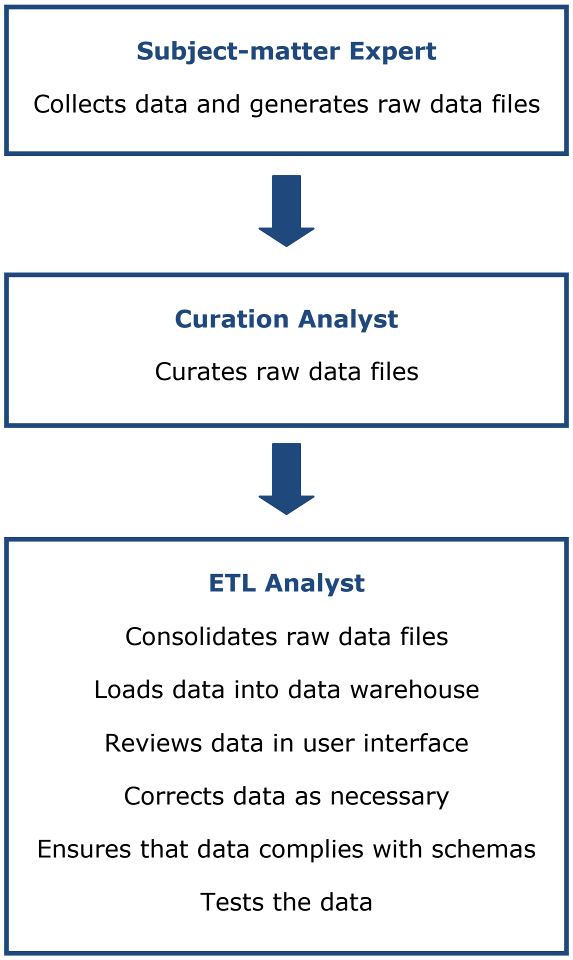
Recombinant

Contents

Chapter 1 Introduction

As the tranSMART Extraction, Transformation, and Loading (ETL) Analyst, you are responsible for loading data into the tranSMART data warehouse. Medical researchers, literature curators, and other subject matter experts in the field provide data files for loading into the tranSMART data warehouse. You are responsible for transforming the data files, cleaning them, and loading them into the warehouse.

The high-level workflow is as follows:


Subject-matter Expert
Collects data and generates raw data files
Curation Analyst
Curates raw data files
ETL Analyst
Consolidates raw data files
Loads data into data warehouse
Reviews data in user interface
Corrects data as necessary
Ensures that data complies with schemas
Tests the data

The Johnson & Johnson Data Steward has overall responsibility for all aspects of the data collection and loading workflow.

ETL Analyst's Contacts

The following file contains the names of key people involved in the data collection and loading workflow:

DataLoadContacts.txt

tranSMART Data Warehouse

The tranSMART data warehouse contains data loaded from a number of different sources – for example, Johnson & Johnson clinical trials, external (to Johnson & Johnson) experiments, biomarker data such as RNA, RBM, and SNP metrics, gene and pathway details, information from peer-reviewed journals, and references to Johnson & Johnson documents.

End-users access all of this data through the tranSMART user interface.

tranSMART Topography

There are actually three separate data warehouses on three separate database servers. Each data warehouse is associated with a different application server, and is accessed by a different kind of user:

- Application developers have access to a development application server and a development database server. This is the database server where you load, correct, and transform the raw data you receive from the Curation Analyst.
- QA staff and project leaders have access to a staging (QA) application server and a staging database server. This is the database server where you run test scripts against the data to ensure that the data is production-ready.
- End-users access production-ready data on the production database server, using tranSMART on the production application server. You place the data on the production database server after you have corrected, transformed, and tested it.

Data Warehouse Schemas

The data warehouse consists of Oracle databases located on the Amazon Web Services (AWS) cloud. The following table summarizes the schemas you will work with on the cloud and the servers where they reside:

Schema	**Description**	**Database Server Location**		
		Development host: devdb*	**Staging/QA** host: tmqa*	**Production** host: proddb*
BIOMART	Data accessed by tranSMART end-users. **Note:** You must log in as Oracle user `biomart_user` (both user ID and password) to see this database.	✓	✓	✓
BIOMART_LZ	Landing zone for new data.	✓		
BIOMART_USER	tranSMART entry point into the data warehouse. **Note:** You must log in as Oracle user `biomart` (both user ID and password) to see this database.	✓	✓	✓
BIOMART_WZ	Working zone where data is manipulated and tested prior to loading to BIOMART.	✓		
CENTCLINRD	Data from curated oncology literature.	✓		
CONTROL	Storage for all ETL code and tables.	✓		

Schema	Description	Database Server Location		
		Development host: devdb*	Staging/QA host: tmqa*	Production host: proddb*
DEAPP	tranSMART extension of i2b2. Includes transactional information such as biomarker data and security for clinical trials.	✓	✓	✓
DEAPP_WZ	Dataset Explorer working zone.	✓		
I2B2_LZ	i2b2 landing zone.	✓		
I2B2_WZ	i2b2 working zone.	✓		
I2B2DEMODATA	I2B2standard database.	✓	✓	✓
I2B2HIVE	I2B2standard database.	✓	✓	✓
I2B2METADATA	I2B2standard database.	✓	✓	✓
JBL_ASTHMA	Data from curated asthma literature.	✓		
JBL_JWB	Data from curated oncology literature.	✓		
OMICSOFT_LZ	Omicsoft raw data.	✓		
OMICSOFT_RAW	Omicsoft raw data.	✓		
PICTOR	Pictor raw data.	✓		
REFERENCE	Reference data such as for genes and pathways.	✓		

Schema	Description	Database Server Location		
		Development **host: devdb***	**Staging/QA** **host: tmqa***	**Production** **host: proddb***
SEARCHAPP	Transactional information about the users who are allowed to access tranSMART, such as the queries they've run and the permissions they are granted.	✓	✓	✓
WEBARRAY	Dana Farber raw data.	✓		
WEBARRAY_READ	Dana Farber raw data.	✓		
*Full host name: ***host*.jnj.recomdata.com**.				

Tools and Requirements

To access the Oracle databases in the data warehouse, you need the following:

- A user ID and password to log into each of the Linux database servers in the data warehouse.

  This document assumes you will authenticate on the database servers through a private key file (`.ppk`) located on your Windows desktop.

- An SSH tunnel from a port on your PC to the database servers in the data warehouse.

  This document assumes you will use PuTTY to set up the tunnel.

- A connection to the databases in the data warehouse

  This document assumes you will use Oracle SQL Developer to connect to the databases and manage the data.

One-Time Setup

This section describes the procedures you perform to set up a tunnel to a database server and to establish a connection to the databases on the server. Establishing a tunnel allows you to use a desktop tool such as Toad or Oracle SQL Developer to access the database.

Step 1. Set Up a Tunnel to a Database Server

The instructions below assume the following:

- You have a `ppk` file on your Windows desktop containing your private ssh key, and the system administrator has registered your public key on the cloud servers. (If you have not generated a private key and registered your public key, please contact the system administrator.)
- You are using PuTTY as your SSH client.

Set up a tunnel for each of the following database instances:

▨	**Host Name**	**IP Address**	**Oracle SID**	**Purpose**
Development server	devdb	174.129.237.81	DW1	Edit the data loaded into the warehouse to ensure that it is correct and complete.
Staging server	tmqa	174.129.234.96	tmqa	Test the edited data before moving it to the production server.
Production server	proddb	174.129.240.115	DW2	Fully edited and tested data. This is the data that end-users access using tranSMART.

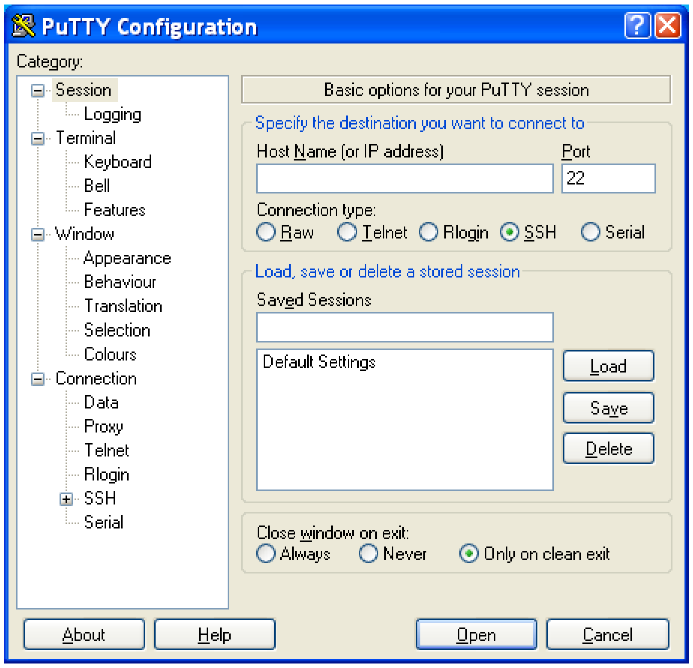
Note: For a list of the databases that reside on each server, see Data Warehouse Schemas on page 3.

Note: There is also a database server used exclusively for training purposes. The data on this server is static. However, if you need to access the data, the host name is `trainingDB`, the IP address is `184.73.224.77`, and the Oracle SID is `dw4tn`.

The following instructions describe how to use PuTTY to set up a tunnel from your client PC to the development database server:

1. Run PuTTY.

   The PuTTY Configuration window appears:



Connection type: Raw Telnet Rlogin [✓ SSH] Serial

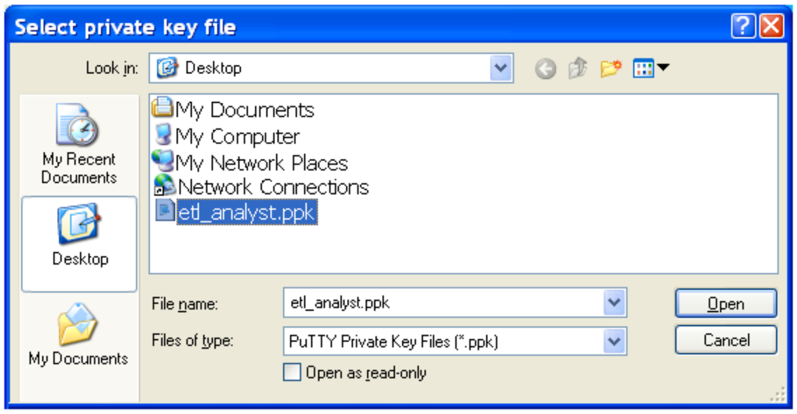
Close window on exit: Always Never [✓ Only on clean exit]

2. In **Host Name (or IP address)**, type **devdb.jnj.recomdata.com**.

3. In the navigation pane on the left, click **Connection > SSH > Auth**.

4. Click the **Browse** button after the **Private key file for authentication** field.

5. In the **Select private key file** dialog, find and select your `ppk` file (**etl_analyst.ppk** in the figure below), then click **Open**.



File name: etl_analyst.ppk

Files of type: PuTTY Private Key Files (*.ppk)

6. In the PuTTY navigation pane, click **Connection > SSH > Tunnels**.

7. In **Source port**, type a client port number for the connection between the SQL client and the host. (In this example, port **1520** is specified.)

Note: If you intend to establish simultaneous connections to data warehouse servers, the source port must be unique for each connection.

8. In **Destination**, type the following:

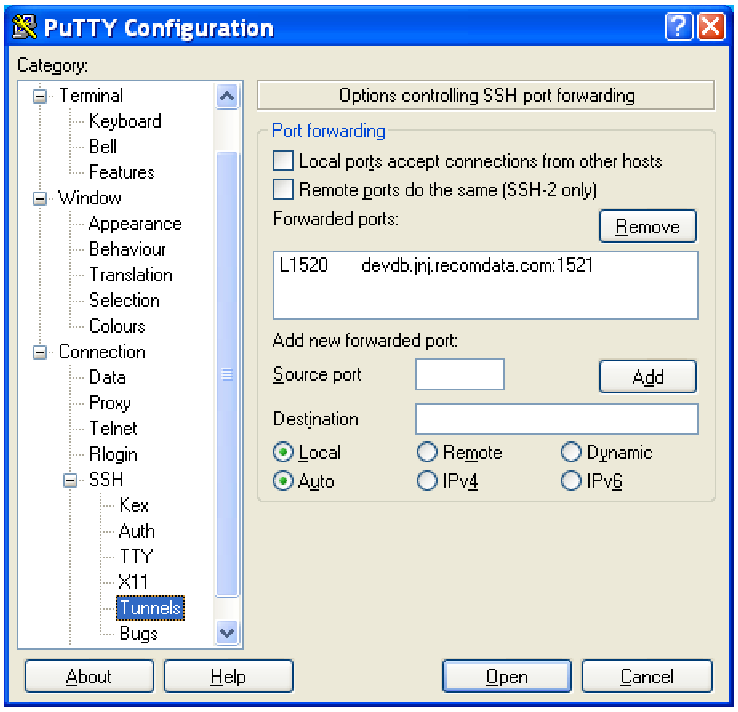
   HostName:1521 or *HostIP*:1521

   For example:

   ```
   devdb.jnj.recomdata.com:1521
   ```

9. Click **Add**.

   The PuTTY window now looks similar to the following:



Options controlling SSH port forwarding

Port forwarding

Local ports accept connections from other hosts [ ]

Remote ports do the same (SSH-2 only) [ ]

Forwarded ports:

L1520 devdb.jnj.recomdata.com:1521

Add new forwarded port:

Source port

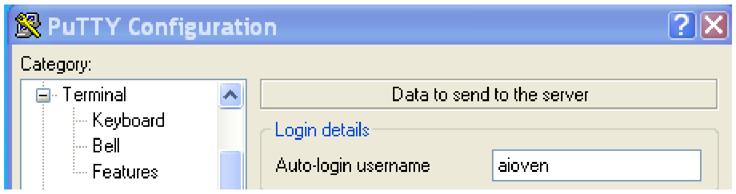
Destination

[✓ Local] Remote Dynamic

[✓ Auto] IPv4 IPv6

10. In the navigation pane, click **Connection > Data**.

11. In the **Auto-login username** field, type your login name for the database server:



Data to send to the server

Login details

Auto-login username: aioven

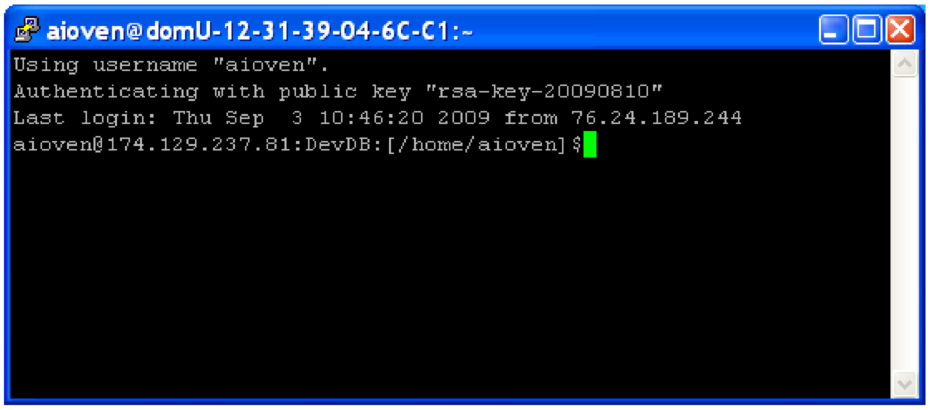
12. In the navigation pane, click **Session**.

13. In the **Saved Sessions** field, type a name for this connection – for example, **devdb**.

14. Click **Save**.

15. Click **Open**.

    PuTTY uses your login name to establish a session with the database server, and then PuTTY closes. A terminal window for your database server session opens:



```
Using username "aioven".
Authenticating with public key "rsa-key-20090810"
Last login: Thu Sep  3 10:46:20 2009 from 76.24.189.244
aioven@174.129.237.81:DevDB:[/home/aioven]$
```

16. Keep the session open, so that you can establish a connection to the data warehouse in the next section.

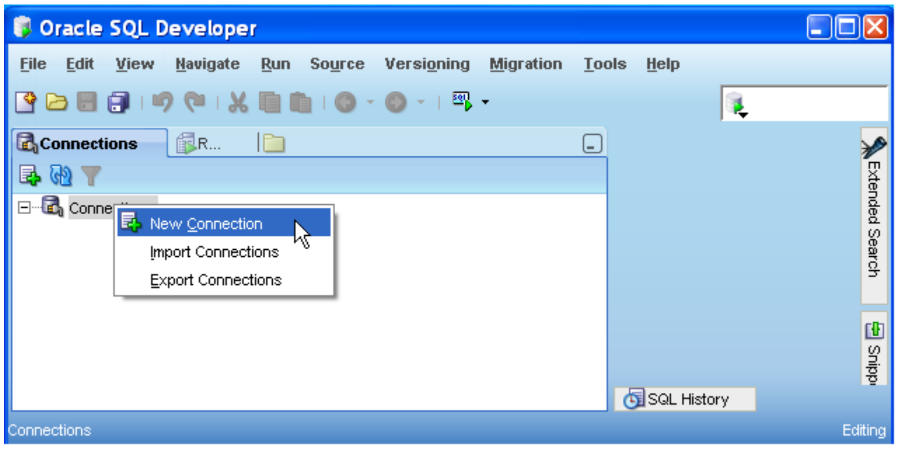
Step 2. Establish a Connection to the Databases on a Server

The instructions below assume the following:

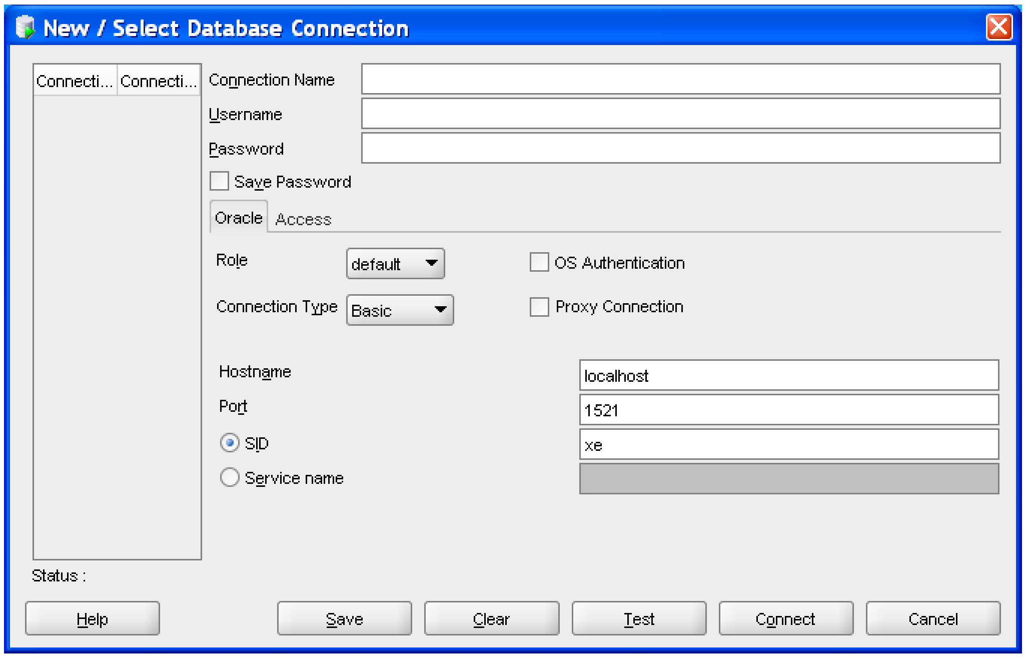
- You have an open tunnel to the database server **devdb** (174.129.237.81).
- You are using Oracle SQL Developer locally to establish connections to the databases on the server (rather than TOAD).

To establish a connection to the databases on a database server:

1. Run SQL Developer.
2. Right-click **Connections**, then click **New Connection**:



The New / Select Database Connection dialog appears.


New / Select Database Connection

Connection Name
Username
Password
☐ Save Password

Role: default
Connection Type: Basic
☐ OS Authentication
☐ Proxy Connection

Hostname: localhost
Port: 1521
(●) SID: xe
( ) Service name

Status :

3. In **Connection Name**, type a name for the new connection – for example, **Dev_User**.
4. In **Username** and **Password**, type the Oracle login credentials for the databases in the data warehouse on this server – for example, type **biomart** in each field.

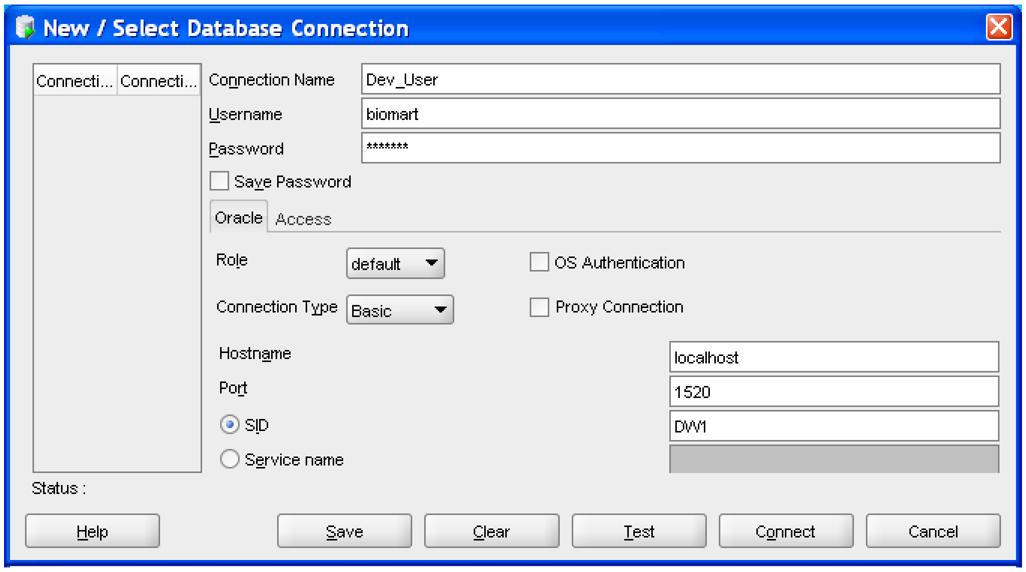
> **Note:** A number of Oracle user IDs and passwords have been created for you. In each set of Oracle credentials, the user ID and password are identical, and are based on the name of a database in the data warehouse – for example:

Database	**User ID**	**Password**
BIOMART	biomart	biomart
BIOMART_LZ	biomart_lz	biomart_lz
BIOMART_WZ	biomart_wz	biomart_wz

5. In the **Port** field, type the client port number for the tunnel. This is the port number you specified in step 7 on page 7 of the previous example.
6. Select the **SID** radio button and type **DW1** in the **SID** field.

> **Note:** The SID for each of the database servers is listed in the table in section Step 1. Set Up a Tunnel to a Database Server on page 6. DW1 is the SID for the database server devdb.

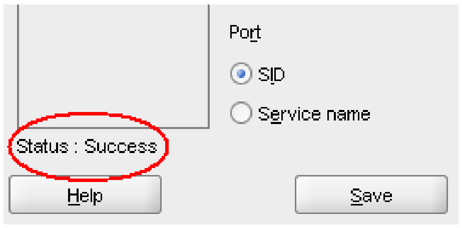
The New / Select Database Connection dialog should look as follows:



Connection Name: Dev_User
Username: biomart
Password: *******
Save Password
Role: default
OS Authentication
Connection Type: Basic
Proxy Connection
Hostname: localhost
Port: 1520
SID: DWI
Service name
Status :

7. Click **Test**.

   The status message **Success** appears in the lower left corner of the dialog:



Status : Success

8. Select **Save Password**.

   Saving your password lets you open this connection in the future without having to provide Oracle database credentials.

9. Click **Save** to save the connection.

   Note: At this point, you can click **Connect** to connect to the databases and begin working with the data. However, the next section will describe the more typical workflow of logging into a database server through an existing tunnel and opening an established database connection.

10. Click **Cancel** to close the New / Select Database Connection dialog.

11. Close Oracle SQL Developer.

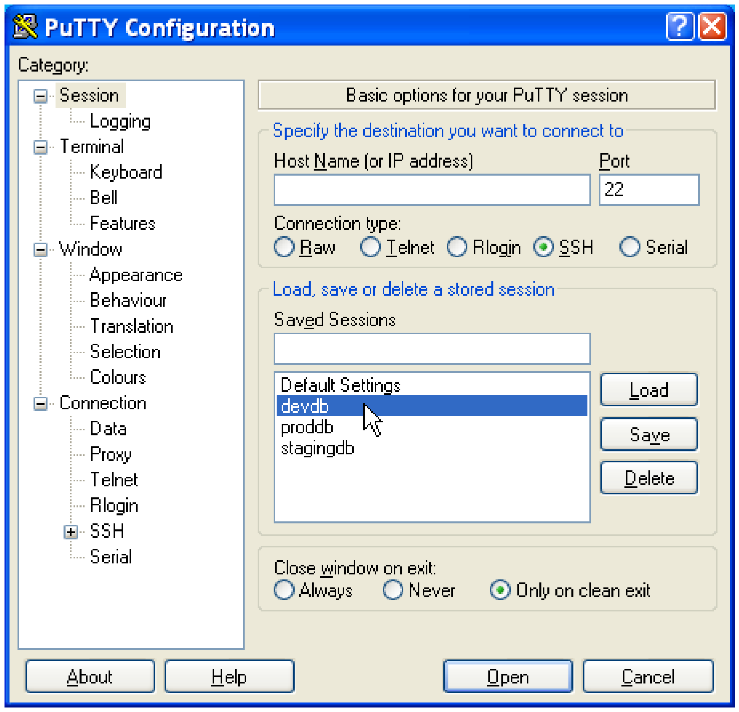
12. Close the connection to the database server by closing its terminal window.

Opening an Established Connection

Once you have set up tunnels to the database servers and have established connections to the databases in the data warehouse, you can open a connection to the databases.

To open an established connection:

1. Run PuTTY.
2. Double-click a saved session – for example, **devdb** in the following figure:



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions

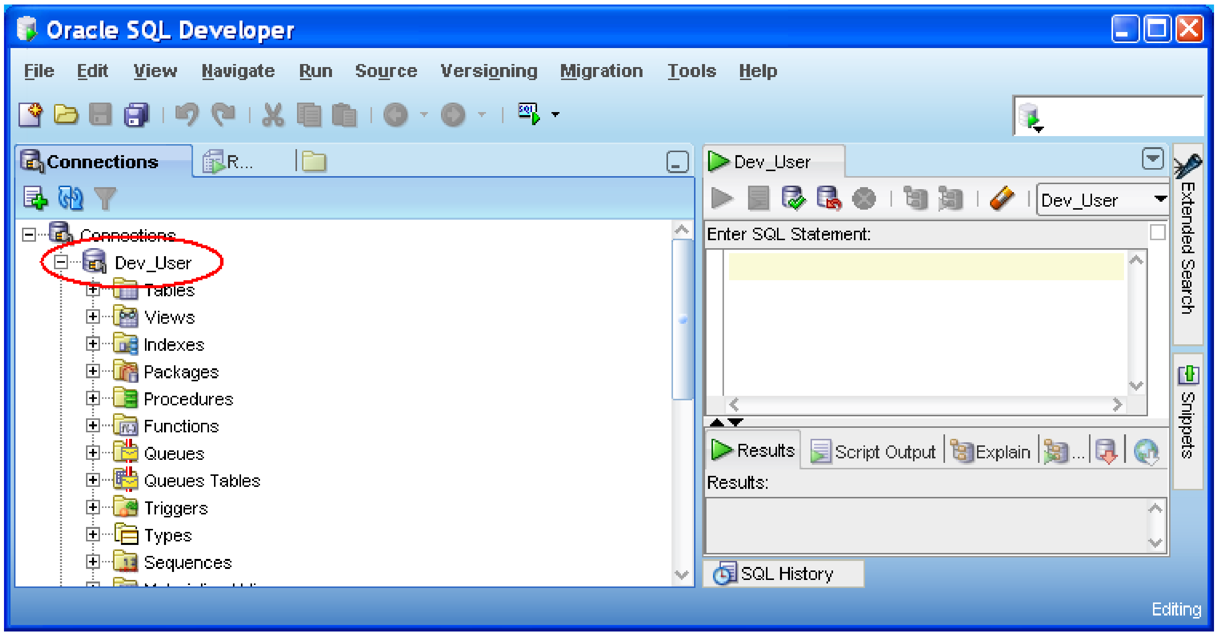
Default Settings
devdb
proddb
stagingdb

Close window on exit: Always Never [✓ Only on clean exit]

   PuTTY uses the login name associated with the saved session to establish a connection to the database server.

3. Run Oracle SQL Developer.
4. Click the **+** icon (⊞) next to the name of a connection to open the connection.

In the following figure, the Dev_User connection has been opened:



You are now ready to manage the data in the data warehouse on this server.

Chapter 2
Responsibilities and Workflow

Your responsibilities as ETL Analyst begin after data has been collected from subject matter experts in the field, curated, and handed off to you.

You are responsible for all data management tasks, from loading the data in the tranSMART data warehouse through making the data available to tranSMART end-users. Responsibilities of the ETL Analyst include:

- Tracking the progress of data loading to assure that the system is loaded completely and efficiently.
- Profiling the incoming data.
- Defining the data according to database schemas.
- Validating metadata.
- Reviewing data in the tranSMART user interface and making corrections as necessary.
- Moving the data between functional zones and servers.
- Performing quality assurance tasks.

Data Warehouse Zones and Environments

The data warehouse is divided into distinct zones, spread out over three servers. The purpose of these separate areas is to ensure that raw data can be continually loaded into the data warehouse and transformed into correct, complete, and schema-compliant data, without affecting users of the production system.

The three servers where the zones reside are called environments. There are three environments – a development environment, a staging/QA environment, and a production environment. Data loading and transformation take place in the development environment (that is, on the development server). Data testing takes place in the staging/QA environment. The final, production-quality data that tranSMART end-users access resides in the production environment.

The following table summarizes these functional zones and environments, and indicates the servers where the zones reside:

Zone	Function	Environment	Server Host Name (IP)
Landing	External data in various formats is curated and reformatted into platform-specific, unified formats and put into this data area. Data may be transformed further in this area before loading into the working zone.	Development	devdb (174.129.237.81)
Working	Core schema and a set of temporary tables. In this zone, data is transformed and loaded into the core schema and data marts.	Development	devdb (174.129.237.81)
Data Warehouse	A clean copy of the working zone without any temporary tables.	Development	devdb (174.129.237.81)
Staging/QA	A copy of core data warehouse and data marts to be used as the QA database, and also as the staging zone for the production environment	Staging/QA	tmqa (174.129.234.96)
Production	A copy of core data warehouse and data marts to be used as the production-quality database for access by tranSMART end-users.	Production	proddb (174.129.240.115)
Reference	A local collection of reference data in its original schema. Database links are also used to reference Johnson & Johnson internal data sources.	All environments	All servers
Control/Curation	Data heritage and lineage information, as well as data curation information.	All environments	All servers

Workflow

The following steps summarize your typical workflow as ETL Analyst for loading, transforming, and moving a set of data through the various data warehouse zones:

1. The Curation Analyst provides you with raw source data. You then consolidate the data into a file such as a Microsoft Excel file, a tab- or comma-separated values file, or a SAS file.

2. Introduce the raw data into the data warehouse by loading it into the landing zone – for example, database `BIOMART_LZ`.

   Data should be loaded as raw character data (`VARCHAR2` fields) whenever possible, rather than loaded into numeric or date formats.

   > **Note:** The reason for loading raw data into the landing zone is to restructure the format of the data (such as pivoting the data) to conform to database structure. However, some source data (such as ResNet data, and web array data from Dana Farber), is already in database format when it's provided to you. This data can be loaded directly into the working zone.

3. Move the data from the landing zone to the working zone – for example, database `BIOMART_WZ`.

   In the working zone, you structure the data to fit a particular database schema. Thus, the working zone consists of the data and control tables that define where the data is loaded within the schema.

   You perform a number of other tasks in the working zone, such as validating the metadata and ensuring that the data presentations in the tranSMART user interface are as the Curation Analyst wants them to be.

4. Move the transformed data from the working zone into the data warehouse – for example, database `BIOMART`.

   The data warehouse zone houses the most up-to-date records, consisting of information that is integrated across all of the data sources. This zone also consists of data marts – that is, "captures" of subsets of the data warehouse that are tailored to a specific application's needs.

   The landing, working, and data warehouse zones constitute the development environment side of the project. The next two zones, the staging/QA and production zones, reside in the staging/QA and production environments, respectively. These two zones are physically isolated from each other by being housed on separate servers.

5. Move the data in the data warehouse to the staging/quality assurance(QA) zone. This zone houses dated snapshots of the data warehouse or data marts, along with a separate instance of tranSMART to query or explore the data.

   The staging area allows you to do extensive testing of the data in a simulated production environment. When you find problems during testing, you need to return the problem data to the work are for editing.

6. When testing is complete, move the production-quality data to the production zone, where tranSMART end-users access the data.

The Reference and Control/Curation Zones

The reference and control/curation zones reside conceptually outside the development, staging/QA, and production environments, and also outside a typical workflow. These two zones contain data that may not be cast into a single zone.

Reference Zone

The data in the reference zone can be accessed by any of the other zones in the development, staging/QA, and production environments.

An example of the data that this zone contains is biological relationship information that changes on a different frequency than experimental data.

Control/Curation Zone

The control/curation zone houses annotations to the data, which help clarify foreign concepts.

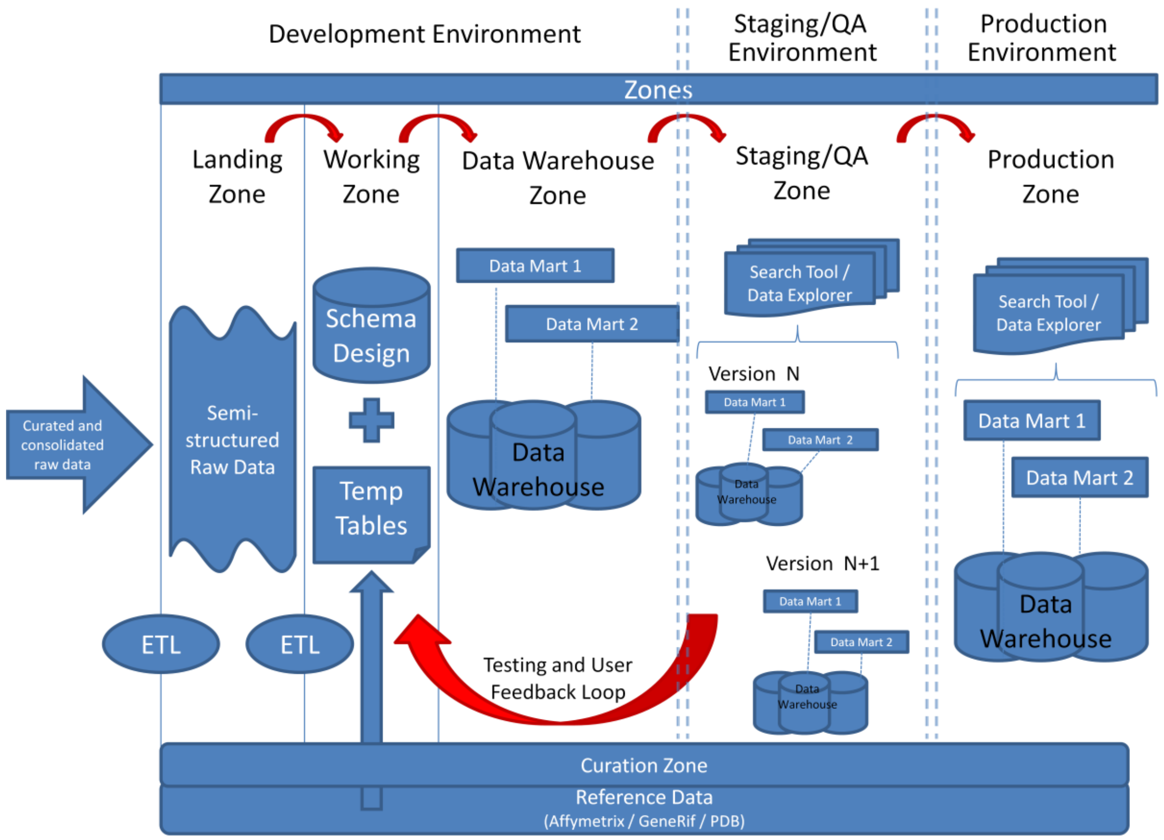
Like the reference zone, this zone spans all three environments. This zone is also accessed by other zones (the landing and working zones) because it contains the custom annotations to the data that enhance the content, but that may need to be previewed by tranSMART end-users in the context of the overall data warehouse.

This zone also contains control information that crosses between zones.

By placing this zone outside of the loading or working zones, the data can be used (loaded, previewed, and edited) across multiple zones.

Workflow Illustration

The following illustration summarizes the workflow that you, as ETL Analyst, are responsible for:



Tables in the Data Warehouse

The following sections summarize the tables in the tranSMART data warehouse. The tables are grouped by core data warehouse tables, tables used by the tranSMART Search feature, and tables used by Dataset Explorer.

Core Data Warehouse

Table Name	**Used For**	**Description**
`BIO_ASSAY`	Assay Data and Expression Profile	Information such as ID, type, and platform about the bio assays used in a clinical trial.
`BIO_ASSAY_ANALYSIS`	Basic Bio Facts	Attributes of an analysis, such as analysis name, analysis description, analysis method, and assay type.
`BIO_ASSAY_ANALYSIS_DATA`	Basic Bio Facts	Analysis information such as results data, p-value, preferred p-value, and fold change value.
`BIO_ASSAY_ANALYSIS_DATA_TEA`	Basic Bio Facts	Similar to `BIO_ASSAY_ANALYSIS_DATA`, but includes only results ranked as statistically significant, based on the TEA algorithm.
`BIO_ASSAY_DATA`	Assay Data and Expression Profile	Test information such as assay ID and values. This is raw data from an experiment.
`BIO_ASSAY_DATA_ANNOTATION`	Assay Data and Expression Profile	Assay feature group ID and associated biomarker ID.
`BIO_ASSAY_DATA_STATS`	Assay Data and Expression Profile	Statistical test data such as minimum and maximum values and standard deviation.
`BIO_ASSAY_DATASET`	Assay Data and Expression Profile	Analysis attributes such as ID, name, description, and criteria.
`BIO_ASSAY_FEATURE_GROUP`	Assay Data and Expression Profile	ID, name, and type of a feature group.
`BIO_ASSAY_PLATFORM`	Assay Data and Expression Profile	Attributes of assay platforms (for example, Affymetrix – HG-U133-PLUS [GPL570]). Attributes include ID, name, description, organism, and vendor.

Table Name	**Used For**	**Description**
BIO_ASSAY_SAMPLE	Assay Data and Expression Profile	Assay ID, sample ID, and study timepoint ID.
BIO_ASY_ANALYSIS_DATA_ALL	Basic Bio Facts	Intermediate table that contains raw analyzed data to be processed and filtered. When these operations are complete, data is loaded into BIO_ASSAY_ANALYSIS_DATA.
BIO_ASY_ANALYSIS_DATASET	Basic Bio Facts	Link between dataset ID and analysis ID.
BIO_ASY_ANALYSIS_PLTFM	Basic Bio Facts	Analysis platform (such as Jubilant, Omicsoft, and Johnson & Johnson) and platform ID.
BIO_ASY_DATA_STATS_ALL	Assay Data and Expression Profile	Intermediate table that contains raw statistical data to be filtered. When these operations are complete, data is loaded into BIO_ASSAY_DATA_STATS.
BIO_CELL_LINE	Basic Bio Facts	Contains data related to gene signatures, such as cell line name and ID, and information about the associated disease. Data is loaded from the table CELLLINESDICT in database CENTCLINRD.
BIO_CGDCP_DATA	Assay Data and Expression Profile	Cancer Gene Data Curation Project data, such as evidence code, negation indicator, and NCI disease concept code.
BIO_CLINC_TRIAL_ATTR	Experiments and Clinical Trials	Clinical trial or experiment ID, attribute code, attribute value.
BIO_CLINC_TRIAL_PT_GROUP	Experiments and Clinical Trials	Group attributes such as ID, name, description, number of patients, and type code.
BIO_CLINC_TRIAL_TIME_PT	Experiments and Clinical Trials	Time point attributes such as start and end times and dates.
BIO_CLINICAL_TRIAL	Experiments and Clinical Trials	Master metadata for clinical trials.
BIO_COMPOUND	Assay Data and Expression Profile Experiments and Clinical Trials	Master table for internal and external compounds.

Table Name	Used For	Description
BIO_CONCEPT_CODE	Assay Data and Expression Profile Biomarkers Experiments and Clinical Trials	Defines all types of concepts (for example, organism, gene, tissue type) that are used in the entire data warehouse.
BIO_CONTENT	Content and Repository	Content references for resources in files such as `.pdf` , `.xls`, and `.doc`.
BIO_CONTENT_REFERENCE	Content and Repository	Content references (such as bio content ID and bio data ID) that represent bio objects in the content management system.
BIO_CONTENT_REPOSITORY	Content and Repository	IDs and locations of repository resources, such as trials, internal documents, and PubMed articles.
BIO_CURATED_DATA	Basic Bio Facts	Reference IDs for curated literature resources.
BIO_CURATION_DATASET	Basic Bio Facts	Information about curated literature datasets.
BIO_DATA_ATTRIBUTE	Assay Data and Expression Profile	Bio data ID, attribute code, attribute value.
BIO_DATA_COMPOUND	Basic Bio Facts	Link between data IDs and bio compounds relating to a study or experiment.
BIO_DATA_CORREL_DESCR	Biomarkers	Correlation types (such as pathway to disease, or pathway to gene) and descriptions.
BIO_DATA_CORRELATION	Biomarkers	Correlation IDs.
BIO_DATA_DISEASE	Assay Data and Expression Profile Basic Bio Facts	Links between bio data IDs and bio disease IDs relating to an experiment or study. Provides a hierarchical structure of MeSH diseases.
BIO_DATA_EXT_CODE	Assay Data and Expression Profile Basic ID Facts Biomarkers	Synonyms for bio data (such as genes, pathways, and compounds), bio data sources (such as Ariadne URN), and disease terms.

Table Name	Used For	Description
BIO_DATA_LITERATURE	Basic Bio Facts	Link between BIO_DATA_ID and the IDs BIO_LIT_REF_DATA_ID (table BIO_LIT_REF_DATA) and BIO_CURATION_DATASET_ID (tables BIO_CURATED_DATA and BIO_CURATION_DATASET).
BIO_DATA_OMIC_MARKER	Basic Bio Facts	Link between bio data IDs and biomarker ID. This is the tranSMART annotation table.
BIO_DATA_UID	Assay Data and Expression Profile Biomarkers Experiments and Clinical Trials	Link between bio data IDs and unique IDs (such as EXP:Omicsoft: GSE2123).
BIO_DISEASE	Assay Data and Expression Profile Basic Bio Facts Experiments and Clinical Trials	A disease master table with cross reference to ICD 9/10 and MeSH coding systems.
BIO_EXPERIMENT	Experiments and Clinical Trials	Metadata about a particular experiment with attributes such as investigator, experiment type, and completion date.
BIO_LIT_ALT_DATA	Basic Bio Facts	Alterations data from curated asthma and oncology literature.
BIO_LIT_AMD_DATA	Basic Bio Facts	Molecule alterations data from curated oncology literature.
BIO_LIT_INH_DATA	Basic Bio Facts	Inhibitors data from curated oncology literature.
BIO_LIT_INT_DATA	Basic Bio Facts	Interactions data from curated asthma and oncology literature.
BIO_LIT_INT_MODEL_MV	Basic Bio Facts	Materialized view that associates IDs for Jubilant interactions data and experimental model types (such as homo sapiens or cavia porcellus).
BIO_LIT_MODEL_DATA	Basic Bio Facts	Information about the experiment that generated the asthma or oncology data described in the curated literature.

Table Name	Used For	Description
BIO_LIT_PE_DATA	Basic Bio Facts	Protein effects data from curated asthma literature.
BIO_LIT_REF_DATA	Basic Bio Facts	Information about the referenced literature, such as title, description, and disease type.
BIO_LIT_SUM_DATA	Basic Bio Facts	Summary of alterations data from curated oncology literature.
BIO_MARKER	Basic Bio Facts Biomarkers	Master table for biomarkers, with attributes such as ID, type, organism, and description.
BIO_MARKER_CORREL_MV	Basic Bio Facts Biomarkers	Materialized view that associates biomarker IDs by correlation type, such as gene and homologene.
BIO_PATIENT	Experiments and Clinical Trials	Master table for patient data with attributes such as subject ID, experiment ID, gender, and race.
BIO_PATIENT_EVENT	Experiments and Clinical Trials	Information about a patient event, including event code, type, date, timepoint ID, and site.
BIO_PATIENT_EVENT_ATTR	Experiments and Clinical Trials	Link between patient attribute ID, patient event ID, clinical trial ID, and the patient event attributes.
BIO_SAMPLE	Experiments and Clinical Trials	Sample name, type, and ID, and associated experiment ID, subject ID, and patient event ID.
BIO_SUBJECT	Experiments and Clinical Trials	Subject ID and type, and site subject ID.

To view a schema diagram of the core data warehouse, see Core Data Warehouse on page 90.

Data Mart for tranSMART Search

Table Name	**Description**
`SEARCH_APP_ACCESS_LOG`	Information about tranSMART search events, including type of search and the user running the search.
`SEARCH_AUTH_USER`	Administrative data about the user running a search, including user email and tranSMART user type (such as Administrator, Study Owner, Study Spectator). Used for authentication.
`SEARCH_AUTH_USER_SEC_ACCESS`	User access ID and object ID.
`SEARCH_BIO_MKR_CORREL_FAST_MV`	Materialized view that associates a domain object ID with the associated biomarker ID and the correlation type.
`SEARCH_CUSTOM_FILTER`	Saved search filter ID, name, and description, and ID of user who created the filter.
`SEARCH_CUSTOM_FILTER_ITEM`	Saved search filter ID and the IDs of one or more components of the search filter.
`SEARCH_GENE_SIG_FILE_SCHEMA`	Description of the format of a gene signature file – for example, *Gene Symbol* <tab> *Metric Indicator*.
`SEARCH_GENE_SIGNATURE`	Complete definition of a gene signature, as defined on the pages of the gene signature wizard.
`SEARCH_GENE_SIGNATURE_ITEM`	Link between gene signature search IDs and biomarker IDs, bio data IDs, and fold change metrics.
`SEARCH_KEYWORD`	Name, ID, bio data ID, category, and other information about the keywords displayed in the autocomplete dropdown.
`SEARCH_KEYWORD_TERM`	Name, ID, rank, length, and other information about the keyword displayed in the autocomplete dropdown.
`SEARCH_REQUEST_MAP`	Link between request map ID, configuration attribute (such as role administrator or anonymous authentication), and target URL. Used for authentication.
`SEARCH_ROLE`	ID and authority description of the tranSMART search roles (administrator, study owner, spectator). Used for authentication.
`SEARCH_ROLE_AUTH_USER`	Link between role IDs and authorities IDs. Used for authentication.

Table Name	Description
SEARCH_SEC_ACCESS_LEVEL	Access level name, value, and ID.
SEARCH_SECURE_OBJECT	Object ID and associated biodata ID and display name (such as a particular clinical trial).
SEARCH_SECURE_OBJECT_PATH	Secure object ID and the object's path in the navigation tree.
SEARCH_USER_FEEDBACK	Log of user feedback about tranSMART, including user ID, date issued, and feedback text.

To view a schema diagram of the tables in the tranSMART Search Data Mart, see Data Mart for tranSMART Search on page 91.

Data Mart for tranSMART Dataset Explorer

The Dataset Explorer data mart is a subset of data from the core data warehouse that is converted into an i2b2 schema for data exploration.

The following tables describe the i2b2-based schemas you need to be aware of.

I2B2METADATA and I2B2DEMODATA Schemas

The i2b2metadata schema contains ontology data. The i2b2demodata schema contains ontology and fact data. The tables and their relationships with i2b2 fields are listed below:

Schema.Table	Description	Relationships
I2B2DEMODATA. CONCEPT_COUNTS	Stores the counts that are displayed at all branch-level nodes in the navigation tree.	CONCEPT_PATH to: I2B2.C_FULLNAME CONCEPT_DIMENSION.CONCEPT_PATH
I2B2DEMODATA. CONCEPT_DIMENSION	Stores all paths. (Similar to i2b2 with small differences.)	CONCEPT_PATH to: CONCEPT_COUNTS.CONCEPT_PATH I2B2.C_FULLNAME CONCEPT_CD to: OBSERVATION_FACT.CONCEPT_CD
I2B2DEMODATA. OBSERVATION_FACT	Stores all measurements recorded for patients during the trial.	CONCEPT_CD to: CONCEPT_DIMENSION.CONCEPT_CD I2B2.C_BASECODE PATIENT_NUM to: PATIENT_DIMENSION.PATIENT_NUM

Schema.Table	**Description**	**Relationships**
I2B2DEMODATA. PATIENT_DIMENSION	Stores patient demographic data.	PATIENT_NUM to: OBSERVATION_FACT.PATIENT_NUM
I2B2DEMODATA. PATIENT_TRIAL	Maps a patient to a trial (contains one record for each trial/patient association).	PATIENT_NUM to: PATIENT_DIMENSION.PATIENT_NUM TRIAL to: wildcard match with CONCEPT_PATH or C_FULLNAME.
I2B2METADATA. I2B2	Stores detailed information about the i2b2 navigation tree.	C_BASECODE to: OBSERVATION_FACT.CONCEPT_CD I2B2.CONCEPT_CD C_FULLNAME to: CONCEPT_DIMNENSION.CONCEPT_PATH I2B2.C_FULLNAME
I2B2METADATA. I2B2_SECURE	Allows security access to be set at the node level of the navigation tree. This is a copy of the i2b2 table with the additional security field SECURE_OBJ_TOKEN.	See relationships for table i2b2.
I2B2METADATA. I2B2_TAGS	Used to map a search Type (Area, Compound, or Disease) to a trial in the Search by Subject tab.	PATH to: I2B2.C_FULLNAME

To view a diagram of the tables in the i2b2metadata and i2b2demodata schemas, see I2B2METADATA and I2B2DEMODATA on page 92.

DEAPP Schema

The DEAPP schema is an extension of i2b2. It includes biomarker data such as RBM, Affymetrix expressions, and SNP, as well as security extensions for clinical trials. The tables in the schema are listed below:

Table Name	**Description**
DE_MRNA_ANNOTATION	Identifiers for mRNA data, such as gene ID, gene symbol, and probe set ID.
DE_SAVED_COMPARISON	Contains the IDs of the subset definitions saved with the Dataset Explorer **Save** button.
DE_SUBJECT_MICROARRAY_DATA	Gene expression data used in heat maps. Includes patient, subject, and assay IDs, and gene symbol, probe set, timepoint, and intensity values.

Table Name	Description
`DE_SUBJECT_PROTEIN_DATA`	Protein data, including patient, subject, assay, and gene IDs, and gene symbol, timepoint, and intensity values.
`DE_SUBJECT_RBM_DATA`	RBM data used in heat maps. Includes patient, assay, and gene IDs, gene symbol and antigen names, and intensity values.
`DE_SUBJECT_SAMPLE_MAPPING`	Mapping table for RBM and microarray data. Each RBM dataset has an observed score and a z-score. Generally, `DE_SUBJECT_SAMPLE_MAPPING` will have two records for every record in `DE_SUBJECT_RBM_DATA` – one for the observed score and one for the z-score.
`DE_XTRIAL_CHILD_MAP`	Information for mapping a concept from a specific trial to a "parent concept" that exists across trials.
`DE_XTRIAL_PARENT_NAMES`	Information about concepts that are mapped across trials.
`DEAPP_ANNOTATION`	Identifiers for data such as RBM data, including annotation name and gene symbol.
`DEAPP_PROBESET`	Identifiers for probe sets, such as probe set ID and GPL platform.
`HAPLOVIEW_DATA`	SNP data for haploviews. Includes i2b2 and JnJ IDs, and trial, gene, and chromosome names.
`SNP_INFO`	SNP data, including SNP ID, gene symbol, chromosome, and SNP reference.

To view a diagram of the tables in the DEAPP schema, see DEAPP on page 93.

Categories and Sources of Data

The following table lists some of the kinds of data you will load into the data warehouse:

Data Category	Description/Source	Initial Destination
Metadata	Data from clinical trials, experiments, analyses, and assay platforms. The Curation Analyst is the primary source of metadata.	Schema `BIOMART_LZ`. `GEO_`... tables (GEO) in the `REFERENCE` schema.

Data Category	**Description/Source**	**Initial Destination**
Analyzed data	Statistical data such as TEA scores, and data used for comparison and correlation purposes. Also includes references to analyzed data in document repositories. Omicsoft is the primary source of analyzed data.	Schema `BIOMART_LZ`. Schema `OMICSOFT_LZ`.
Profile data	Microarray statistics – minimum values, maximum values, mean values, 25th percentile, 75th percentile. Dana Farber is the primary source of mRNA profile data.	Schema `WEBARRAY` for Dana Farber data.
Biomarker data	Attributes of diseases, compounds and biomarkers such as genes and pathways. Also correlations between biomarkers, such as gene/pathway or pathway/disease. Sources include Pictor, Entrez Gene, KEGG, GeneGO, Ingenuity, ResNet, Pathway Studio, and Jubilant.	Schema `PICTOR`: ■ `KEGG_`... tables (KEGG) ■ `GO`... and `GENEGO`... tables (GeneGO) ■ `INGENUITY` table (Ingenuity) ■ `AGI`... tables (Pathway Studio) Schema `PSE62` (ResNet) Jubilant compound data is copied into the `BIO_COMPOUND` and `BIO_DATA_COMPOUND` tables from previously loaded Jubilant data. See `literature_rebuild_compound.sql` in Subversion (`https://svn.recomdata.com/repo1/jnj/trunk/db/etl/literature`).
Reference data	Data used to build the data warehouse. For example, reference data is used to refresh annotation data. Sources include Entrez Gene, GEO, Hydra, and MeSH.	Schema `REFERENCE`: ■ `GEO_`... tables (GEO) ■ `MESH_`... tables (MeSH) ■ `PROBESET2ANNOTATION_AFFY`, `PROBESET2CHIPTYPE`, `CHIP_TYPE` (Hydra) ■ `EXT_ALL_GENE_INFO`, `EXT_HOMO_SAPIEN_GENE_INFO`, `EXT_GENE_HISTORY` (Entrez Gene and Homology gene)

Data Category	**Description/Source**	**Initial Destination**
Curated literature	Data curated from published articles on topics such as oncology and asthma. Jubilant Biosys is the primary source of this data.	Schema `BIOMART_WZ`: ■ `STG_LIT_ALT` ■ `STG_LIT_ALT_AMD` ■ `STG_LIT_INH` ■ `STG_LIT_INT` ■ `STG_LIT_PE` ■ `STG_LIT_SUM`
JnJ documents	Documents (such as conference abstracts and oncology papers) contained in Johnson & Johnson repositories.	See Loading Data for Document Repository Searches on page 44 for the location of the documents and document indexes.
Cell line data	Data required for gene signature functionality. This data is loaded from the table `CELLLINESDICT` in database `CENTCLINRD`. The data is from Johnson & Johnson internal sources and from external sources.	`BIOMART_WZ.BIO_CELL_LINE`

Clinical trial (Dataset Explorer)	Data from a clinical trial.
	The Curation Analyst is the primary source of clinical trial data.
		`I2B2_LZ.STG_CATEGORY`
		`I2B2_LZ.TIME_POINT_MEASUREMENT`
		`I2B2_LZ.PATIENT_INFO`

RBM (Dataset Explorer)	RBM measurements from a clinical trial. The Curation Analyst is the primary source of RBM data.	`DEAPP_WZ.STG_SUBJECT_RBM_DATA`
mRNA Affymetrix (Dataset Explorer)	mRNA Affymetrix measurements from a clinical trial. The Curation Analyst is the primary source of mRNA Affymetrix data.	`DEAPP_WZ.STG_SUBJECT_MRNA_DATA`

Coordination with the Application Developers

Keep in mind that new sets of data you receive – especially new data sets that require changes to a schema – might also require changes to the application UI. For example, if you receive curated literature data for a new disease category, such as diabetes, the application developers will need to modify the UI to accommodate the new data.

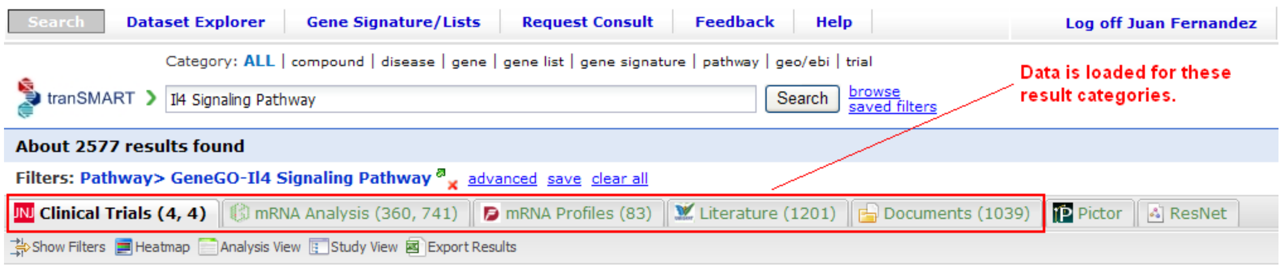
Chapter 3

Loading Data for Searches

This chapter provides information on loading data for tranSMART searches conducted on the Search tab. Search results involving this data are displayed in the following result categories:

- **Clinical Trials** – Internal Johnson & Johnson clinical trials.
- **mRNA Analysis** – Public or internal mRNA experiments.
- **mRNA Profiles** – Gene expression profiles from public cancer data sets curated by the Dana Farber Cancer Institute GCOD database.
- **Literature** – Curated documents within broad subject matter categories such as oncology and asthma.
- **Documents** – Documents from Johnson & Johnson's document repositories.

The following figure shows the result categories that display the data you will load for tranSMART searches:



Loading Data for Clinical Trial, mRNA Analysis, and mRNA Profile Searches

The following table summarizes the kinds of data you must load for the Clinical Trial, mRNA Analysis, and mRNA Profile result categories:

Type of Data	Description	For More Information...
Metadata	Data from clinical trials, experiments, analyses, and assay platforms.	Loading Metadata (page 32)
Analyzed data	Statistical data such as TEA scores, and data used for comparison and correlation purposes. Also includes references to analyzed data in document repositories.	Loading Analyzed Data (page 33)

Type of Data	Description	For More Information...
Profile data	Microarray statistics – minimum values, maximum values, mean values, 25th percentile, 75th percentile.	Loading Profile Data (page 35)
Biomarker data	Compounds, diseases, human and mouse genes, pathway information from sources such as Pictor and KEGG – anything that may be of interest to a research scientist.	Loading Biomarker Data (page 35)
Reference data	Data, such as annotation data, used to build the data warehouse.	Loading Reference Data (page 38)

Loading Metadata

Metadata includes the following kinds of information:

- Attributes of **clinical trials**, such as trial number, study owner, study type, number of patients, number of weeks, dosing regimen, description of control groups, and primary and secondary endpoints.
- Attributes of **experiments**, such as experiment ID, experiment title, experiment description, experiment type, primary investigator, and ETL ID.
- Attributes of **assay platforms**, such as platform name, platform ID, platform description, platform array, organism involved, and vendor.

Sources of Data

Metadata is provided by the Curation Analyst, typically in a Microsoft Excel spreadsheet.

Tasks

The structure of some of the metadata files you receive from the Curation Analyst will differ from study to study. Therefore, there can be no standard set of tables in the landing zone to receive the data in these files. You may need to create or modify some landing zone tables to accommodate this metadata.

Use the SQL scripts in Subversion to help you transform the metadata and load it into the landing zone. The section Tables below lists the data warehouse tables that you will ultimately load the metadata into.

Scripts

Scripts for loading and transforming metadata are found in the branches of the following Subversion directory:

`https://svn.recomdata.com/repo1/jnj/trunk/db`

A list of the specific scripts to use for managing metadata is TBD.

Tables

Metadata is loaded into the following data warehouse tables:

- `BIO_ASSAY_PLATFORM`
- `BIO_CLINICAL_TRIAL`
- `BIO_CLINICAL_TRIAL_ATTR`
- `BIO_CLINC_TRIAL_PT_GROUP`
- `BIO_CLINC_TRIAL_TIME_PT`
- `BIO_CONCEPT_CODE`
- `BIO_DATA_ATTRIBUTE`
- `BIO_DATA_EXT_CODE`
- `BIO_EXPERIMENT`
- `BIO_PATIENT`
- `BIO_PATIENT_EVENT`
- `BIO_PATIENT_EVENT_ATTR`

Loading Analyzed Data

Analyzed data is derived from analyses of clinical trials and experiments. The data includes the following kinds of information:

- **Attributes** of the analysis, such as analysis name, analysis description, analysis method, and assay type.
- **Statistics** from the analysis, such as fold change value, raw and adjusted p-value, and TEA value.
- **Attributes of** a **data set**, such as dataset name and dataset ID.
- Attributes of **associated document references**, such as file name, file location, repository ID, and repository type.

Sources of Data

Sources of analyzed data include:

- Data parsed by Omicsoft and provided in their proprietary format.
- Gene Expression Omnibus (GEO) datasets.
- Internal Johnson & Johnson data files from the Curation Analyst. Data could be stored in a variety of formats – for example, XLS, DOC, RTF, PDF, SAS, Microsoft Access database.

Tasks

Use the SQL scripts in Subversion to help you transform the data and load it into the landing or working zone. The section Tables below lists the data warehouse tables that you will ultimately load the analyzed data into.

In the working zone, perform the following tasks in the following order:

1. Load data from clinical trials and experiments first.
2. Load analysis descriptions.
3. Load the raw analyzed data into the table `BIO_ASY_ANALYSIS_DATA_ALL`.
4. Run TEA score calculations on the data.
5. Filter out rows of data that do not satisfy the following thresholds:
   - TEA scores must be less than 0.
   - p-values must be less than 0.1.
   - Fold change must be greater than absolute 1.0.

   Further, rows of data that contain null values in the above columns should not be copied to the table `BIO_ASSAY_ANALYSIS_DATA`.
6. Load the filtered data into `BIO_ASSAY_ANALYSIS_DATA`.

Scripts

Scripts for loading, transforming, and filtering analyzed data are found in the branches of the following Subversion directory:

`https://svn.recomdata.com/repo1/jnj/trunk/db`

A list of the specific scripts to use for managing analyzed data is TBD.

Tables

Analyzed data is loaded into the following data warehouse tables:

- `BIO_ASSAY_ANALYSIS`
- `BIO_ASSAY_ANALYSIS_DATA`

  This table is loaded from the intermediate table `BIO_ASY_ANALYSIS_DATA_ALL`.
- `BIO_ASSAY_DATASET`
- `BIO_CONTENT_REFERENCE`
- `BIO_CONTENT_REPOSITORY`
- `BIO_CONTENT`
- `BIO_DATA_EXT_CODE`
- `BIO_DATA_LITERATURE`
- `BIO_DATA_OMIC_MARKER`
- `BIO_DATA_UID`

Loading Profile Data

Profile data is derived from experiments. The data includes the following kinds of information:

- **Microarray statistics** from the profile data, such as minimum values, maximum values, mean values, 25th percentile, and 75th percentile.
- **Raw data** from an experiment, such as feature group name, floating point value, experiment ID, assay ID, and assay dataset ID.
- Attributes of a **data set**, such as dataset name and dataset ID.

Sources of Data

Sources of profile data include:

- An Oracle GeneChip oncology database provided by Dana-Farber Cancer Institute (as a web array database).
- Processed microarray data, typically in XLS format.

Tasks

Use the SQL scripts in Subversion to help you transform the data and load it into the landing or working zone. The section Tables below lists the data warehouse tables that you will ultimately load the profile data into.

In the working zone, perform the following tasks in the following order:

1. Load statistical data into the intermediate table `BIO_ASY_DATA_STATS_ALL`.
2. Filter out any rows where the following values are null:
   - Minimum value
   - Maximum value
   - 25th percentile
   - 75th percentile
   - Mean value (that is, 50th percentile)
3. Load the filtered data into `BIO_ASSAY_DATA_STATS`.

Scripts

Scripts for loading, transforming, and filtering profile data are found in the branches of the following Subversion directory:

https://svn.recomdata.com/repo1/jnj/trunk/db

A list of the specific scripts to use for managing profile data is TBD.

Tables

Profile data is loaded into the following data warehouse tables:

- `BIO_ASSAY`
- `BIO_ASSAY_DATA`
- `BIO_ASSAY_DATA_STATS`

  This table is loaded from the intermediate table `BIO_ASY_DATA_STATS_ALL`.
- `BIO_ASSAY_DATASET`
- `BIO_DATA_EXT_CODE`
- `BIO_DATA_OMIC_MARKER`
- `BIO_DATA_UID`

Loading Biomarker Data

Biomarker data relates to anything that may be of interest to a research scientist. The data includes the following kinds of information:

- Attributes of **compounds**, such as Johnson & Johnson ID, code name, brand name, chemical name, mechanism, and category.
- **Disease** names and associated identifiers such as disease IDs, MeSH codes, and ICD 9/10 codes.
- Attributes of **biomarkers** (for example, genes and pathways), such as biomarker name, description, type, and organism.
- Biomarker **correlations**, such as gene/pathway or pathway/disease.

Sources of Data

Sources of biomarker data include:

- Gene and homologene downloads from Entrez Gene.
- Pathway databases such as Pictor, KEGG, GeneGO, and Pathway Studio.
- Small molecule compounds from Jubilant and ResNet.
- MeSH (UMLs).

Tasks

Use the SQL scripts in Subversion to help you transform the biomarker data and load it into the landing or working zone. The section Tables below lists the data warehouse tables that you will ultimately load the biomarker data into.

Scripts

Scripts for loading and transforming biomarker data are found in the branches of the following Subversion directory:

`https://svn.recomdata.com/repo1/jnj/trunk/db`

A list of the specific scripts to use for managing biomarker data is TBD.

Tables

Biomarker data is loaded into the following data warehouse tables:

- `BIO_COMPOUND`
- `BIO_CONCEPT_CODE`
- `BIO_DATA_ATTRIBUTE`
- `BIO_DATA_CORREL_DESC`
- `BIO_DATA_CORRELATION`
- `BIO_DATA_EXT_CODE`
- `BIO_DATA_OMIC_MARKER`
- `BIO_DATA_UID`
- `BIO_DISEASE`
- `BIO_MARKER`

Loading Reference Data

Reference data is any data used to build the data warehouse. For example, reference data is used to refresh annotation data in the data warehouse.

Annotation data must be refreshed regularly. For example, probe sets are annotations for genes. Over time, a probe set mapping to a gene may change as more experiments are done and more data is collected.

Sources of Data

Sources of reference data include:

- Entrez Gene.
- GEO, Hydra (Affymetrix annotations).
- GEO (experiment sample associations).
- MeSH (UMLs).
- ResNet.

Tasks

Use the schema `Reference` as the landing zone for reference data. For example, the following tables contain Affymetrix reference data from Hydra:

- `CHIP_TYPE`
- `PROBESET2ANNOTATION_AFFY`
- `PROBESET2CHIPTYPE`

Map annotations with annotated data through the data warehouse table `BIO_DATA_OMIC_MARKER`. The mapping is established between the columns `BIO_MARKER_ID` (for example, a gene) and `BIO_DATA_ID` (which includes the annotation).

In the following figure, a row from the table `BIO_MARKER` is mapped with a row from the table `BIO_ASSAY_ANALYSIS_DATA`. In the figure, the gene `THBS1` is annotated by the probe set `115_at`:

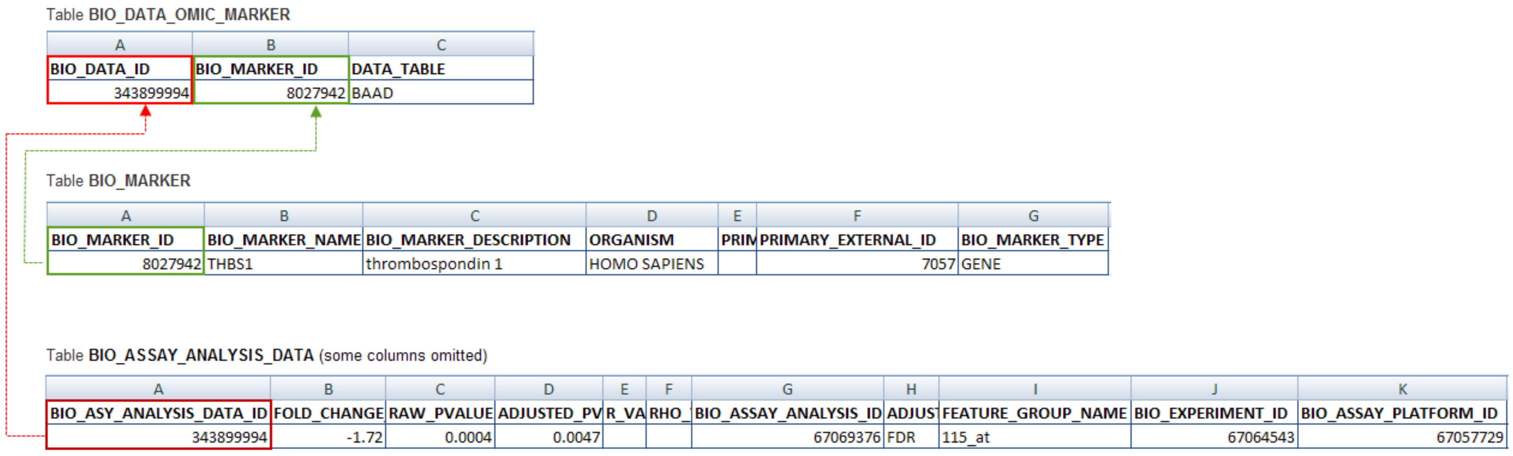


Table BIO_DATA_OMIC_MARKER

A	B	C
BIO_DATA_ID	BIO_MARKER_ID	DATA_TABLE
343899994	8027942	BAAD

Table BIO_MARKER

A	B	C	D	E	F	G
BIO_MARKER_ID	BIO_MARKER_NAME	BIO_MARKER_DESCRIPTION	ORGANISM	PRIM	PRIMARY_EXTERNAL_ID	BIO_MARKER_TYPE
8027942	THBS1	thrombospondin 1	HOMO SAPIENS		7057	GENE

Table BIO_ASSAY_ANALYSIS_DATA (some columns omitted)

A	B	C	D	E	F	G	H	I	J	K
BIO_ASY_ANALYSIS_DATA_ID	FOLD_CHANGE	RAW_PVALUE	ADJUSTED_PV	R_VA	RHO_	BIO_ASSAY_ANALYSIS_ID	ADJUS	FEATURE_GROUP_NAME	BIO_EXPERIMENT_ID	BIO_ASSAY_PLATFORM_ID
343899994	-1.72	0.0004	0.0047			67069376	FDR	115_at	67064543	67057729

Changes to annotation data may require existing data to be updated or replaced. For example:

- Disease and compound information can be updated.
- Pathway information must be deleted and re-loaded.

Scripts

Scripts for loading reference data and refreshing annotations in the data warehouse are found in the branches of the following Subversion directory:

https://svn.recomdata.com/repo1/jnj/trunk/db

A list of the specific scripts to use for managing reference data is TBD.

Tables

Data in the data warehouse is refreshed based on the reference data that you load into the landing zone. But typically, the reference data itself is not carried forward into the data warehouse. However, the scripts that refresh annotations in the data warehouse do use staging tables, including the following:

- `STG_NON_OMICSOFT_BAAD_FT2`
- `STG_NON_OMICSOFT_DATA_BAAD`
- `STG_OMIC_MARKER_BEERSE`
- `STG_OMIC_MARKER_CENTRD`
- `STG_OMIC_MARKER_GPL`
- `STG_OMIC_MARKER_GPL_2`
- `STG_OMIC_MARKER_JUB`
- `STG_OMIC_MARKER_LIT`
- `STG_OMIC_MARKER_RBM`
- `STG_OMIC_MARKER_STATS`
- `STG_OMIC_MARKER_TRIAL`

Loading Data for Curated Literature Searches

The current store of curated literature datasets are in the following development environment databases:

- `CENTCLINRD`. Contains the original set of curated oncology data.
- `JBL_JWB`. Contains supplementary oncology data. This database plus `CENTCLINRD` constitute the entire set of existing oncology data.
- `JBL_ASTHMA`. Contains the entire set of curated asthma data.

Typically the literature curator (such as Jubilant) will give you updates to the existing datasets, but it is possible that you will receive a completely new version of an existing dataset.

When you receive a set of data from the curator, you must index the new data, and also re-index and re-load all the existing data, using the databases listed above as your sources for the existing data.

Workflow for Loading Data from Curated Literature

The process for loading data from curated literature is as follows:

1. When new data is available from the curator, ask the curator for the following:
   - An Oracle database containing the data, and connection details for accessing the database.
   - A schema describing the organization of the data.
   - Sample queries.
2. The literature curator provides you with the new data.
3. Analyze the data for any issues that might cause compatibility problems with the existing data or with the presentation of the data in the tranSMART UI.
4. Build a source-to-target map to ensure that the new data is compatible with the data warehouse schema, making any changes to the schema as needed.

   The following sample maps have been used to map new, raw curated data to the data warehouse schema:
   - Jub_Asthma_S2T.xlsx
   - Jub_Centclinrd_S2T.xlsx
   - Jub_Jwb_S2T.xlsx

   Pay particular attention to the one-to-many relationships.
5. Load the new data into the staging tables in `BIOMART_WZ`.

   The staging tables and their corresponding `BIOMART_WZ` data warehouse tables are as follows:

Staging Table	Corresponding Data Warehouse Table
STG_LIT_ALT	BIO_LIT_ALT_DATA
STG_LIT_ALT_AMD	BIO_LIT_AMD_DATA
STG_LIT_INH	BIO_LIT_INH_DATA
STG_LIT_INT	BIO_LIT_INT_DATA
STG_LIT_PE	BIO_LIT_PE_DATA
STG_LIT_SUM	BIO_LIT_SUM_DATA

6. Reload the existing data from the from CENTCLINRD, JBL_JWB, and JBL_ASTHMA tables into the staging tables.

   For examples of scripts that reload the existing data, see the scripts listed in section Sample Scripts on page 43.

7. Copy the data from the staging tables to the data warehouse in the development environment.

   The tables that contain curated literature data have the prefix BIO_LIT_ ..., and an additional table is named BIO_DATA_LITERATURE. For a brief description of the tables, see the section Core Data Warehouse on page 20.

   See Core Data Warehouse on page 90 for an illustration of the schema that contains the curated literature tables.

   For an example of a script that copies the data to the data warehouse, see lit_copy.sql in the following SVN location:

   https://svn.recomdata.com/repo1/jnj/trunk/db/etl/literature

8. Index the new data and re-index the existing data.

   For examples of scripts that re-index the existing data, see the scripts listed in section Sample Scripts on page 43.

9. Validate the data through SQL queries and in the tranSMART UI.

   For example, the sample scripts include row counts of existing data in the comments, such as the following:

```
select
  trim(target_name),
   ...
  'JBL_JWB.DISEASE_SUMMARY-'
from
  jbl_jwb.disease_summary a; -- 2043
```

   The number 2043 indicates how many records are expected to be retrieved in this particular operation.

10. Copy the data to the corresponding data warehouse tables in the staging/QA environment.

11. Run test scripts against the data.

12. Copy the production-ready data to the corresponding data warehouse tables in the production environment.

Sample Scripts

The following scripts perform functions such as copying data from the various sources of curated literature to the staging tables (`STG_` ...), copying data from the staging tables to the data warehouse, and indexing the data.

The scripts manage existing data in the `CENTCLINRD`, `JBL_JWB`, and `JBL_ASTHMA` tables. When you receive new curated literature data, you must modify the scripts or write new ones to load the new data.

The script names in the table are located in source control:

https://svn.recomdata.com/repo1/jnj/trunk/db/etl/literature

Script	Description
`lit_alt_create.sql`	Copies existing alterations data from `CENTCLINRD`, `JBL_JWB`, and `JBL_ASTHMA` to staging tables in `BIOMART_WZ`. Also processes the data.
`lit_copy.sql`	Copies curated literature data from the staging tables to the `BIOMART` data warehouse in the development environment.
`lit_inh_create.sql`	Copies existing inhibitors data from `CENTCLINRD` and `JBL_JWB` to staging tables in `BIOMART_WZ`. Also processes the data.
`lit_int_create.sql`	Copies existing interactions data from `CENTCLINRD`, `JBL_JWB`, and `JBL_ASTHMA` to staging tables in `BIOMART_WZ`. Also processes the data.
`lit_int_ref_patch.sql`	Patches reference component and gene IDs for interactions.
`lit_pe_create.sql`	Copies existing protein effects data from `JBL_ASTHMA` to staging tables in `BIOMART_WZ`. Also processes the data.
`lit_sum_create.sql`	Copies existing oncology alterations summary data from `CENTCLINRD` and `JBL_JWB` to staging tables in `BIOMART_WZ`. Also processes the data.
`literature_rebuild_compound.sql`	Rebuilds the compound index table.
`literature_rebuild_disease.sql`	Rebuilds the disease index table.
`literature_rebuild_omic_marker.sql`	Rebuilds the omic marker index table for alterations, inhibitors, and interactions.

Loading Data for Document Repository Searches

tranSMART performs searches against the following kinds of documents in Johnson & Johnson document repositories:

- PDF, HTML and other files containing information from conferences.

  Files are stored in a directory named for the associated conference. Conference directories are in the following location on application servers:

  ```
  usr/local/tomcat-5.5.27/appdata/transmart/documents/Conferences
  ```

  For example:

  ```
  usr/local/tomcat-5.5.27/appdata/transmart/documents/Conferences/
  American Academy of Dermatology 2008
  ```

  Copy the files to the appropriate conference directory on the application servers.

- Oncology papers curated by Jubilant.

  Load the files into the following directory on the application servers:

  ```
  usr/local/tomcat-5.5.27/appdata/transmart/documents/Jubilant Oncology
  ```

  Note: Files from conferences and Jubilant oncology documents are opened through links in tranSMART.

- Full text indexing of the Biomarker folder. Requires authorization to access the following shared drive:

  ```
  \\eu.jnj.com\rndbedfsroot\All\DrugDiscovery\biomarker
  ```

- Full text indexing of a DIP folder specific to the cMET project. Requires authorization to access the following shared drive:

  ```
  \\na.jnj.com\jnjdfsroot\eu\rndbe\All\DrugDiscovery\_Discovery Projects\
  ```

Indexing the Documents in the Repositories

The Indexer tool is a Java console application that uses Lucene (http://lucene.apache.org/) to index disk-based document repositories.

Building the Indexer Tool

The Indexer project is an Eclipse project. It is available at `https://svn.recomdata.com/repo1/jnj/trunk/util/indexer`.

To build the Indexer tool:

1. Check out or update the project from Subversion.
2. Import it as an existing project into Eclipse.
3. Build the project.
4. Do an Ant build on `indexer.xml` (right-click this XML file and select **Run as... -> Ant Build**).

   This generates an `indexer.jar` file with all the required libraries.

Usage

Copy the `indexer.jar`, `indexer.bat`, `merger.bat`, and `finder.bat` files to a host with access to the directories you want to index. Make sure Java 1.6 is on the host, and that its `bin` directory is on the `PATH`.

Building Repository Indexes

1. Edit the `indexer.bat` file, then run it to build indexes. You may include one or more commands like the following in the batch file.

   ```
   java -Xms256m -Xmx1024m -cp indexer.jar
      com.recomdata.search.Indexer -create -index "indexes\\Biomarker"
      -repository "Biomarker" -path Z:\
   ```

   This command creates a new index in the `indexes\Biomarker` folder of all the files found under the network share `Z:\` drive. It also sets each indexed document’s repository field to `Biomarker`.

2. Create three files in the `indexes\Biomarker` folder with names similar to the following:

   - `_wwc.cfs`
   - `segments.gen`
   - `segments_tm0`

3. Zip the `Biomarker` file and name it `Biomarker-`*`yyyymmdd`*`.zip`.
4. Copy the zip file to `devapp.jnj.recomdata.com server:/data/indexes`.

If you are indexing new Conference files:

1. Copy the new files to subdirectories with meaningful names under the following devapp server location:

   /usr/local/tomcat-5.5-27/appdata/transmart/Documents/Conferences

2. Build a new Conferences index by indexing that directory.

   ```
   java -Xms256m -Xmx1024m -cp indexer.jar
      com.recomdata.search.Indexer -create -index
      "/data/indexes/Conferences" -repository "Conferences"
      -path "/usr/local/tomcat-5.5-27/appdata/transmart/Documents/Conferences"
   ```

Testing an Index

Edit the `finder.bat` file, then run it to test indexes.

You may include one or more commands like the following in the batch file. This command searches for the term **met** in the Biomarker index:

```
java -cp indexer.jar
    com.recomdata.search.Finder -index "C:\\indexes\\Biomarker" met
```

After you have created and tested new indexes, zip them up and copy them to `devapp.jnj.recomdata.com:/data/indexes`.

Building the Master Index

tranSMART uses one master index as its document index. You built it by merging the individual repository indexes into one master index.

Run this process on the `devapp.jnj.recomdata.com` server.

Edit the `merger.bat` file, then run it to create the master index:

```
mkdir indexes\Master
cp indexes\Biomarkers\* indexes\Master\*
java -Xms256m -Xmx1024m -cp indexer.jar
    com.recomdata.search.Merger -index "indexes\Master" "indexes\Conferences"
java -Xms256m -Xmx1024m -cp indexer.jar
    com.recomdata.search.Merger -index "indexes\Master" "indexes\DIP"
java -Xms256m -Xmx1024m -cp indexer.jar
    com.recomdata.search.Merger -index "indexes\Master" "indexes\Jubilant Oncology"
```

Loading Cell Line Data for Gene Signatures

The following script loads cell line data into the table `BIO_CELL_LINE` (schema `BIOMART_WZ`) for use by the gene signature wizard. The script loads the data from `CELLLINESDICT` (schema `CENTCLINRD`:

```
-- dump all records into biomart
insert into bio_cell_line
( DISEASE, PRIMARY_SITE, METASTATIC_SITE, SPECIES, ATTC_NUMBER,
CELL_LINE_NAME, BIO_DISEASE_ID, ETL_REFERENCE_LINK )
  select
    d.disease DISEASE,
    c.primarysite PRIMARY_SITE,
    c.METASTATICSITE METASTATIC_SITE,
    c.SPECIES SPECIES,
    c.ATTCNO ATTC_NUMBER,
    c.name CELL_LINE_NAME,
    d.bio_disease_id BIO_DISEASE_ID,
    'centclinrd' ETL_REFERENCE_LINK
  from centclinrd.celllinesdict c left join bio_disease d on c.meshid =
    d.mesh_code
  order by name;
-- fix attc null references
update bio_cell_line set ATTC_NUMBER = null where ATTC_NUMBER = 'ATCC:
Not Known';
-- fix attc column
update bio_cell_line c
set attc_number = replace(attc_number,'ATCC: ','')
where attc_number like 'ATCC:%';
commit;
```

Loading a Description of a Clinical Trial

A tranSMART user can open a details box containing information about a clinical trial in either of the following circumstances:

- In a search result, when the user clicks the name of a clinical trial or hovers the cursor over the name of the trial.
- In the Dataset Explorer navigation tree, when the user right-clicks the name of a clinical trial and then clicks **Show Definition**.

The following sample script illustrates how to load information to display in a details box for the clinical trial C1034T06. The script also populates the `I2B2_TAGS` database table with the `AREA`, `DISEASE`, and `COMPOUND` search types so that the trial can be selected in a search:

Clinical_Data_Add_Trial_Description.sql

Creating Synonyms

For the ETL process, synonyms should be created for the following databases in the Control zone:

- i2b2_demodata
- i2b2_metadata
- deapp
- i2b2_lz
- i2b2_wz
- biomart
- biomart_lz

The procedure to create synonyms is:

```
Create_synonyms(<from Database>, <to Database>);
```

Sample script:

```
CODE:
BEGIN
      Create_synonyms('I2B2_DEMODATA', 'CONTROL');
      Create_synonyms('I2B2_METADATA', 'CONTROL');
      Create_synonyms('I2B2_LZ', 'CONTROL');
      Create_synonyms('I2B2_WZ', 'CONTROL');
      Create_synonyms('DEAPP', 'CONTROL');
      Create_synonyms('BIOMART', 'CONTROL');
      Create_synonyms('BIOMART_LZ', 'CONTROL');

END;
```

Chapter 4

Loading Data for Dataset Explorer

To run with full functionality, Dataset Explorer requires key tables and schemas to be properly loaded with data. The following table lists these functions and their required data:

Function	**Required Data**
Basic functionality	`I2B2METADATA` schema ■ Load with ontology data. ■ `C_COMMENT` field of `I2B2` table: For trial nodes, fill this field with "trial:TRIALNAME" where TRIALNAME is the name of the trial. This allows Dataset Explorer to link to the search application to retrieve information about a trial. `I2B2DEMODATA` schema ■ Load with ontology and fact data. ■ Use the `I2B2_CREATE_CONCEPT_COUNTS` stored procedure to populate the `CONCEPT_COUNTS` table. ■ Use the `PATIENT_TRIAL` table to map a subject to trial.
Cross-trial queries	`DEAPP schema` ■ To allow a tranSMART user to query information across trials, the curator needs to provide you with information about how to map concepts in one trial to concepts in another trial. ■ Use the `DE_XTRIAL_PARENT_NAMES` table to hold information about concepts that are mapped across trials. ■ Use the `DE_XTRIAL_CHILD_MAP` table to describe how to map a concept from a specific trial to a “parent concept” that exists across trials.
Trial search	`I2B2METADATA` schema ■ Ensure that table `I2B2_TAGS` is populated with metadata to enable searching by compound, area, or disease (the **Type** dropdown box in the Search by Subject tab). For an example of tagging a clinical trial to enable searching, see Loading a Description of a Clinical Trial on page 47.

Function	Required Data
Heat map and haploview	DEAPP schema ■ DE_PATHWAY and DE_PATHWAY_GENE are required for the autosearch feature in the **Select a Gene/Pathway** field of the Compare Subsets-Pathway Selection dialog. ■ DE_SUBJECT_MICROARRAY_DATA is required for gene expression data in heat maps. ■ DE_SUBJECT_RBM_DATA is required for RBM data in heat maps. ■ HAPLOVIEW_DATA is required for SNP data in haploviews. ■ DE_SUBJECT_SAMPLE_MAPPING is required for the mapping tables for RBM and gene expression data.
Security	I2B2METADATA schema ■ Populate the SECURE_OBJ_TOKEN field in I2B2_SECURE. ■ Populate the SecureObjects with matching tokens (usually EXP:TrialNumber). I2B2DEMODATA schema ■ For each patient, you must place an observation with the corresponding EXP:TrialNumber in the TVAL_CHAR field of the OBSERVATION_FACT table. I2B2METADATA and I2B2DEMODATA schemas ■ Place a concept in I2B2 and CONCEPT_DIMENSION tables matching \\Clinical_Trials\SECURITY\.

Note: To avoid confusion over the meaning of the notation **NA**, please use the following notations consistently:

- Use **NA** to indicate *not applicable*
- Use **Unknown** to indicate *not available*

Naming Conventions

The following naming conventions are used for the names of clinical trials and studies displayed on the Dataset Explorer Navigate Terms tab:

Study Type	**Naming Convention**
Clinical Trial	Johnson & Johnson trial name. Example: C1034T07
Experimental Medicine Study	Descriptive name of study. Example: BRC Depression Study.
Internal Studies	Name segments in the following format: StudySponsor_Disease_Year Example: Veridex_BreastCancer_2003
Public Studies	Name segments in the following format: StudyAuthor_Disease_GEOid Example: Ambs_ProstateCancer_GSE6956

Loading Clinical Trials

The following sections describe information and procedures to help you load data for Clinical Trials.

Raw Data Files

Consolidate the raw data you receive from the Curation Analyst in two files:

- One file for Category data
- One file for Time Point Measurement data

Table Structure for Category Data

Here is the structure of the data in the Category file:

```
CREATE TABLE I2B2_LZ.CATEGORY
(STUDY_ID VARCHAR2(25 BYTE),
CATEGORY_CD VARCHAR2(100 BYTE),
CATEGORY_PATH VARCHAR2(250 BYTE));
```

Note that:

- `CATEGORY_CD` contains the name of the text file after removing the trial number and extension.

  Example: `C0168T32+SAMPLES_AND_TIMEPOINTS+SERUM.txt` becomes `SAMPLES_AND_TIMEPOINTS+SERUM`.

- `CATEGORY_PATH` contains the `CATEGORY_CD` value transformed as a syntactically correct path – that is, replace "+" with "\" and replace "_" with " " (one space).

  Example: `SAMPLES_AND_TIMEPOINTS+SERUM` becomes `SAMPLES AND TIMEPOINTS\SERUM`.

Table Structure for Time Point Measurement Data

Here is the structure of the data in the Time Point Measurement file:

```
CREATE TABLE I2B2_LZ.TIME_POINT_MEASUREMENT
(STUDY_ID VARCHAR2(25 BYTE),
USUBJID VARCHAR2(50 BYTE),
SITE_ID VARCHAR2(10 BYTE),
SUBJECT_ID VARCHAR2(10 BYTE),
VISIT_NAME VARCHAR2(100 BYTE),
DATASET_NAME VARCHAR2(500 BYTE),
SAMPLE_TYPE VARCHAR2(100 BYTE),
DATA_LABEL VARCHAR2(500 BYTE),
DATA_VALUE VARCHAR2(500 BYTE),
CATEGORY_CD VARCHAR2(100 BYTE),
PERIOD VARCHAR2(100 BYTE));
```

Note that:

- When pivoting data (restructuring the data to prepare it for manipulation by your SQL transformation scripts), if a value is moved to its own column (such as `STUDY_ID`, `SUBJECT_ID`, or `PERIOD`) it should not be included in the `DATA_LABEL` and `DATA_VALUE` fields.
- Every category in the Category table should have related records in the `TIME_POINT_MEASUREMENT` table. This should be validated automatically.
- Records with `null` or a single period '.' in the `DATA_VALUE` field should be removed.

- Records must be distinct based on the following fields:
  - `USUBJID`, `STUDY_ID`, `CATEGORY_CD`, `VISIT_NAME`, `SAMPLE_TYPE`, `PERIOD`, `DATA_LABEL`.
  - Any category with duplicates should be sent back to the Curation Analyst for review and clarification, as duplicate data cannot be loaded into i2b2.

Workflow Summary

The following steps outline a typical workflow for loading and transforming clinical data:

1. The Johnson & Johnson curation team provides you, the ETL Analyst, with comma-delimited text files containing study data.
2. Place the study's text files in a folder, along with the following files (located in `/transmart/PDF/ExternalLinks`):
   - **run_TrialDataPivoter.bat**. Runs **TrialDataPivoter.jar**, which extracts, pivots, and loads the data into the following tables:
     - `I2B2_LZ_TIME_POINT_MEASUREMENT`
     - `I2B2_LZ_STG_CATEGORY`
   - TrialDataPivoter_columns.txt. A mapping file that TrialDataPivoter.jar uses to find the data to extract from the study's text files provided by the curator.

   Note: The batch file, mapping file, and the study's text files can be in different directories. See the batch file for information about the arguments it takes.

3. Modify the following script for your study and then run the script. The referenced stored procedure, `I2B2_PROCESS_RAW_DATA`, is in `Control`.

```
DECLARE
   studyID VARCHAR2(100);
   studyType VARCHAR2(100);
BEGIN

-- Set the Trial Number and the Root node the trial will be under in
-- the Dataset Explorer.
studyID := 'C0168T54';
studyType := 'Clinical Trials';

-- Takes files from the staging table, backs them up and moves them
-- into the Work Zone.
I2B2_PROCESS_RAW_DATA_EXTRNL(studyID);

END;
```

4. Modify the following script for your study and then run the script:

   Clinical_Data_PreProcess_WZ.sql

   This script checks the data that was just loaded into the `I2B2_WZ.TIME_POINT_MEASUREMENT` and `I2B2_WZ.CATEGORY` tables for errors – for example, it checks for and deletes records with duplicate key fields (category_cd, usubjid, visit_name, data_label, period).

5. Modify the following script for your study and then run the script:

   Clinical_Data_Load_Script.sql

   This script contains the commands to back up the tables, to apply the curation rules identified in Step 4, to load the lowest level nodes in I2B2, and finally to complete the loading of the observation_fact, concept_counts, and security data.

   Note that the script has an intermediate end point, after it creates the nodes in the navigation tree. This gives you a chance to review the paths in the tree, make any corrections, and optionally to check any issues with the Curation Analyst, before continuing to use the script to load the data.

6. Load the following three `I2B2_WZ` tables – `TIME_POINT_MEASUREMENT`, `CATEGORY`, and `PATIENT_INFO` – into the i2b2 schemas `I2B2METADATA` and `I2B2DEMODATA`.

   There are eight tables in two categories in these schemas:

   - ☐ Main tables for i2b2:
     - `I2B2METADATA.I2B2`
     - `I2B2DEMODATA.CONCEPT_DIMENSION`
     - `I2B2DEMODATA.PATIENT_DIMENSION`
     - `I2B2DEMODATA.OBSERVATION_FACT`
   - ☐ Secondary tables
     - `I2B2METADATA.I2B2_SECURE`
     - `I2B2METADATA.I2B2_TAGS`
     - `I2B2DEMODATA.CONCEPT_COUNTS`
     - `I2B2DEMODATA.PATIENT_TRIAL`

   The file DatasetExplorer.ddl defines the structure of the above tables. For more information about these tables, see the section Data Mart for tranSMART Dataset Explorer on page 26.

7. Load the trial metadata that is displayed in a details box when the user right-clicks on the trial name. Also tag the data by type (`AREA`, `COMPOUND`, `DISEASE`, `WORKFLOW`) so that it can be found in a Dataset Explorer search.

   For more information, including a link to a sample script, see Loading a Description of a Clinical Trial on page 47.

8. Once the load is complete, perform a manual review of the Dataset Explorer UI to ensure that the data is presented properly. For example, check that:
   - ☐ All categories exist.
   - ☐ Counts exist for all branch-level nodes.
   - ☐ Branch- and leaf-level nodes can be dragged into the subset boxes of the Comparison tab.
   - ☐ Statistics can be generated when the **Generate Summary Statistics** button is clicked, and the statistics appear correct.
   - ☐ Nodes do not have duplicate folders, misspellings, or other errors.
   - ☐ A heat map can be generated if applicable.
   - ☐ The trial can be successfully retrieved when the trial name is entered in the **Search** field of the Search by Subject tab.
   - ☐ Right-clicking on the trial name in the navigation tree displays the definition of the trial.
   - ☐ A user with view access to the trial can open and navigate through the trial.
9. Create a Jira ticket listing the corrections that need to be made, and transfer the ticket to the Curation Analyst.
10. Repeat the steps necessary to load the data corrected by the Curation Analyst.

SQL for Loading Clinical Trials

The following file runs stored procedures, located in the `CONTROL` database, for loading clinical trials:

StoredProceduresForTrialLoads.sql

The following table is an alphabetized list of these and other stored procedures used for loading and managing clinical trials:

Stored Procedure	**Description**
`I2B2_ADD_NODE`	Creates the root node. Parameters: Trial ID, Full Path, Path Name.
`I2B2_APPLY_CURATION_RULES`	Applies curation rules (hard-coded and from the `NODE_CURATION` table) to the time point measurement table.
`I2B2_BACKOUT_TRIAL`	Backs out of a trial. **Note:** Use with caution. This procedure deletes all i2b2-related and patient information.

Stored Procedure	**Description**
`I2B2_COPY_WZ_TO_CURATED`	Copies curated data to `time_point_measure_curated`, `category`, and `patient_info` in `i2b2_wz`.
`I2B2_CREATE_CONCEPT_COUNTS`	Loads the `CONCEPT_COUNTS` table. The UI uses this table to display node counts. This procedures also hides any node with a zero count.
`I2B2_CREATE_PATIENT_DIM`	Loads patients into the `PATIENT_DIMENSION` table from the `PATIENT_INFO` table.
`I2B2_CREATE_PATIENT_TRIAL`	Inserts records into the `PATIENT_TRIAL` table.
`I2B2_CREATE_SECURITY_FOR_TRIAL`	Loads a record into the `OBSERVATION_FACT` table for each patient in the trial. This procedures is used for security.
`I2B2_DELETE_1_NODE`	Deletes one node. Does not delete any of the node’s children.
`I2B2_DELETE_ALL_NODES`	Deletes the existing tree and any data under it.

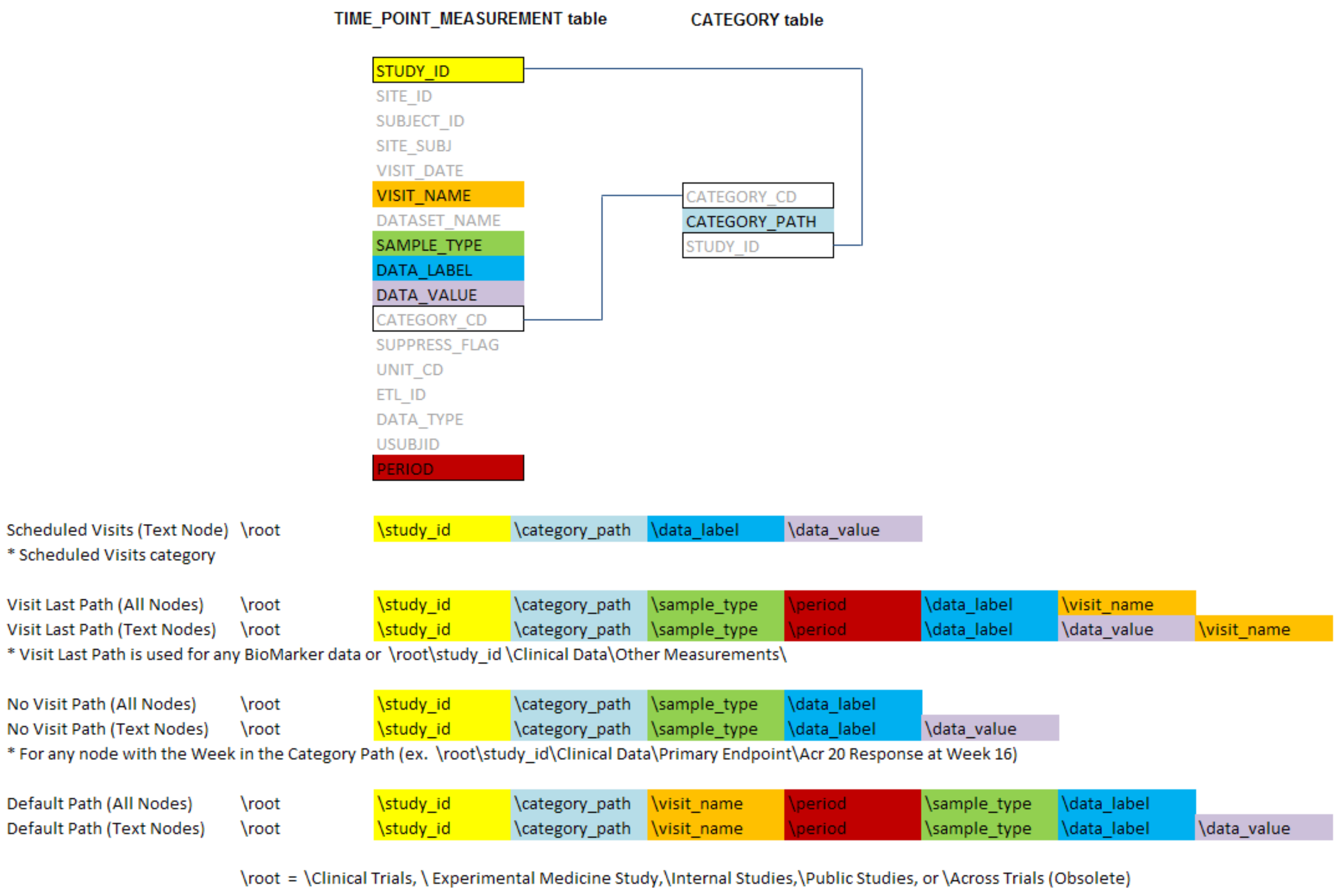
`I2B2_FILL_IN_TREE`	Fills in the tree. This builds all the sub-paths for a trial.
	If a path is created from the `I2B2_APPLY_CURATION_RULES` procedure, `I2B2_FILL_IN_TREE` ensures that all the sub-paths exist.
	For example, if the following path is built:
	`\Clinical Trials\T32\Clinical Data\Other Measurements\Week 1\Blood Pressure{color}`
	`I2B2_FILL_IN_TREE` ensures that sub-paths such as the following exist:
	`\Clinical Trials\T32\Clinical Data\`
	`\Clinical Trials\T32\Clinical Data\Other Measurements\`

Stored Procedure	**Description**
`I2B2_HIDE_NODE`	Hides rather than deletes a node in the i2b2 tree. Updates `I2B2.C_VISUAL_ATTRIBUTES` to `FH` or `LH` (Folder Hidden or Leaf Hidden). For an example of "unhiding" a node, see the following file: SampleUnhideTrial.sql
`I2B2_LOAD_RBM_DATA`	Loads the `DE_SUBJECT_SAMPLE_MAPPING` table and the i2b2 tree with RBM data.
`I2B2_LOAD_SECURITY_DATA`	Loads a record for into the `I2B2METADATA.I2B2_SECURE` table for each tree node. Rebuilds the table for the entire i2b2 instance.
`I2B2_LOAD_TPM_POST_CURATION`	Loads i2b2 tables with all data from the time point measurement table.
`I2B2_MOVE_NODE`	Moves a tree node and all its children under a new path. **Note:** Do not use this procedure to rename a node, as it does not update the `i2b2.c_name` field.
`I2B2_MRNA_ZSCORE_CALC`	Calculates Z-score for mRNA data.
`I2B2_PROCESS_MRNA_DATA`	Loads the mRNA data from the `stg_` tables, calls stored procedure `I2B2_MRNA_ZSCORE_CALC` to calculate z-score, and inserts appropriate records into `I2B2DEMODATA.OBSERVATION_FACT`, `I2B2DEMODATA.CONCEPT_DIMENSION`, and `I2B2METADATA.I2B2`. It also calls the stored procedures to create concept counts and load security.
`I2B2_PROCESS_RAW_DATA_EXTERNL`	Takes files from the external tables, backs them up, and moves them into the working zone.

Stored Procedure	**Description**
`I2B2_PROCESS_RBM_DATA`	Loads the RBM data from the `stg_` tables, calls stored procedure `I2B2_RBM_ZSCORE_CALC` to calculate z-score, and calls `I2B2_LOAD_RBM_DATA` to insert records into the `DE_SUBJECT_SAMPLE_MAPPING` table and the `I2B2` tree.
`I2B2_RBM_ZSCORE_CALC`	Calculates Z-score for RBM data.
`I2B2_RENAME_NODE`	Renames a node. This procedure searches the tree for any node in the specified trial with this name, and replaces it with the new name.
`I2B2_SHOW_NODE`	Shows a node. Updates `I2B2.C_VISUAL_ATTRIBUTES` to `FH` or `LH` (Folder Hidden or Leaf Hidden) Updates i2b2.c_visual_attributes to FA or LA, depending on which type it was previously. For an example of "unhiding" a node, see the following file: SampleUnhideTrial.sql
`I2B2_TABLE_BKP`	Backs up the existing i2b2 tables (`I2B2`, `CONCEPT_DIMENSION`, `OBSERVATION_FACT`, `PATIENT_DIMENSION`, `CONCEPT_COUNTS`).
`I2B2_TABLE_DEFRAG`	Moves the five basic i2b2 tables (`I2B2`, `CONCEPT_DIMENSION`, `OBSERVATION_FACT`, `PATIENT_DIMENSION`, `CONCEPT_COUNTS`). In Oracle, this action defragments the tables. Due to all the inserts and deletes, performance degrades very quickly.

Navigation Tree Structure

The following figure shows the fields in the `TIME_POINT_MESUREMENT` and `CATEGORY` tables and their position in the structure of the navigation tree. The colors of the fields are intended to help you quickly locate each field's position in the tree:


TIME_POINT_MEASUREMENT table
CATEGORY table
STUDY_ID
SITE_ID
SUBJECT_ID
SITE_SUBJ
VISIT_DATE
VISIT_NAME
DATASET_NAME
SAMPLE_TYPE
DATA_LABEL
DATA_VALUE
CATEGORY_CD
SUPPRESS_FLAG
UNIT_CD
ETL_ID
DATA_TYPE
USUBJID
PERIOD
CATEGORY_CD
CATEGORY_PATH
STUDY_ID
Scheduled Visits (Text Node) \root \study_id \category_path \data_label \data_value
* Scheduled Visits category
Visit Last Path (All Nodes) \root \study_id \category_path \sample_type \period \data_label \visit_name
Visit Last Path (Text Nodes) \root \study_id \category_path \sample_type \period \data_label \data_value \visit_name
* Visit Last Path is used for any BioMarker data or \root\study_id \Clinical Data\Other Measurements\
No Visit Path (All Nodes) \root \study_id \category_path \sample_type \data_label
No Visit Path (Text Nodes) \root \study_id \category_path \sample_type \data_label \data_value
* For any node with the Week in the Category Path (ex. \root\study_id\Clinical Data\Primary Endpoint\Acr 20 Response at Week 16)
Default Path (All Nodes) \root \study_id \category_path \visit_name \period \sample_type \data_label
Default Path (Text Nodes) \root \study_id \category_path \visit_name \period \sample_type \data_label \data_value
\root = \Clinical Trials, \ Experimental Medicine Study,\Internal Studies,\Public Studies, or \Across Trials (Obsolete)

Rules for Editing the Navigation Tree

Use the following table as a guide for correcting the nodes in the navigation tree. The Field Name column contains the names of fields in the `NODE_CURATION` table.

Field Name	Operation	Rule
`VISIT_NAME`	HIDE	Do not process any record containing: 'UEV%'
`VISIT_NAME`	HIDE	Do not process any record containing: 'UNSCHED%'
`VISIT_NAME`	HIDE	Do not process any record containing: 'UV%'

Field Name	Operation	Rule
`VISIT_NAME`	HIDE	Do not process any record containing: 'VISIT%'
`DATA_LABEL`	HIDE	Do not process any record containing: 'SUBJID'
`DATA_LABEL`	HIDE	Do not process any record containing: 'VISIT NAME (VISIT)'
`DATA_LABEL`	HIDE	Do not process any record containing: 'PERIOD'
`DATA_LABEL`	HIDE	Do not process any record containing: 'SAMPLE_TYPE'
`DATA_LABEL`	HIDE	Do not process any record containing: 'UNIQUE SUBJECT ID (USUBJID)'
`DATA_LABEL`	HIDE	Do not process any record containing: 'STUDY VISIT (VISIT)'
`DATA_LABEL`	HIDE	Do not process any record containing: 'PERIOD (PERIOD)'
`DATA_LABEL`	HIDE	Do not process any record containing: 'STUDYID'
`DATA_LABEL`	HIDE	Do not process any record containing: 'SITEID'
`DATA_LABEL`	HIDE	Do not process any record containing: 'SITE-SUBJECT ID'
`DATA_LABEL`	HIDE	Do not process any record containing: '%COUNTRY%'
`DATA_LABEL`	HIDE	Do not process any record containing: '%SEQUENCE%'
`DATA_LABEL`	HIDE	Do not process any record containing: 'VISIT NAME'
`DATA_LABEL`	HIDE	Do not process any record containing: 'VISIT NUMBER%'
`DATA_LABEL`	HIDE	Do not process any record containing: '%DEMOGRAPHIC%'
`DATA_LABEL`	HIDE	Do not process any record containing: 'SITE ID'
`DATA_VALUE`	HIDE	Do not process any record containing: 'UNSCHED%'
`DATA_VALUE`	HIDE	Do not process any record containing: 'UV%'
`DATA_VALUE`	HIDE	Do not process any record containing: 'VISIT%'
`DATA_VALUE`	HIDE	Do not process any record containing: '.'
`DATA_VALUE`	HIDE	Do not process any record containing: 'UEV%'
`DATA_LABEL`	CHANGE	Replace ACTUAL TREATMENT GROUP With Treatment Groups
`DATA_LABEL`	CHANGE	Replace ACTUAL TREATMENT GROUP (TRTGRPA) With Treatment Groups
`DATA_LABEL`	CHANGE	Replace AGE IN YEARS (AGE) With Age

Field Name	Operation	Rule
DATA_LABEL	CHANGE	Replace BODY MASS INDEX (BMI) (BMI) With Body Mass Index (BMI)
DATA_LABEL	CHANGE	Replace Baseline Weight (B_WGT) With Weight in KG
DATA_LABEL	CHANGE	Replace Baseline weight (kg) (WEIGHT) With Weight in KG
DATA_LABEL	CHANGE	Replace Height (cm) (HEIGHT) With Height in CM
DATA_LABEL	CHANGE	Replace Height in Centimeters (HEIGHT) With Height in CM
DATA_LABEL	CHANGE	Replace Height | cm With Height in CM
DATA_LABEL	CHANGE	Replace RACE (RACE) With Race
DATA_LABEL	CHANGE	Replace SCHEDULED VISITS With Scheduled Visits
DATA_LABEL	CHANGE	Replace SEX (SEX) With Sex
DATA_LABEL	CHANGE	Replace Weight (kg) (WEIGHT) With Weight in KG
DATA_LABEL	CHANGE	Replace Weight in Kilograms (WEIGHT) With Weight in KG
DATA_LABEL	CHANGE	Replace Weight | kg With Weight in KG
DATA_VALUE	CHANGE	Replace BASELINE With Baseline
DATA_VALUE	CHANGE	Replace EARLYWD With Early Withdrawal
DATA_VALUE	CHANGE	Replace N With No
DATA_VALUE	CHANGE	Replace NEG With Negative
DATA_VALUE	CHANGE	Replace NO With No
DATA_VALUE	CHANGE	Replace POS With Positive
DATA_VALUE	CHANGE	Replace SCREEN With Screening
DATA_VALUE	CHANGE	Replace SCREENING With Screening
DATA_VALUE	CHANGE	Replace Y With Yes
DATA_VALUE	CHANGE	Replace YES With Yes
PERIOD	CHANGE	Replace PRE With Pre-Study Agent
VISIT_NAME	CHANGE	Replace BASELINE With Baseline
VISIT_NAME	CHANGE	Replace EARLYWD With Early Withdrawal

Field Name	Operation	Rule
VISIT_NAME	CHANGE	Replace SCREEN With Screening
VISIT_NAME	CHANGE	Replace STUDY With Study
DATA_VALUE	CHANGE	Replace any value for a Time Period (WEEK, W, CYCLE, DAY) to a standard format with 3 digits for sorting. WEEK should be Week. CYCLE, Cycle, etc.
VISIT_NAME	CHANGE	Replace any value for a Time Period (WEEK, W, CYCLE, DAY) to a standard format with 3 digits for sorting. WEEK should be Week. CYCLE, Cycle, etc.

The following table contains rules you can enforce on fields in the `CATEGORY` and `TIME_POINT_MEASUREMENT` tables, using the stored procedure `I2B2_APPLY_CURATION_RULES`.

Field Name	Operation	Rule
DATA_VALUE	REMOVE	Remove any '|' (PIPE) character from the beginning or end of the value
DATA_VALUE	CHANGE	Replace any '|' (PIPE) character from within the string with a '-' (DASH). Do not replace ones from the beginning or end of the field. (See rule DV1)
DATA_VALUE	REMOVE	Remove the following Parentheses'()' combinations. Left Paren alone. Right Paren Alone. Empty paren pair '()' '( )'
UNIT_CODE	ADD	Parse Unit Code information from the Data_label and put into the unit_Cd field. The Unit info is at the end of the value, separated by a Pipe. Example: 1234 | M/L. The M/L would be entered into the unit_cd field.
DATA_LABEL	CHANGE	Replace any '|' (PIPE) character from within the string with a '-' (DASH). This MUST be performed AFTER UC1.
CATEGORY_PATH	REMOVE	Set category_path = '' where category_cd = 'SCHEDULED_VISITS'
CATEGORY_PATH	REMOVE	Set category_path = '' where category_cd = 'TREATMENT_GROUPS'
CATEGORY_PATH	REMOVE	General rule for `Subjects\Demographics`: All node levels below `Subjects\Demographics` must be removed to avoid duplicate nodes.

Loading Public Studies and Internal Studies

Note: For information on loading RBM and mRNA data for a study, see Loading RBM Data on page 65, and Loading mRNA Affymetrix Data on page 70.

1. Download the study from eRoom and unzip the Curated Dataset.
2. Determine the name that will be used as the study_id during loading:

   For public studies, use the GSE number if available, otherwise create a short (20 characters max, no spaces) name that will uniquely identify the study. This short, temporary name will be used throughout the loading process for both study and biomarker data.

   The short name will be easier to manage when your are working with the data. After all the data has been loaded and the curator approves the study, update the name in the Dataset Explorer to the format shown in Naming Conventions on page 51.
3. Open the study_file_name_correlation.xls and study_tree.xls spreadsheets. They will show how the curator wants the nodes to appear in the Dataset Explorer. The category_cd and category_path values will be derived from this data.
4. Open the study_template.xlsx file and save it as a new spreadsheet with the name that will be used as the study id.
5. Open the new spreadsheet and select the category worksheet tab.
6. Populate the category_cd column with the information in the "Category_Code" column of the study_file_name_correlation.xls spreadsheet. Create the category_path by replacing the "+" in the category_cd with a backslash () and replacing the "_" in the category_cd with a space. Save the spreadsheet.
7. Select the tpm (time point management) worksheet tab in the spreadsheet. The required columns are highlighted in yellow.
8. Using the files in the curated dataset folder:
   a. Open the clinical text file in Excel.
   b. Populate the tpm columns in the next empty row.
      - Study_ID: enter the name for the Study_ID.
      - USUBJID: =concatenate(Study_ID, Subject_ID) first column has the formula already entered.
      - Subject_ID: select all the Subject_ID's in the clinical data, copy and paste into the Subject_ID column.

  - data_label: evaluate the column headers in the clinical data to determine what value should be entered in data_label. The process for dealing with multiple column headers that are potential data_labels is discussed later.
  - category_cd: copy the appropriate category_cd from the category worksheet tab.

c. Select the Study_ID and USUBJID values that were just entered, copy and paste for all Subject_IDs just entered.

d. Select the data_label and category_cd values that were just entered, copy and paste for all Subject_IDs just entered.

e. Populate data_value column in the tmp worksheet.

f. Select all the data values from the clinical data and paste into the tpm worksheet starting with the first empty value.

g. Save the worksheet.

h. Repeat the above steps for each clinical data file.

Note: Sometimes the data in the clinical file can represent more than one type of data. There will be multiple column headers that are to be used as the data_label. The data must be pivoted before it can be copied into the tpm worksheet. To pivot the data, edit **run_JnJFilePivot.bat** (located in `/transmart/PDF/ExternalLinks` with **JnJFilePivot.jar**) and update the pivotstart, columncount, inputfile, and outputfile parameters for the clinical data file that needs to be pivoted. Be sure to check the delimiter character in the clinical file and include the delimiter parameter if necessary. When the file has been pivoted, open the pivoted file in excel and populate the tpm worksheet columns as above. In this case the data_label value is found on each individual row.

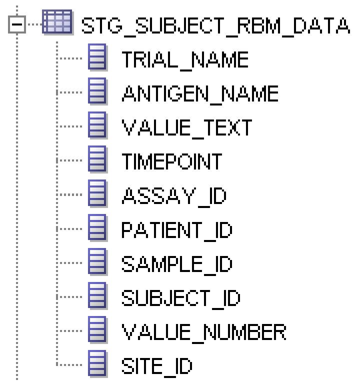
9. When all the clinical data has been loaded into the spreadsheet, save both the category and tmp worksheets as tab-delimited files. These are the category and time_point_measurement files that will be used to load the study. Copy these files to the development server as you would files from a clinical trial.

10. To load the study, follow the same process as in loading a clinical trial (see Loading Clinical Trials on page 51).

Loading RBM Data

RBM data is provided to you by the Curation Analyst. Typically, you will load RBM data after you load the associated clinical trial data.

Load RBM data into `DEAPP_WZ.STG_SUBJECT_RBM_DATA` in the development environment. As you perform the steps to load and transform the raw RBM data, it may help to be aware of table's columns, as shown in the figure below:



Workflow Summary

The following steps outline a typical workflow for loading and transforming RBM data:

1. Download the data files (typically Excel spreadsheets) from the eRoom and unzip any zipped files.

   You should have the following raw data files:

   - A file containing the RBM data.
   - A file containing information about the subjects and samples in the study (subject/sample mapping file).

2. Make sure that the trial still has the "study id" in the node name path for the i2b2 and concept_dimension tables. If not, use stored procedure `i2b2_rename_node` to rename the node back to its original short name.

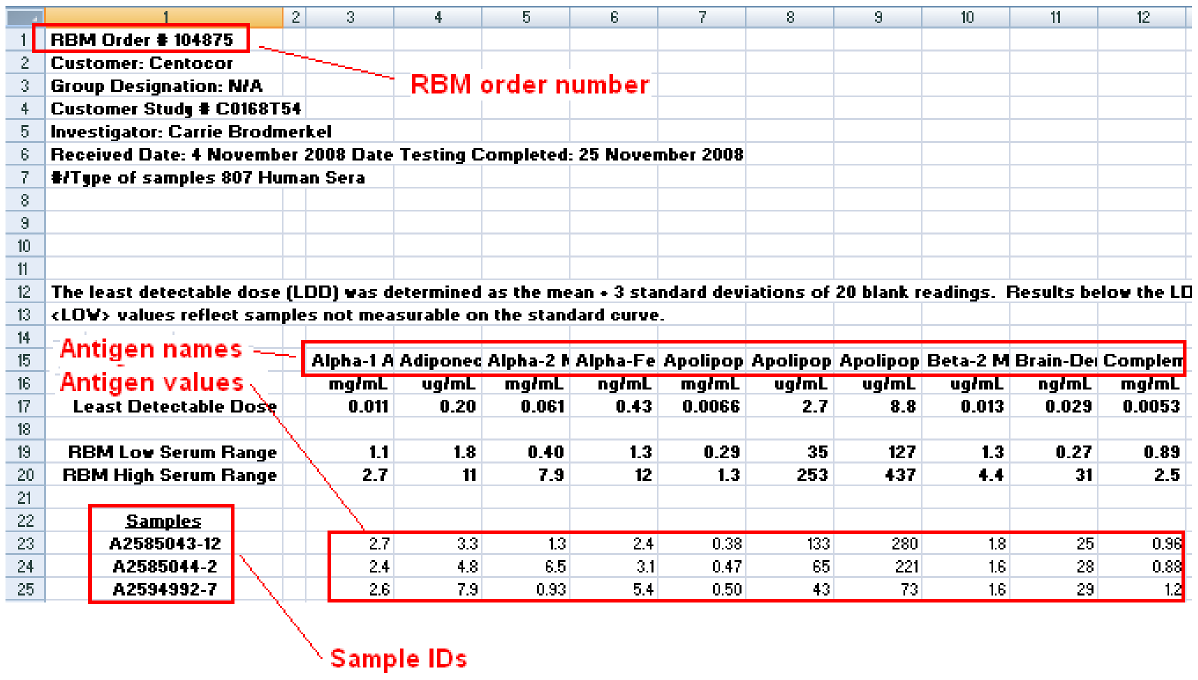
3. Manual file cleanup – RBM data spreadsheet.

   Create a copy of the spreadsheet and work with the copy. If the spreadsheet has more than one worksheet, ask the curator which worksheet to use.

You need the following data from this spreadsheet file:

Spreadsheet Data	Purpose
RBM Order #	Assign this number to the `ASSAY_ID` column of `STG_SUBJECT_RBM_DATA`. **Note:** Different sets of samples may be associated with different RBM order numbers. If so, use the order number associated with a particular set of samples as the `ASSAY_ID` for the related samples.
Antigen names	Assign these names to the `ANTIGEN_NAME` column of `STG_SUBJECT_RBM_DATA`.
Antigen values	Assign these values to the `VALUE_TEXT` column of `STG_SUBJECT_RBM_DATA`. You will later convert these text values to numeric values.
Sample IDs	Use the values in the Samples column of the RBM data spreadsheet to map to the data in the clinical trial spreadsheet.

The following figure shows the spreadsheet data you need:



	1	2	3	4	5	6	7	8	9	10	11	12
1	RBM Order # 104875											
2	Customer: Centocor											
3	Group Designation: N/A											
4	Customer Study # C0168T54											
5	Investigator: Carrie Brodmerkel											
6	Received Date: 4 November 2008 Date Testing Completed: 25 November 2008											
7	#/Type of samples 807 Human Sera											
8												
9												
10												
11												
12	The least detectable dose (LDD) was determined as the mean + 3 standard deviations of 20 blank readings. Results below the LD											
13	<LOW> values reflect samples not measurable on the standard curve.											
14												
15			Alpha-1 A	Adiponec	Alpha-2 M	Alpha-Fe	Apolipop	Apolipop	Apolipop	Beta-2 M	Brain-De	Complem
16			mg/mL	ug/mL	mg/mL	ng/mL	mg/mL	ug/mL	ug/mL	ug/mL	ng/mL	mg/mL
17	Least Detectable Dose		0.011	0.20	0.061	0.43	0.0066	2.7	8.8	0.013	0.029	0.0053
18												
19	RBM Low Serum Range		1.1	1.8	0.40	1.3	0.29	35	127	1.3	0.27	0.89
20	RBM High Serum Range		2.7	11	7.9	12	1.3	253	437	4.4	31	2.5
21												
22	Samples											
23	A2585043-12		2.7	3.3	1.3	2.4	0.38	133	280	1.8	25	0.96
24	A2585044-2		2.4	4.8	6.5	3.1	0.47	65	221	1.6	28	0.88
25	A2594992-7		2.6	7.9	0.93	5.4	0.50	43	73	1.6	29	1.2

Additionally, do the following in the spreadsheet file:

- ☐ Delete all rows above the antigen names.
- ☐ Delete all rows below the antigen names and above the start of the data.
- ☐ Scroll right, locate the last antigen name (usually von Willebrand Factor), and delete all columns to the right of the last antigen name that contain data.
- ☐ If there is a column for "Visit Name" or other timepoint-type columns, identify the timepoint (visit_name) in the column, and determine if you need to extract the timepoint from the column and if an entry needs to be made in the node_curation table.
- ☐ Save the spreadsheet.
- ☐ Generate the data in the worksheet to a tab-delimited text file – for example:

```
Sample_ID   Subject     Antigen     Value Assay_ID
A1   Day 0   1     Alpha-1 Antitrypsin     2.08  SJ_092007
A1   Day 0   1     Adiponectin 3.69  SJ_092007
A1   Day 0   1     Alpha-2 Macroglobulin   0.724 SJ_092007
A1   Day 0   1     Alpha-Fetoprotein 0.77  SJ_092007
A1   Day 0   1     Apolipoprotein A1 0.436 SJ_092007
A1   Day 0   1     Apolipoprotein CIII     47.4  SJ_092007
A1   Day 0   1     Apolipoprotein H  146   SJ_092007
A1   Day 0   1     Beta-2 Microglobulin    1.41  SJ_092007
```

- ☐ Close Excel without saving.

4. Pivot the RBM data in the tab-delimited file using the **run_JnJFilePivot.bat** file (located in `/transmart/PDF/ExternalLinks` with **JnJFilePivot.jar**). This creates a unique record for each sample, subject, and antigen.
5. Zip up the output from the above step (due to the size of the output).
6. Determine if a subject (patient)/sample mapping file exists. There may be enough data in the RBM file to successfully map a subject/site to a patient. If not, see if the clinical data for the study has enough information to manually create a subject/sample mapping file . If not, contact the curator.
7. Manual file cleanup – subject/sample data.

   You need the following data from this spreadsheet file:

Spreadsheet Data	Purpose
Study	Assign the study name to the `TRIAL_NAME` column of `STG_SUBJECT_RBM_DATA`.
Aliquot #	Use the values in the Aliquot # column of the clinical trial spreadsheet to map to the data in the RBM data spreadsheet.

Spreadsheet Data	Purpose
Site	Used with data in the Patient and Study columns to map to the patient data in the I2B2DEMODATA.PATIENT_DIMENSION table.
Patient	Used with data in the Site and Study columns to map to the patient data in the I2B2DEMODATA.PATIENT_DIMENSION table.
Timept	Assign the timepoint values to the TIMEPOINT column of STG_SUBJECT_RBM_DATA.

8. Upload the pivoted RBM zip file and the subject/sample mapping file to the development server.
9. Unzip the RBM data file and move all files to /u01/biomart/biomart_lz.
10. Create external Oracle tables for the RBM data file and the subject/sample mapping file (if one exists) based on the columns found in the data.

    The following DDL sample code illustrates the creation of an external table named BIOMART_LZ.SEBASTIAN_JOHNSTON_RBM, based on the RBM data file only. The source data is in a tab-separated text file named sj_092007_rbm_p.txt, which contains the data extracted from the RBM data spreadsheet.

```
CREATE TABLE "BIOMART_LZ"."SEBASTIAN_JOHNSTON_RBM"
(  "SAMPLE_ID" NVARCHAR2(50),
"SUBJECT" NVARCHAR2(50),
"ANTIGEN" NVARCHAR2(200),
"VALUE" NVARCHAR2(200),
"ASSAY_ID" NVARCHAR2(200)
)
ORGANIZATION EXTERNAL
 ( TYPE ORACLE_LOADER
   DEFAULT DIRECTORY "FILE_MOUNT"
   ACCESS PARAMETERS
   ( RECORDS DELIMITED BY NEWLINE
             SKIP 1
             FIELDS TERMINATED BY 0x'09'
             missing field values are null
                                     )
   LOCATION
    ( 'sj_092007_rbm_p.txt'
    )
 )
REJECT LIMIT UNLIMITED;
```

11. Execute the preprocess sql for the trial.

    See the sample preprocessor example_Preprocess_RBM_data.sql.

    - ☐ If there is a separate subject/sample mapping file, validate that the subject/sample mappings are correct.
    - ☐ Ensure that the subject/site (patient) can map to a valid patient_num in the `i2b2demodata.patient_dimension` table.
    - ☐ Ensure that the antigen_name can map to a valid antigen_name in `deapp.stg_rbm_antigen_gene`. Create node curation records if needed.
    - ☐ Truncate and populate the `deapp_wz.stg_subject_rbm_data` table.

      You will need to analyze the RBM data to determine if any data manipulation needs to be done to load the data into the columns correctly.

12. Run the stored procedure `I2B2_PROCESS_RBM_DATA`. This stored procedure creates the entries in the i2b2 tables, calculates the z-score for the trial, and populates the `deapp.de_subject_rbm_data` table.

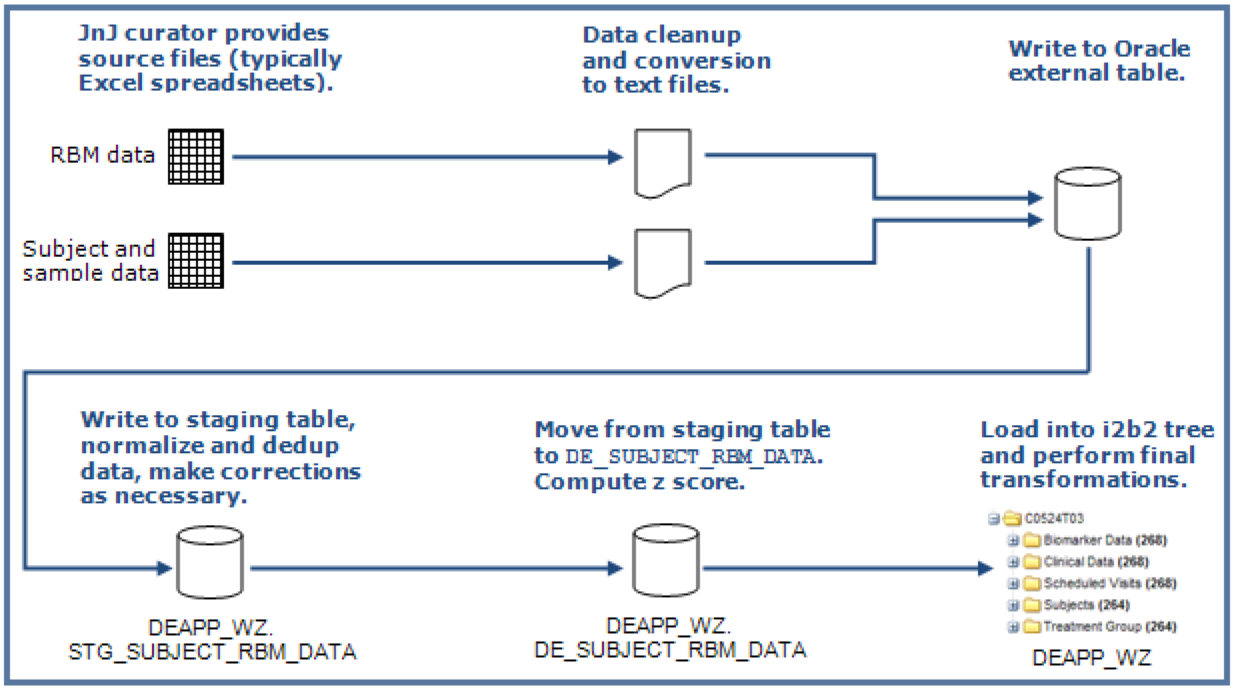
    ```
    - load RBM data
    declare
    TrialID varchar2(100);
    begin
    TrialID := 'XXXXX'; -- < enter the trial/study id here
    i2b2_process_rbm_data(TrialID);
    end;
    ```

13. Check that the nodes were created in the `i2b2` and `concept_dimension` tables.

14. In Dataset Explorer, drag one of the timepoint folders to a box in Subset 1 and generate a heat map.

Illustration of the RBM Data Loading Workflow

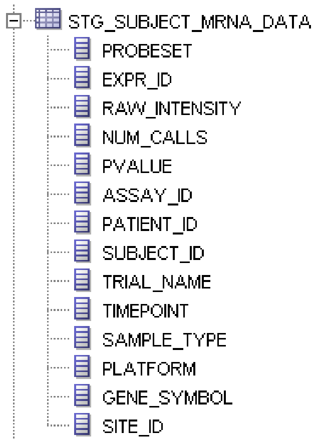
The following figure illustrates the workflow in the development environment:


JnJ curator provides source files (typically Excel spreadsheets).
Data cleanup and conversion to text files.
Write to Oracle external table.
RBM data
Subject and sample data
Write to staging table, normalize and dedup data, make corrections as necessary.
Move from staging table to DE_SUBJECT_RBM_DATA. Compute z score.
Load into i2b2 tree and perform final transformations.
C0524T03
Biomarker Data (268)
Clinical Data (268)
Scheduled Visits (268)
Subjects (264)
Treatment Group (264)
DEAPP_WZ. STG_SUBJECT_RBM_DATA
DEAPP_WZ. DE_SUBJECT_RBM_DATA
DEAPP_WZ

Loading mRNA Affymetrix Data

mRNA Affymetrix data is provided to you by the Curation Analyst. Typically, you will load mRNA data after you load the associated clinical trial data.

Load mRNA Affymetrix data into `DEAPP_WZ.STG_SUBJECT_MRNA_DATA` in the development environment. As you perform the steps to load and transform the raw mRNA Affymetrix data, it may help to be aware of table's columns, as shown in the figure below:


STG_SUBJECT_MRNA_DATA
- PROBESET
- EXPR_ID
- RAW_INTENSITY
- NUM_CALLS
- PVALUE
- ASSAY_ID
- PATIENT_ID
- SUBJECT_ID
- TRIAL_NAME
- TIMEPOINT
- SAMPLE_TYPE
- PLATFORM
- GENE_SYMBOL
- SITE_ID

Workflow Summary

The following steps outline a typical workflow for loading and transforming mRNA Affymetrix data:

1. Download the data files (typically Excel spreadsheets) from the eRoom and unzip any zipped files.

   You should have the following raw data files:

   - A file containing mRNA Affymetrix data.
   - A file containing information about the subjects and samples in the study (subject/sample mapping file).

   It is likely that the source files will not have the same columns of data from study to study.

2. Make sure that the trial still has the "study id" in the node name path for the i2b2 and concept_dimension tables. If not, use stored procedure `i2b2_rename_node` to rename the node back to its original short name.

3. Examine the data in the source files, noting the kinds of changes you need to make to load the data into the `STG_SUBJECT_MRNA_DATA` table.

   For example, in one study, the patient ID appeared in the same column as other data. The patient ID needed to be extracted from the column to prepare it for loading into the `PATIENT_ID` column of the `STG_SUBJECT_MRNA_DATA` table.

4. Pivot the mRNA data using the **run_JnJFilePivot.bat** file (located in `/transmart/PDF/ExternalLinks` with **JnJFilePivot.jar**). This creates a unique record for each experiment ID (`expr_id`) and probe set. There may be up to three columns of data (raw_intensity, number of calls, p_value).

5. Zip up the output from the above step (due to the size of the output).

6. Determine if a subject (patient)/sample mapping file exists. If not, see if the clinical data for the study has enough information to manually create a subject/sample mapping file using the platform, sample type, and optionally timepoint identified in the steps below. This will usually be found in a file used to develop the "Samples And Timepoints" data. If not, contact the curator.

7. Check that reference data is available for the platform. The platform is specified in the "Public Trial Tracking Survey of GEO Sets" spreadsheet maintained by the Johnson & Johnson Data Steward.

   The tracking spreadsheet is an attachment on the Recombinant wiki (Dashboard > J & J Biomarkers > Project Management > Track Data Loads), or in the gene expression subject-to-sample mapping file from the curator.

   ```
   select count( * ) from reference.annotation_deapp
   where gpl_id = 'GPL5981' <= substitute platform here
   ```

If the count = 0, the platform will need to be loaded:

- ☐ Locate the data for the platform in GEO:

  http://www.wip.ncbi.nlm.nih.gov/projects/geo/

- ☐ Enter the GPL number (GPLXXXX) in the GEO Accession box.
- ☐ Download the data using the "Download Full Table" button or the "Soft formatted family file(s)" and unzip the file if needed.
- ☐ Edit **run_GPLInfo.bat** (located in `/transmart/PDF/ExternalLinks` with **GPLInfo.jar**) and modify parameters as needed. The .bat file contain examples for Affymetrix, Agilent, and Illumina platforms.
- ☐ Run **run_GPLInfo.bat** to create the platform/probe/gene symbol/gene id file.
- ☐ Upload the file to the development server /u01/biomart/biomart_lz directory.
- ☐ In SQLDeveloper:

```
alter table biomart_lz.deapp_annot_extrnl location('your filename
here');
begin
i2b2_load_annotation_deapp;
end;

select * from deapp.de_gpl_info
where platform = 'GPL5981' <= substitute platform here
and organism = 'Homo sapiens'
```

Record should exist. If not, insert a record for the platform and its title from GEO.

8. Determine the sample type(s). Some gene expression data will have the sample type in the subject/sample mapping file. Otherwise, the curator should supply the sample type(s).

   If the study data that is associated with the mRNA data has any "Sample And Timepoints" samples, check if the sample type provided is in this list with the EXACT spelling. If the spellings are merely similar, contact the curator for direction. It is not necessary for the sample type of the gene expression data to be in the study data.

9. Check if there are timepoints associated with the gene expression data or samples. If there are, ensure that the appropriate records exist in the node_curation table to transform the timepoints into standard formats if the timepoints are not in the same format as in the study data.

   Check if the curated timepoints exist in "Samples And Timepoints" with the EXACT spelling. If the spellings are merely similar, contact the curator for direction.

10. Upload the pivoted mRNA zip file and subject/sample mapping file to the development server.

11. Unzip the mRNA data and move all files to `/u01/biomart/biomart_lz`.

12. Clean up the data in each file, and load the data into two tab-separated text files named `MM_mRNA_DATA.txt` and `MM_mRNA_subjects_samples.txt`.

13. Move the data from both text files into Oracle external tables that are based on the columns found in the data. A sample script is shown below:

```
CREATE TABLE "BIOMART_LZ"."MULTMYEL_MRNA_SAMP_EXTRNL"
 ( "EXPR_ID" VARCHAR2(100 BYTE)
  ,"SUBJECT_ID" VARCHAR2(100 BYTE)
 )
 ORGANIZATION EXTERNAL
  ( TYPE ORACLE_LOADER
    DEFAULT DIRECTORY "BIOMART_LZ"
    ACCESS PARAMETERS
    ( records delimited by newline  skip 1
      fields terminated by 0X'09'
     LRTRIM
      MISSING FIELD VALUES ARE NULL
         )
    LOCATION
     ( 'MM_mRNA_subjects_samples.txt'
     )
  )
;

CREATE TABLE "BIOMART_LZ"."MULTMYEL_MRNA_DATA_EXTRNL"
 ( "PROBESET" VARCHAR2(100 BYTE)
  ,"EXPR_ID" VARCHAR2(100 BYTE)
  ,"RAW_INTENSITY" varchar2(50)
 )
 ORGANIZATION EXTERNAL
  ( TYPE ORACLE_LOADER
    DEFAULT DIRECTORY "BIOMART_LZ"
    ACCESS PARAMETERS
    ( records delimited by newline  skip 1
      fields terminated by 0X'09'
      LRTRIM
      MISSING FIELD VALUES ARE NULL
          )
    LOCATION
     ( 'MM_mRNA_Data.txt'
     )
  )
;
```

14. Execute the preprocess sql for the trial.

    See the sample preprocessor example_PreProcess_mRNA.sql.

    - ☐ Validate that the subject/sample mappings are correct.
    - ☐ Ensure that the subject (patient) can map to a valid `patient_num` in the `i2b2demodata.patient_dimension` table.
    - ☐ Truncate and populate the `DEAPP_WZ.STG_SUBJECT_MRNA_DATA` staging table.
        - You will need to analyze the gene expression data to determine if any data manipulation needs to be done to load the data into the columns correctly.
        - `assay_id` is numeric and is usually the numeric part of the `expr_id`.

15. Run the stored procedure `i2b2_process_mrna_data`. This stored procedure creates the entries in the i2b2 tables, calculate the z-score for the trial, and populates the `deapp.de_subject_microarray_data` table.

```
-- load mRNA data
declare
TrialID varchar2(100);
begin
TrialID := 'XXXXX'; <= enter the trial/study id here
i2b2_process_mrna_data(TrialID,null,'R');   -- use R if the values
    are raw, L if the values are already log-transformed, and T if
    the values are already z-scored.  Check with the curator as to
    which value to use.
end;
```

16. Check that a partition was created in `deapp.de_subject_microarray_data` for the trial.

17. Check that the nodes were created in the `i2b2` and `concept_dimension` tables.

For additional information, click the following link:

- Public_Study_subject_sample_mapping_template.xltm

Loading SNP Data

Loading SNP data requires these high-level steps:

- Step 1. Prepare and Annotate the SNP Data (page 74)
- Step 2. Transform and Load the SNP Data (page 75)
- Step 3. Load the SNP Data into HAPLOVIEW_DATA (page78)

Step 1. Prepare and Annotate the SNP Data

All SNP data should populated into the temporary table `DEAPP_WZ.TMP_SNP_DATA`:

```
drop table tmp_snp_data purge;
create table tmp_snp_data(
  I2B2_ID                    NUMBER(20),
  JNJ_ID                     VARCHAR2(30),
  FATHER_ID                  NUMBER(5) default 0,
  MOTHER_ID                  NUMBER(5) default 0,
  SEX                        NUMBER(1) default 0,
  AFFECTION_STATUS           NUMBER(1) default 0,
  CHROMOSOME                 VARCHAR2(10),
  GENE                       VARCHAR2(50),
  TRIAL_NAME                 VARCHAR2(50),
  SNP_ID                     VARCHAR2(50),
  POS                        VARCHAR2(50),  -- SNP's phyical location
  RS                        VARCHAR2(50),   -- SNP's RS #
  SNP_VALUE                 VARCHAR2(10) default '0 0',
  primary key (i2b2_id, gene, snp_id, trial_name)
) nologging;
```

Note that:

- If `SEX`, `MOTHER_ID`, `FATHER_ID`, and `AFFECTION_STATUS` are unknown, set them to 0.
- Values in `SNP_VALUE` should be converted to 1 (A), 2(C), 3(G), 4(T) and 0 (others).

Step 2. Transform and Load the SNP Data

Once SNP data is populated in the temporary table `TMP_SNP_DATA`, it can be transformed and loaded into the temporary table `TMP_HAPLOVIEW_DATA`:

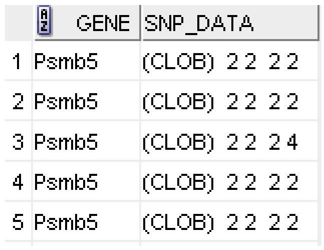
```
DROP TABLE TMP_HAPLOVIEW_DATA PURGE;
CREATE TABLE TMP_HAPLOVIEW_DATA
(
  I2B2_ID            NUMBER(20),
  JNJ_ID             VARCHAR2(30 BYTE),
  FATHER_ID          NUMBER(5),
  MOTHER_ID          NUMBER(5),
  SEX                NUMBER(1),
  AFFECTION_STATUS   NUMBER(1),
  CHROMOSOME         VARCHAR2(10 BYTE),
  GENE               VARCHAR2(50 BYTE),
  RELEASE            NUMBER(4),
  RELEASE_DATE       DATE,
  TRIAL_NAME         VARCHAR2(50 BYTE),
  SNP_DATA           CLOB
) NOLOGGING;
```

Transformation and loading procedure:

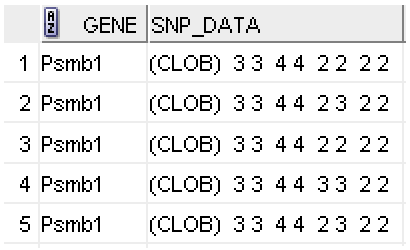
```
   NAME:     load_haploview_data

   PURPOSE:  extract data from table tmp_snp_data, transform it and
             then load into tmp_haploview_data. Both tables must
             exist before starting this procedure.

      drop table TMP_HAPLOVIEW_DATA purge;
      CREATE TABLE TMP_HAPLOVIEW_DATA
        (
          I2B2_ID            NUMBER(20),
          JNJ_ID             VARCHAR2(30 BYTE),
          FATHER_ID          NUMBER(5),
          MOTHER_ID          NUMBER(5),
          SEX                NUMBER(1),
          AFFECTION_STATUS   NUMBER(1),
          CHROMOSOME         VARCHAR2(10 BYTE),
          GENE               VARCHAR2(50 BYTE),
          RELEASE            NUMBER(4),
          RELEASE_DATE       DATE,
          TRIAL_NAME        VARCHAR2(50 BYTE),
          SNP_DATA          CLOB
        )  NOLOGGING;


      Load SNP data into tmp_snp_data table:
          SNP Calls: A -> 1;  C -> 2;  G -> 3; T -> 4; others -> 0

      drop table tmp_snp_data purge;
      create table tmp_snp_data(
         I2B2_ID             NUMBER(20),
         JNJ_ID              VARCHAR2(30),
         FATHER_ID           NUMBER(5) default 0,
         MOTHER_ID           NUMBER(5) default 0,
         SEX                 NUMBER(1) default 0,
         AFFECTION_STATUS    NUMBER(1) default 0,
         CHROMOSOME          VARCHAR2(10),
         GENE                VARCHAR2(50),
         TRIAL_NAME          VARCHAR2(50),
         SNP_ID              VARCHAR2(50),
         POS                 VARCHAR2(50),  -- SNP's phyical location
         RS                  VARCHAR2(50),  -- SNP's RS #
         SNP_VALUE           VARCHAR2(10) default '0 0',
         primary key (i2b2_id, gene, snp_id, trial_name)
       ) nologging;

For LYM3001:

insert into tmp_snp_data(i2b2_id, jnj_id, sex, gene, snp_id, pos, rs, snp_value, trial_name)
select i2b2_id, jnj_id, sex, gene, snp_id, snp_id, snp_id, snp, 'LYM3001'
from snp_lym3001;

update tmp_snp_data set chromosome='chr1' where gene in ('Fcgr2a', 'Fcgr3a', 'Psmb2');
update tmp_snp_data set chromosome='chr6' where gene in ('Psmb1', 'Psmb9', 'Psmb8');
update tmp_snp_data set chromosome='chr14' where gene in ('Psmb5');
update tmp_snp_data set chromosome='chr17' where gene in ('Psmb6');
commit;

******************************************************************************/

create or replace PROCEDURE load_haploview_data IS

  cnt NUMBER;
  gene varchar2(100);
  snp_id varchar2(100);

BEGIN

-- check if temporary table tmp_haploview_data exists is not null, if so,
-- then truncate it, otherwise go to the next step

select count(*) into cnt from TMP_HAPLOVIEW_DATA;
if cnt > 0 then
    execute immediate 'truncate table TMP_HAPLOVIEW_DATA';
    commit;
end if;

-- populate metadata for Haploview: SNP ID
execute immediate
    'insert into TMP_HAPLOVIEW_DATA(i2b2_id, jnj_id, chromosome, gene, trial_name)
     select distinct 0, ''0'', chromosome, gene, upper(trial_name)
     from tmp_snp_data';
commit;

-- populate metadata for Haploview: Dummy SNP Calls
execute immediate
    'insert into TMP_HAPLOVIEW_DATA(i2b2_id, jnj_id, chromosome, gene, trial_name)
     select distinct 0, ''1'', chromosome, gene, upper(trial_name)
     from tmp_snp_data';
commit;
```

```
-- populate metadata for Haploview: SNP physical location
execute immediate
    'insert into TMP_HAPLOVIEW_DATA(i2b2_id, jnj_id, chromosome, gene, trial_name)
     select distinct 0, ''2'', chromosome, gene, upper(trial_name)
     from tmp_snp_data';
commit;

-- populate metadata for Haploview: SNP RS#
execute immediate
    'insert into TMP_HAPLOVIEW_DATA(i2b2_id, jnj_id, chromosome, gene, trial_name)
     select distinct 0, ''3'', chromosome, gene, upper(trial_name)
     from tmp_snp_data';
commit;

-- add a record for each subject's each gene
execute immediate
    'insert into TMP_HAPLOVIEW_DATA(i2b2_id, jnj_id, sex, father_id, mother_id,
            affection_status, chromosome, gene, trial_name)
     select distinct i2b2_id, jnj_id, sex, father_id, mother_id,
            affection_status, chromosome, gene, upper(trial_name)
     from tmp_snp_data';
commit;

-- populate column SNP_DATA
for c in (select distinct gene, snp_id, pos, rs from tmp_snp_data order by gene, snp_id) loop

     dbms_output.put_line(c.gene || '  ' || c.snp_id);

     -- fill SNP_ID for SNP_DATA

     execute immediate
         'update TMP_HAPLOVIEW_DATA
          set snp_data=snp_data||'', '||c.snp_id||'''
          where i2b2_id=0 and jnj_id=''0'' and gene='''||c.gene||'''';
    commit;

    -- fill dummy SNP calls for SNP_DATA
    execute immediate
         'update TMP_HAPLOVIEW_DATA
          set snp_data=snp_data||''  0 0 ''
          where i2b2_id=0 and jnj_id=''1'' and gene='''||c.gene||'''';
    commit;

    -- fill SNP Position for SNP_DATA
    execute immediate
         'update TMP_HAPLOVIEW_DATA
          set snp_data=snp_data||'', '||c.pos||'''
          where i2b2_id=0 and jnj_id=''2'' and gene='''||c.gene||'''';
    commit;

    -- fill RS# for SNP_DATA
    execute immediate
         'update TMP_HAPLOVIEW_DATA
          set snp_data=snp_data||'', '||c.rs||'''
          where i2b2_id=0 and jnj_id=''3'' and gene='''||c.gene||'''';
    commit;

    -- fill subject's SNP Data for SNP_DATA
    update TMP_HAPLOVIEW_DATA t1
    set snp_data=snp_data||'  '||(select snp_value from tmp_snp_data
                           where i2b2_id=t1.i2b2_id and gene=c.gene and snp_id=c.snp_id)
    where i2b2_id>0 and gene=c.gene;
    commit;

   end loop;

  -- remove SNP_DATA column's leading comma ","
  execute immediate 'update TMP_HAPLOVIEW_DATA set snp_data=substr(snp_data, 2)';
  commit;

END load_haploview_data;
```

Note: After SNP data is transformed and loaded into the table `TMP_HAPLOVIEW_DATA`, make sure that `SNP_DATA` for all records with the same gene are the same length. If they are not, tranSMART's Haploview feature will not work. For example, the following partial result sets were extracted from `TMP_HAPOVIEW_DATA` for the genes Psmb5 and Psmb1:



	GENE	SNP_DATA
1	Psmb5	(CLOB) 2 2 2 2
2	Psmb5	(CLOB) 2 2 2 2
3	Psmb5	(CLOB) 2 2 2 4
4	Psmb5	(CLOB) 2 2 2 2
5	Psmb5	(CLOB) 2 2 2 2



	GENE	SNP_DATA
1	Psmb1	(CLOB) 3 3 4 4 2 2 2 2
2	Psmb1	(CLOB) 3 3 4 4 2 3 2 2
3	Psmb1	(CLOB) 3 3 4 4 2 2 2 2
4	Psmb1	(CLOB) 3 3 4 4 3 3 2 2
5	Psmb1	(CLOB) 3 3 4 4 2 3 2 2

Step 3. Load the SNP Data into HAPLOVIEW_DATA

The final step is to copy the data from the `DEAPP_WZ.TMP_HAPLOVIEW_DATA` table to `DEAPP_HAPLOVIEW_DATA`. For example, you can run any of these SQL commands:

- From the `deapp_wz` account:

```
insert into deapp.haploview_data select * from tmp_haploview_data;
```

- From `deapp` account:

```
insert into haploview_data select * from deapp_wz.tmp_haploview_data;
```

- From other Oracle account:

```
insert into deapp.haploview_data select * from deapp_wz.tmp_haploview_data;
```

At runtime, several Java classes extract SNP data from the table `DEAPP.HAPLOVIEW_DATA` and format the data as required for the Haploview visualization feature. These Java classes are embedded in tranSMART and work transparently.

Adding Security for a Study

As part of the data loading process for all trials and studies, two tables are updated with security information through the following procedures:

- `I2B2_CREATE_PATIENT_TRIAL` – Inserts a record for each patient in the trial or study into the `i2b2demodata.patient_trial` table.
- `I2B2_LOAD_SECURITY_DATA` procedure truncates and reloads the `i2b2metadata.i2b2_secure` table for all trials and studies.

Each of these tables has a `secure_obj_token` column. This column is populated with "EXP:" and the trial or study id (EXP:C0168T54). For Public Studies, the `secure_obj_token` is always EXP:PUBLIC.

All studies other than those in the Public Studies tree must have security defined before a user can be granted access to the study. Security for a study is enabled through the Admin function in the tranSMART user interface.

For instructions on enabling security protection for a study, see the following document:

SetUpStudyForSecurity.doc

Viewing Audit and Error Logs

The Oracle procedures used to load clinical trial data, RBM data, and mRNA Affymetrix data insert records into an audit table as they are run. These records can be used to view the progress of the procedure and to check record counts for the steps.

The audit data columns are:

Column	Description
SEQ_ID:	Sequential record number.
JOB_ID:	Job number.
DATABASE_NAME:	Name of the database where the procedure is being executed.
PROCEDURE_NAME:	Name of the procedure that is running.
STEP_DESC:	Description of the current step or activity.
STEP_STATUS:	Status information supplied by the procedure.
RECORDS_MANIPULATED:	Number of records processed in the step. For steps that do not return a row count, the number is 0.
STEP_NUMBER:	Sequential number of the step within the job. This is determined by the procedure.
JOB_DATE:	Date/time stamp of when the record was inserted into the audit table.
TIME_ELAPSED_SECS:	Elapsed time since last record for the job was inserted.

To view the latest run

```
select a.* from control.cz_job_audit a,control.cz_job_master m
where m.job_id = a.job_id
and m.job_id = (select max(x.job_id) from control.cz_job_master x)
order by a.job_date
```

To view a specific run, specify a job_id

```
select a.* from control.cz_job_audit a,control.cz_job_master m
where m.job_id = a.job_id and m.job_id = 681
order by a.job_date
```

If the job has failed, the `STEP_DESC` will contain the following message:

```
Job Failed: See error log for details.
```

To display the error

```
select * from control.cz_job_error
where job_id = 681
```

The `ERROR_MESSAGE` column will contain the Oracle error message, and `ERROR_BACKTRACE` will contain the name and line number of the failing procedure.

```
ERROR_MESSAGE: ORA-12899: value too large for column
"DEAPP_WZ"."DE_SUBJECT_MICROARRAY_DATA"."TIMEPOINT" (actual: 32,
maximum: 30)
ERROR_BACKTRACE: ORA-06512: at "CONTROL.I2B2_PROCESS_MRNA_DATA", line
239
```

Troubleshooting

Issue	**Cause**	**Solution**
A numeric node (such as Age) does not show any data when the **Show Histogram** button is clicked in the Set Value dialog.	Duplicate nodes in the `CONCEPT_DIMENSION` table: `select * from concept_dimension where concept_path like '%\C0743T10\Subjects\Demographics\Age\;`	Remove one node.
The UI is available but no trials appear in the Navigate Terms tab. Trials appear when searching on the Search by Subject tab, but cannot be expanded.	JBoss is down.	Restart JBoss.
A user has access to the Dataset Explorer tree view for a trial, but no data is returned (zero patients) when the user attempts to generate results.	Security records for the specific trial are not loaded in the `OBSERVATION_FACT` table. The table must contain one record per every patient in the trial.	Run the stored procedure `I2B2_CREATE_SECURITY_FOR_TRIAL` for this trial.

Chapter 5
Data Quality Assurance

The involvement in the data testing effort will be shared across a number of team members including the Data Steward, the Curation Analyst, SMEs, architects, power users, project managers, and you, the ETL Analyst.

Data QA should be overseen by a single owner who has overall responsibility for the quality of the data in the data warehouse, as well as the primary role of executing data quality test plans. The quality assurance owner will be held accountable for quality issues, achieving their resolution, and sharing all results with the Data Steward.

Data testing takes place in the staging/QA environment on the following server:

Server Host Name	IP Address	Purpose
`tmqa`	174.129.234.96	Database and application server

QA Considerations

Things to consider during data testing:

1. **Was all of the data from the input files loaded into the warehouse?**

   Example issue

   A load procedure is designed to load lab data, but it is unable to handle rows that use a string in a field where there is normally a number. The result is that the input file has 100 rows and the representative table in the warehouse has only 90.

   Row count integrity checks

   Run and document tests during ETL development comparing row counts of input to resulting data tables. Establish counting routines between transformations to determine whether the count of rows attempted to be loaded are in the resulting table.

2. **Is the loaded data complete? Does it contain what it should?**

   Example issue

   A successfully loaded table is missing a large block of records because the source system failed to extract records of a specific type, or does not contain a complete set of records that the user is interested in viewing.

   Examples of what could cause missing data:

   - A data source that should contain 5,000 probes only contains 2,000 probes.
   - A certain subset of data is excluded – for example, suppose there are 500 subjects in a trial, but only 340 samples used for gene expression match the subjects. This may be correct, but it also may be that the identifiers have been incorrectly logged, or that there is a second set of gene expression data in a separate file.
   - The production file contains 6,000 records, but only 50 were loaded (which happens to be the same volume used for testing).

   Reasonableness test

   For any complex data structure, a set of specific tests should be created to check whether the data described and expected from a source matches the data to be loaded and ultimately included in the warehouse. Prior to loading, a review of the data is made prior to check that interrelated data is aligned properly across sources. When suspected anomalies occur, a request should be made to data sources regarding the potential data integrity errors to clarify and resolve any potential issues with integrity.

3. **Does the data look right? Does it deliver accurate information from the end-user's stand-point?**

   Example issue

   Often major errors are easy for end-users to see when using applications, but go unnoticed when looking at the data directly in the database. So it may be that a field is always blank, a link goes to a page that doesn't represent the proper drill-down for an item, or searches don't retrieve any results.

   Smoke test

   Test any complete build of data in end-user applications before releasing it to the end-users. Run through 5-to-10 pre-defined user scenarios with careful attention to the integrity of the application and to the validity and appropriateness of data on the screen.

   User beta test

   Prior to releasing any major new data set, release the data to an experienced set of beta users who are informed that the status is a pre-production data build. Ask the users to validate usability, looking for any possible errors in the data or the application.

SME validation test

For newly curated content (such as a specific clinical trial), a subject matter expert on that content should review the information as it will appear in the application to an end-user, and sign-off on the validity of the curated descriptions and the appearance of the data.

Data use validation pilot test

For data that will be used in a complex workflow, first run the data through a QA test use case, and then run it through a production-level use case, with a real-world end-user providing feedback on the quality of the data. Often, one or several end-users can identify key errors before a larger scale audience can, due to gaps between their expected outcomes and results they experience.

4. **Is the current build of the database or data marts getting new data feeds, and is it reporting tables generated from new procedures that was not in the last one?**

   Example issue

   A new version or build of the data warehouse or data marts contains tables that are truncated or elongated due to a systematic process.

   Table version match

   Builds between versions of the data warehouse or data marts are automatically compared across all tables in the schema to identify differences between the row counts in each table. For tables that have abnormal growth or reduction, a QA analyst seeks rational reasons before promoting the build into a production environment for end-user access.

5. **Are there errors and problems found in prior builds or prior processes that had been fixed but that are reappearing?**

   Example issue

   A problem is identified in testing that is identical to a problem that occurred in a prior implementation or test.

   Learned QA tests

   As the data warehouse matures, new and unexpected quality issues may be discovered late in the release schedule by developers or end-users. Resolving them in a prompt fashion is necessary.

   Just-in-time QA requires the occasional fire drill to resolve an issue found at an inconvenient time. These unexpected issues are logged, and the team comes up with new tests to address them. As the new tests are executed in the next cycle, watch for the reoccurrence of the symptoms or directly seek the cause itself.

6. **Are all QA test procedures in the QA test plan actually executed?**

   QA test checklist

   Review all QA items that have been executed to verify that the entire QA plan has been executed for each build.

7. **When the testers are satisfied that the data is production-ready, the Data Steward reviews the testing results and signs off on the migration of the data to the production server.**

   Having an accountable business resource validate and accept the data, including the security and usage mechanisms associated with each release, serves to provide formal business oversight and responsibility for all aspects of the solution about to be moved into production.

Sample Test Harness

The test harness runs specific tests against a database's tables, or does comparison tests between two sets of results.

All tables and stored procedures referenced in this section are located in the `CONTROL` database.

Tables

The following tables in the `CONTROL` database contain information about the tests in the test harness:

Table Name	Description
`CZ_TEST`	Contains names and other information about the tests, the table being tested, and the SQL code executed against the table.
`AZ_TEST_RUN`	Contains results of the execution of a collection of tests (Pass/Fail, start/end).
`AZ_TEST_STEP_RUN`	Contains results of the execution of one test (Pass/Fail, start/end).
`AZ_TEST_STEP_ACT_RESULTS`	Contains detailed information about the results of one test, including any error information.
`AZ_TEST_COMPARISON_INFO`	Contains information about the current and previous tests for comparison purposes.

Stored Procedures

The following sections describe the stored procedures `EXEC_TEST_RUN` and `EXEC_TEST`.

EXEC_TEST_RUN

In the stored procedure outline shown below:

- The `CATEGORY` information is used to select the tests to be run (for example, Functional, Functional-I2B2, Clinical Trials).
- If the value for any of the `CATEGORY` fields is `null`, the stored procedure retrieves everything for that category.
- PARAM1 is used to replace a value in the SQL stored in `CZ_TEST`.
- The SQL value needs to be `PARAM1` to be replaced at runtime.

```
INPUT
   CATEGORY: (From CZ_TEST)
   SUB_CATEGORY1: (From CZ_TEST)
   SUB_CATEGORY2: (From CZ_TEST)
   PARAM1: Parameter for SQL
OUTPUT
   NONE
```

EXEC_TEST

In the stored procedure outline shown below, all parameters are passed to this stored procedure from `EXEC_TEST_RUN`.

```
INPUT
   TEST RUN ID
   TEST ID
   PARAM1
OUTPUT
   NONE
```

Note: This stored procedure is called by `EXEC_TEST_RUN`. It should not be called directly.

Process Flow

Tests are created in Microsoft Excel or directly in the `CZ_TEST` table.

Tests include a description, the SQL code, expected results, and categories of tests (three levels – Category, SubCategory1, SubCategory2).

Typical process flow:

1. `When EXEC_TEST_RUN` is executed, it is passed the Categories of tests to run and any required parameters.
2. `EXEC_TEST_RUN` retrieves a list of all the test IDs based on the categories, then calls `EXEC_TEST` for each.
3. `EXEC_TEST` then executes each test, stores the results, and determines if whether the result is pass/fail or error.
4. Once all tests are executed, `EXEC_TEST_RUN` determines an overall status for the run.

Example Usage

1. Run the Test

```
BEGIN
 --params
 --1.CATEGORY
 --2.SUB CATEGORY 1
 --3.SUB CATEGORY 2
 --4.PARAMETER 1
 --5. TEST RUN NAME

exec_test_run('FUNCTIONAL','','','','Function Tests');

exec_test_run('REGRESSION','DATASET
EXPLORER','TOTAL_COUNT','','Regression Tests');

exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY TRIAL','C-2006-
009','Trial Comparison C-2006-009');
exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY
TRIAL','C0168T29','Trial Comparison C0168T29');
exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY
TRIAL','C0168T32','Trial Comparison C0168T32');
exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY
TRIAL','C0168T41','Trial Comparison C0168T41');
exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY
TRIAL','C0168T48','Trial Comparison C0168T48');
exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY
TRIAL','C0168T50','Trial Comparison C0168T50');
```

```
exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY
TRIAL','C0168T54','Trial Comparison C0168T54');
exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY
TRIAL','C0524T02','Trial Comparison C0524T02');
exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY
TRIAL','C0524T03','Trial Comparison C0524T03');
exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY
TRIAL','C0743T10','Trial Comparison C0743T10');
exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY
TRIAL','C1034T07','Trial Comparison C1034T07');
exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY TRIAL','BRC
Antidepressant Study','Trial Comparison BRC Antidepressant Study');
exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY TRIAL','BRC
Depression Study','Trial Comparison BRC Depression Study');
exec_test_run('REGRESSION','DATASET EXPLORER','COUNT BY
TRIAL','C0743X01','Trial Comparison C0743X01');
END;
```

2. Prepare for the Next Test Run

Once a build has been verified, run the following code to write the current test results to the `PREV_` (Previous) fields of `AZ_TEST_COMPARISON_INFO`.

```
UPDATE AZ_TEST_COMPARISON_INFO
  SET PREV_DW_VERSION_ID = CURR_DW_VERSION_ID,
    prev_test_step_run_id = curr_test_step_run_id,
    prev_act_record_cnt = curr_act_record_cnt;
COMMIT;
```

Database Views

- View of all test summaries based on the categories entered.

  `CONTROL.TEST_SUMMARY_VIEW`

- View of detailed test results by run.

  `CONTROL.TEST_DETAIL_VIEW`

Sample Test Spreadsheet

The following file can be loaded directly into the `CZ_TEST` table:

TEST_SPREADSHEET.xls

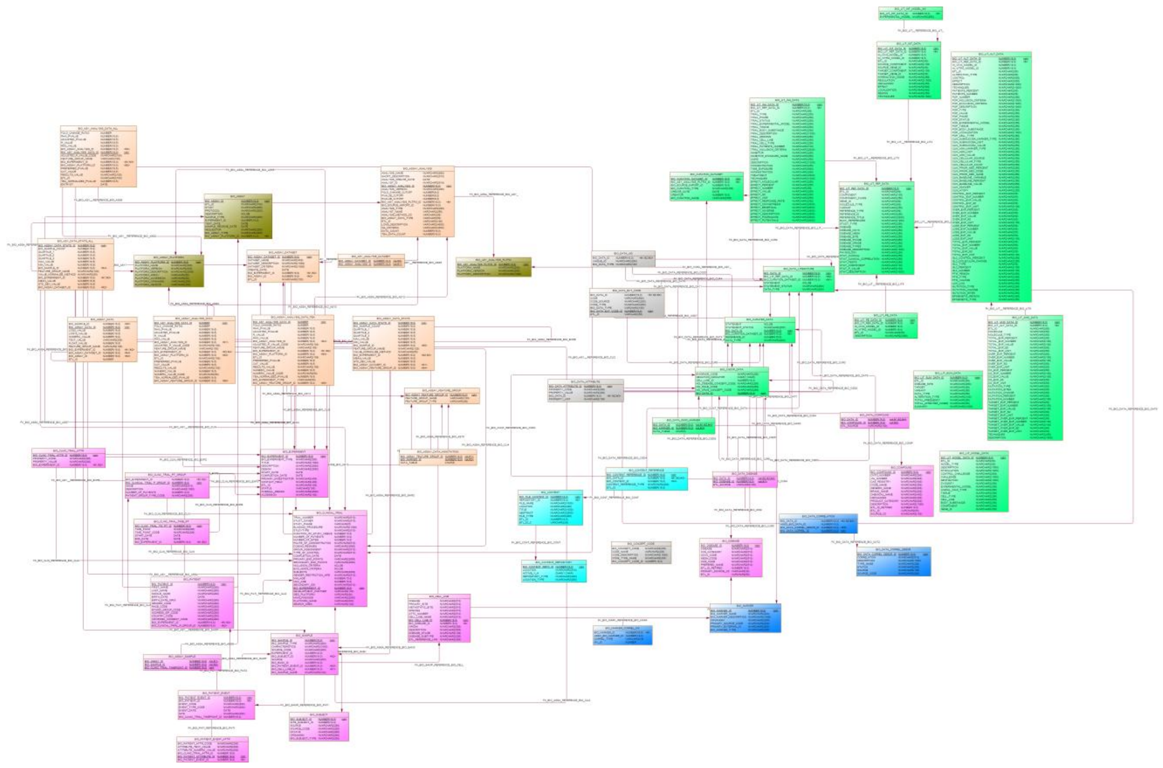
Appendix A
Data Warehouse Schemas

This appendix contains illustrations of the following categories of tables in the tranSMART data warehouse:

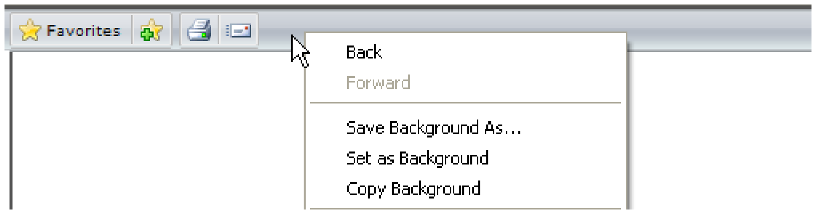
- Core Data Warehouse (page 90)
- Data Mart for tranSMART Search (page 91)
- Dataset Explorer:
  - I2B2METADATA and I2B2DEMODATA (page 92)
  - DEAPP (page 93)

Core Data Warehouse

Click the image to display a larger version of the diagram.

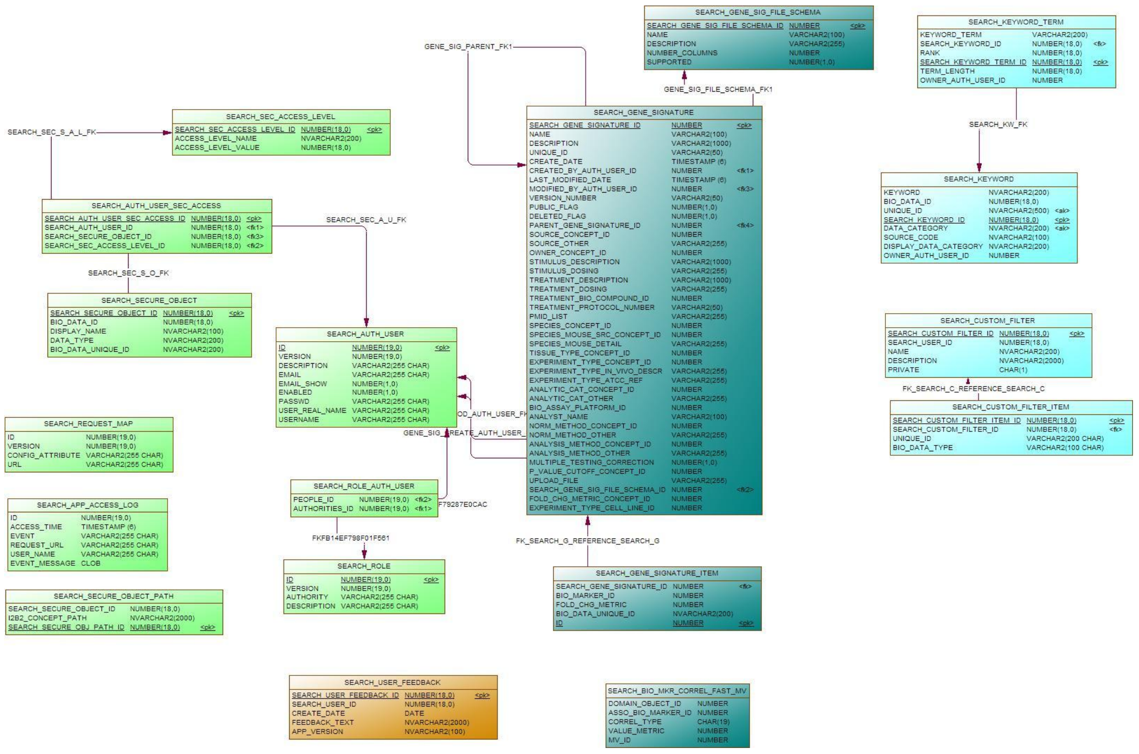


Note: To return to this PDF page after viewing the full-size diagram, click your browser's Back button, if it is displayed, or right-click the solid bar above the diagram, then click Back.



Data Mart for tranSMART Search

Click the image to display a larger version of the diagram.



SEARCH_GENE_SIG_FILE_SCHEMA		
SEARCH_GENE_SIG_FILE_SCHEMA_ID	NUMBER	<pk>
NAME	VARCHAR2(100)	
DESCRIPTION	VARCHAR2(255)	
NUMBER_COLUMNS	NUMBER	
SUPPORTED	NUMBER(1,0)	

SEARCH_KEYWORD_TERM		
KEYWORD_TERM	VARCHAR2(200)	
SEARCH_KEYWORD_ID	NUMBER(18,0)	<fk>
RANK	NUMBER(18,0)	
SEARCH_KEYWORD_TERM_ID	NUMBER(18,0)	<pk>
TERM_LENGTH	NUMBER(18,0)	
OWNER_AUTH_USER_ID	NUMBER	

SEARCH_SEC_ACCESS_LEVEL		
SEARCH_SEC_ACCESS_LEVEL_ID	NUMBER(18,0)	<pk>
ACCESS_LEVEL_NAME	NVARCHAR2(200)	
ACCESS_LEVEL_VALUE	NUMBER(18,0)	

SEARCH_GENE_SIGNATURE		
SEARCH_GENE_SIGNATURE_ID	NUMBER	<pk>
NAME	VARCHAR2(100)	
DESCRIPTION	VARCHAR2(1000)	
UNIQUE_ID	VARCHAR2(50)	
CREATE_DATE	TIMESTAMP (6)	
CREATED_BY_AUTH_USER_ID	NUMBER	<fk1>
LAST_MODIFIED_DATE	TIMESTAMP (6)	
MODIFIED_BY_AUTH_USER_ID	NUMBER	<fk3>
VERSION_NUMBER	VARCHAR2(50)	
PUBLIC_FLAG	NUMBER(1,0)	
DELETED_FLAG	NUMBER(1,0)	
PARENT_GENE_SIGNATURE_ID	NUMBER	<fk4>
SOURCE_CONCEPT_ID	NUMBER	
SOURCE_OTHER	VARCHAR2(255)	
OWNER_CONCEPT_ID	NUMBER	
STIMULUS_DESCRIPTION	VARCHAR2(1000)	
STIMULUS_DOSING	VARCHAR2(255)	
TREATMENT_DESCRIPTION	VARCHAR2(1000)	
TREATMENT_DOSING	VARCHAR2(255)	
TREATMENT_BIO_COMPOUND_ID	NUMBER	
TREATMENT_PROTOCOL_NUMBER	VARCHAR2(50)	
PMID_LIST	VARCHAR2(255)	
SPECIES_CONCEPT_ID	NUMBER	
SPECIES_MOUSE_SRC_CONCEPT_ID	NUMBER	
SPECIES_MOUSE_DETAIL	VARCHAR2(255)	
TISSUE_TYPE_CONCEPT_ID	NUMBER	
EXPERIMENT_TYPE_CONCEPT_ID	NUMBER	
EXPERIMENT_TYPE_IN_VIVO_DESCR	VARCHAR2(255)	
EXPERIMENT_TYPE_ATCC_REF	VARCHAR2(255)	
ANALYTIC_CAT_CONCEPT_ID	NUMBER	
ANALYTIC_CAT_OTHER	VARCHAR2(255)	
BIO_ASSAY_PLATFORM_ID	NUMBER	
ANALYST_NAME	VARCHAR2(100)	
NORM_METHOD_CONCEPT_ID	NUMBER	
NORM_METHOD_OTHER	VARCHAR2(255)	
ANALYSIS_METHOD_CONCEPT_ID	NUMBER	
ANALYSIS_METHOD_OTHER	VARCHAR2(255)	
MULTIPLE_TESTING_CORRECTION	NUMBER(1,0)	
P_VALUE_CUTOFF_CONCEPT_ID	NUMBER	
UPLOAD_FILE	VARCHAR2(255)	
SEARCH_GENE_SIG_FILE_SCHEMA_ID	NUMBER	<fk2>
FOLD_CHG_METRIC_CONCEPT_ID	NUMBER	
EXPERIMENT_TYPE_CELL_LINE_ID	NUMBER	

SEARCH_KEYWORD		
KEYWORD	NVARCHAR2(200)	
BIO_DATA_ID	NUMBER(18,0)	
UNIQUE_ID	NVARCHAR2(500)	<ak>
SEARCH_KEYWORD_ID	NUMBER(18,0)	<pk>
DATA_CATEGORY	NVARCHAR2(200)	<ak>
SOURCE_CODE	NVARCHAR2(100)	
DISPLAY_DATA_CATEGORY	NVARCHAR2(200)	
OWNER_AUTH_USER_ID	NUMBER	

SEARCH_AUTH_USER_SEC_ACCESS		
SEARCH_AUTH_USER_SEC_ACCESS_ID	NUMBER(18,0)	<pk>
SEARCH_AUTH_USER_ID	NUMBER(18,0)	<fk1>
SEARCH_SECURE_OBJECT_ID	NUMBER(18,0)	<fk3>
SEARCH_SEC_ACCESS_LEVEL_ID	NUMBER(18,0)	<fk2>

SEARCH_SECURE_OBJECT		
SEARCH_SECURE_OBJECT_ID	NUMBER(18,0)	<pk>
BIO_DATA_ID	NUMBER(18,0)	
DISPLAY_NAME	NVARCHAR2(100)	
DATA_TYPE	NVARCHAR2(200)	
BIO_DATA_UNIQUE_ID	NVARCHAR2(200)	

SEARCH_AUTH_USER		
ID	NUMBER(19,0)	<pk>
VERSION	NUMBER(19,0)	
DESCRIPTION	VARCHAR2(255 CHAR)	
EMAIL	VARCHAR2(255 CHAR)	
EMAIL_SHOW	NUMBER(1,0)	
ENABLED	NUMBER(1,0)	
PASSWD	VARCHAR2(255 CHAR)	
USER_REAL_NAME	VARCHAR2(255 CHAR)	
USERNAME	VARCHAR2(255 CHAR)	

SEARCH_CUSTOM_FILTER		
SEARCH_CUSTOM_FILTER_ID	NUMBER(18,0)	<pk>
SEARCH_USER_ID	NUMBER(18,0)	
NAME	NVARCHAR2(200)	
DESCRIPTION	NVARCHAR2(2000)	
PRIVATE	CHAR(1)	

SEARCH_CUSTOM_FILTER_ITEM		
SEARCH_CUSTOM_FILTER_ITEM_ID	NUMBER(18,0)	<pk>
SEARCH_CUSTOM_FILTER_ID	NUMBER(18,0)	<fk>
UNIQUE_ID	VARCHAR2(200 CHAR)	
BIO_DATA_TYPE	VARCHAR2(100 CHAR)	

SEARCH_REQUEST_MAP	
ID	NUMBER(19,0)
VERSION	NUMBER(19,0)
CONFIG_ATTRIBUTE	VARCHAR2(255 CHAR)
URL	VARCHAR2(255 CHAR)

SEARCH_ROLE_AUTH_USER		
PEOPLE_ID	NUMBER(19,0)	<fk2>
AUTHORITIES_ID	NUMBER(19,0)	<fk1>

SEARCH_APP_ACCESS_LOG	
ID	NUMBER(19,0)
ACCESS_TIME	TIMESTAMP (6)
EVENT	VARCHAR2(255 CHAR)
REQUEST_URL	VARCHAR2(255 CHAR)
USER_NAME	VARCHAR2(255 CHAR)
EVENT_MESSAGE	CLOB

SEARCH_ROLE		
ID	NUMBER(19,0)	<pk>
VERSION	NUMBER(19,0)	
AUTHORITY	VARCHAR2(255 CHAR)	
DESCRIPTION	VARCHAR2(255 CHAR)	

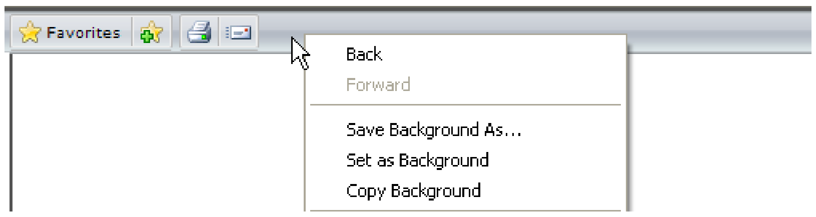
SEARCH_GENE_SIGNATURE_ITEM		
SEARCH_GENE_SIGNATURE_ID	NUMBER	<fk>
BIO_MARKER_ID	NUMBER	
FOLD_CHG_METRIC	NUMBER	
BIO_DATA_UNIQUE_ID	NVARCHAR2(200)	
ID	NUMBER	<pk>

SEARCH_SECURE_OBJECT_PATH		
SEARCH_SECURE_OBJECT_ID	NUMBER(18,0)	
I2B2_CONCEPT_PATH	NVARCHAR2(2000)	
SEARCH_SECURE_OBJ_PATH_ID	NUMBER(18,0)	<pk>

SEARCH_USER_FEEDBACK		
SEARCH_USER_FEEDBACK_ID	NUMBER(18,0)	<pk>
SEARCH_USER_ID	NUMBER(18,0)	
CREATE_DATE	DATE	
FEEDBACK_TEXT	NVARCHAR2(2000)	
APP_VERSION	NVARCHAR2(100)	

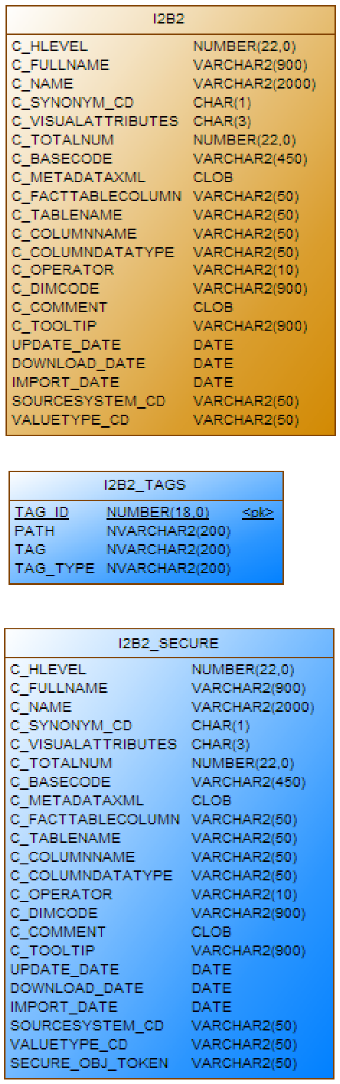
SEARCH_BIO_MKR_CORREL_FAST_MV	
DOMAIN_OBJECT_ID	NUMBER
ASSO_BIO_MARKER_ID	NUMBER
CORREL_TYPE	CHAR(19)
VALUE_METRIC	NUMBER
MV_ID	NUMBER

Note: To return to this PDF page after viewing the full-size diagram, click your browser's Back button, if it is displayed, or right-click the solid bar above the diagram, then click Back.



I2B2METADATA and I2B2DEMODATA

I2B2METADATA



I2B2

C_HLEVEL	NUMBER(22,0)
C_FULLNAME	VARCHAR2(900)
C_NAME	VARCHAR2(2000)
C_SYNONYM_CD	CHAR(1)
C_VISUALATTRIBUTES	CHAR(3)
C_TOTALNUM	NUMBER(22,0)
C_BASECODE	VARCHAR2(450)
C_METADATAXML	CLOB
C_FACTTABLECOLUMN	VARCHAR2(50)
C_TABLENAME	VARCHAR2(50)
C_COLUMNNAME	VARCHAR2(50)
C_COLUMNDATATYPE	VARCHAR2(50)
C_OPERATOR	VARCHAR2(10)
C_DIMCODE	VARCHAR2(900)
C_COMMENT	CLOB
C_TOOLTIP	VARCHAR2(900)
UPDATE_DATE	DATE
DOWNLOAD_DATE	DATE
IMPORT_DATE	DATE
SOURCESYSTEM_CD	VARCHAR2(50)
VALUETYPE_CD	VARCHAR2(50)

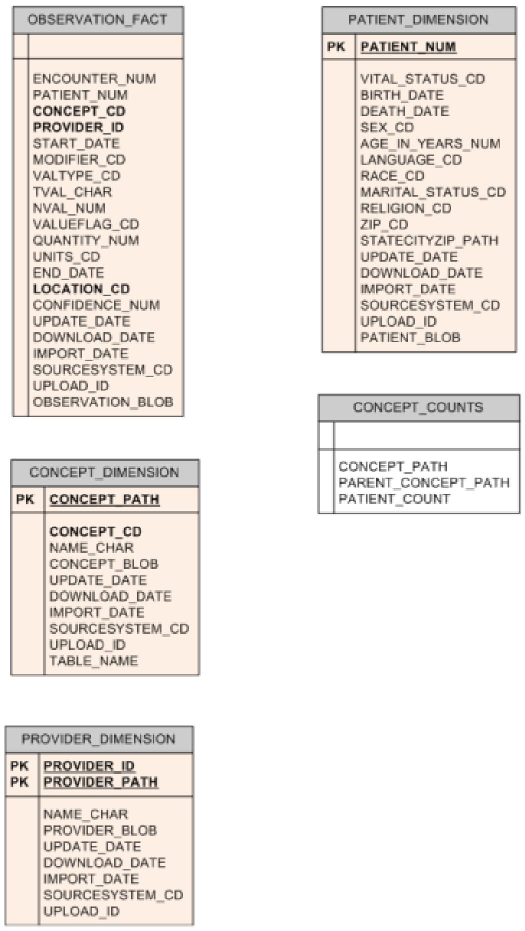
I2B2_TAGS

TAG_ID	NUMBER(18,0)	<pk>
PATH	NVARCHAR2(200)	
TAG	NVARCHAR2(200)	
TAG_TYPE	NVARCHAR2(200)	

I2B2_SECURE

C_HLEVEL	NUMBER(22,0)
C_FULLNAME	VARCHAR2(900)
C_NAME	VARCHAR2(2000)
C_SYNONYM_CD	CHAR(1)
C_VISUALATTRIBUTES	CHAR(3)
C_TOTALNUM	NUMBER(22,0)
C_BASECODE	VARCHAR2(450)
C_METADATAXML	CLOB
C_FACTTABLECOLUMN	VARCHAR2(50)
C_TABLENAME	VARCHAR2(50)
C_COLUMNNAME	VARCHAR2(50)
C_COLUMNDATATYPE	VARCHAR2(50)
C_OPERATOR	VARCHAR2(10)
C_DIMCODE	VARCHAR2(900)
C_COMMENT	CLOB
C_TOOLTIP	VARCHAR2(900)
UPDATE_DATE	DATE
DOWNLOAD_DATE	DATE
IMPORT_DATE	DATE
SOURCESYSTEM_CD	VARCHAR2(50)
VALUETYPE_CD	VARCHAR2(50)
SECURE_OBJ_TOKEN	VARCHAR2(50)

I2B2DEMODATA



OBSERVATION_FACT

	ENCOUNTER_NUM
	PATIENT_NUM
	CONCEPT_CD
	PROVIDER_ID
	START_DATE
	MODIFIER_CD
	VALTYPE_CD
	TVAL_CHAR
	NVAL_NUM
	VALUEFLAG_CD
	QUANTITY_NUM
	UNITS_CD
	END_DATE
	LOCATION_CD
	CONFIDENCE_NUM
	UPDATE_DATE
	DOWNLOAD_DATE
	IMPORT_DATE
	SOURCESYSTEM_CD
	UPLOAD_ID
	OBSERVATION_BLOB

PATIENT_DIMENSION

PK	**PATIENT_NUM**
	VITAL_STATUS_CD
	BIRTH_DATE
	DEATH_DATE
	SEX_CD
	AGE_IN_YEARS_NUM
	LANGUAGE_CD
	RACE_CD
	MARITAL_STATUS_CD
	RELIGION_CD
	ZIP_CD
	STATECITYZIP_PATH
	UPDATE_DATE
	DOWNLOAD_DATE
	IMPORT_DATE
	SOURCESYSTEM_CD
	UPLOAD_ID
	PATIENT_BLOB

CONCEPT_COUNTS

	CONCEPT_PATH
	PARENT_CONCEPT_PATH
	PATIENT_COUNT

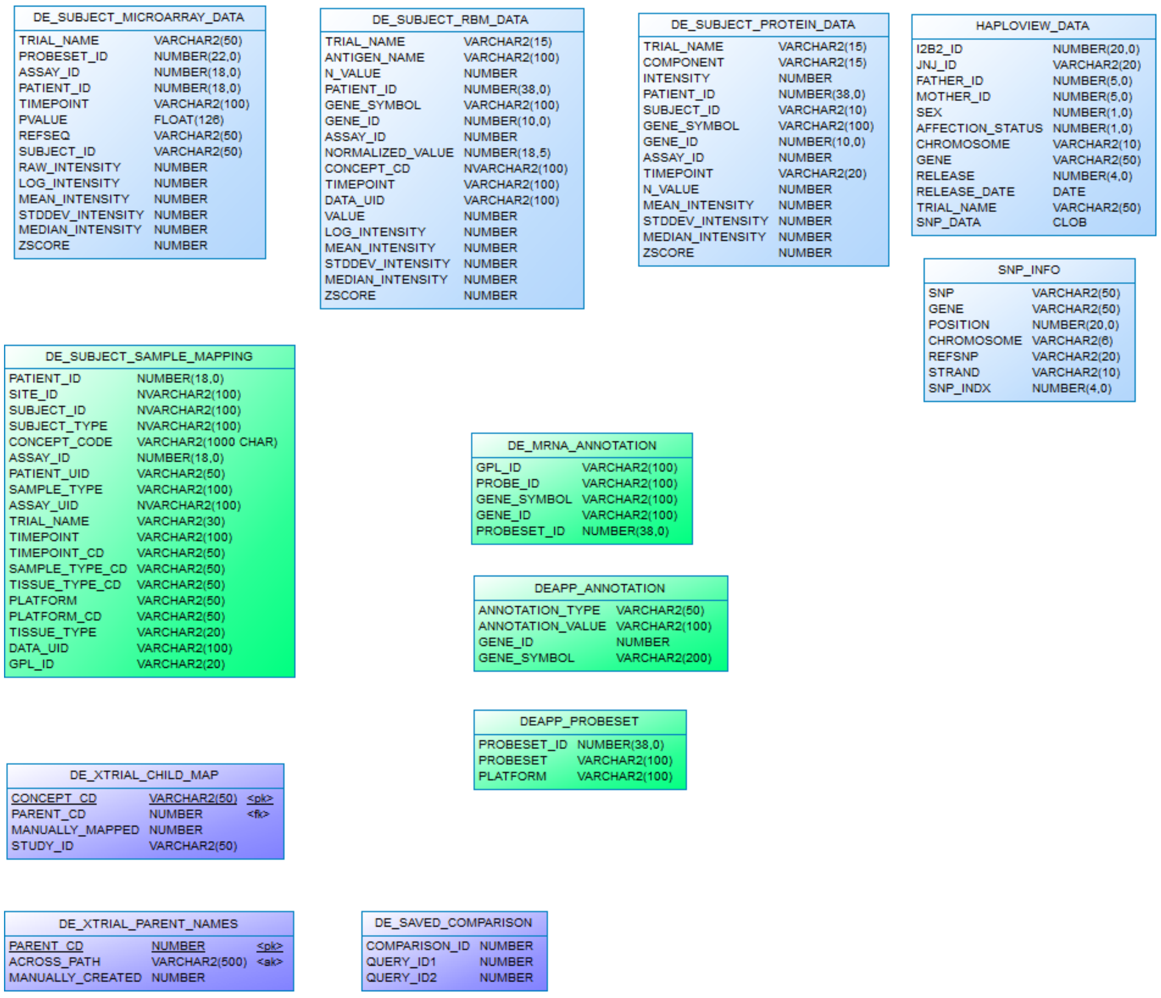
CONCEPT_DIMENSION

PK	**CONCEPT_PATH**
	CONCEPT_CD
	NAME_CHAR
	CONCEPT_BLOB
	UPDATE_DATE
	DOWNLOAD_DATE
	IMPORT_DATE
	SOURCESYSTEM_CD
	UPLOAD_ID
	TABLE_NAME

PROVIDER_DIMENSION

PK	**PROVIDER_ID**
PK	**PROVIDER_PATH**
	NAME_CHAR
	PROVIDER_BLOB
	UPDATE_DATE
	DOWNLOAD_DATE
	IMPORT_DATE
	SOURCESYSTEM_CD
	UPLOAD_ID

DEAPP



DE_SUBJECT_MICROARRAY_DATA	
TRIAL_NAME	VARCHAR2(50)
PROBESET_ID	NUMBER(22,0)
ASSAY_ID	NUMBER(18,0)
PATIENT_ID	NUMBER(18,0)
TIMEPOINT	VARCHAR2(100)
PVALUE	FLOAT(126)
REFSEQ	VARCHAR2(50)
SUBJECT_ID	VARCHAR2(50)
RAW_INTENSITY	NUMBER
LOG_INTENSITY	NUMBER
MEAN_INTENSITY	NUMBER
STDDEV_INTENSITY	NUMBER
MEDIAN_INTENSITY	NUMBER
ZSCORE	NUMBER

DE_SUBJECT_RBM_DATA	
TRIAL_NAME	VARCHAR2(15)
ANTIGEN_NAME	VARCHAR2(100)
N_VALUE	NUMBER
PATIENT_ID	NUMBER(38,0)
GENE_SYMBOL	VARCHAR2(100)
GENE_ID	NUMBER(10,0)
ASSAY_ID	NUMBER
NORMALIZED_VALUE	NUMBER(18,5)
CONCEPT_CD	NVARCHAR2(100)
TIMEPOINT	VARCHAR2(100)
DATA_UID	VARCHAR2(100)
VALUE	NUMBER
LOG_INTENSITY	NUMBER
MEAN_INTENSITY	NUMBER
STDDEV_INTENSITY	NUMBER
MEDIAN_INTENSITY	NUMBER
ZSCORE	NUMBER

DE_SUBJECT_PROTEIN_DATA	
TRIAL_NAME	VARCHAR2(15)
COMPONENT	VARCHAR2(15)
INTENSITY	NUMBER
PATIENT_ID	NUMBER(38,0)
SUBJECT_ID	VARCHAR2(10)
GENE_SYMBOL	VARCHAR2(100)
GENE_ID	NUMBER(10,0)
ASSAY_ID	NUMBER
TIMEPOINT	VARCHAR2(20)
N_VALUE	NUMBER
MEAN_INTENSITY	NUMBER
STDDEV_INTENSITY	NUMBER
MEDIAN_INTENSITY	NUMBER
ZSCORE	NUMBER

HAPLOVIEW_DATA	
I2B2_ID	NUMBER(20,0)
JNJ_ID	VARCHAR2(20)
FATHER_ID	NUMBER(5,0)
MOTHER_ID	NUMBER(5,0)
SEX	NUMBER(1,0)
AFFECTION_STATUS	NUMBER(1,0)
CHROMOSOME	VARCHAR2(10)
GENE	VARCHAR2(50)
RELEASE	NUMBER(4,0)
RELEASE_DATE	DATE
TRIAL_NAME	VARCHAR2(50)
SNP_DATA	CLOB

SNP_INFO	
SNP	VARCHAR2(50)
GENE	VARCHAR2(50)
POSITION	NUMBER(20,0)
CHROMOSOME	VARCHAR2(6)
REFSNP	VARCHAR2(20)
STRAND	VARCHAR2(10)
SNP_INDX	NUMBER(4,0)

DE_SUBJECT_SAMPLE_MAPPING	
PATIENT_ID	NUMBER(18,0)
SITE_ID	NVARCHAR2(100)
SUBJECT_ID	NVARCHAR2(100)
SUBJECT_TYPE	NVARCHAR2(100)
CONCEPT_CODE	VARCHAR2(1000 CHAR)
ASSAY_ID	NUMBER(18,0)
PATIENT_UID	VARCHAR2(50)
SAMPLE_TYPE	VARCHAR2(100)
ASSAY_UID	NVARCHAR2(100)
TRIAL_NAME	VARCHAR2(30)
TIMEPOINT	VARCHAR2(100)
TIMEPOINT_CD	VARCHAR2(50)
SAMPLE_TYPE_CD	VARCHAR2(50)
TISSUE_TYPE_CD	VARCHAR2(50)
PLATFORM	VARCHAR2(50)
PLATFORM_CD	VARCHAR2(50)
TISSUE_TYPE	VARCHAR2(20)
DATA_UID	VARCHAR2(100)
GPL_ID	VARCHAR2(20)

DE_MRNA_ANNOTATION	
GPL_ID	VARCHAR2(100)
PROBE_ID	VARCHAR2(100)
GENE_SYMBOL	VARCHAR2(100)
GENE_ID	VARCHAR2(100)
PROBESET_ID	NUMBER(38,0)

DEAPP_ANNOTATION	
ANNOTATION_TYPE	VARCHAR2(50)
ANNOTATION_VALUE	VARCHAR2(100)
GENE_ID	NUMBER
GENE_SYMBOL	VARCHAR2(200)

DEAPP_PROBESET	
PROBESET_ID	NUMBER(38,0)
PROBESET	VARCHAR2(100)
PLATFORM	VARCHAR2(100)

DE_XTRIAL_CHILD_MAP		
CONCEPT_CD	VARCHAR2(50)	<pk>
PARENT_CD	NUMBER	<fk>
MANUALLY_MAPPED	NUMBER	
STUDY_ID	VARCHAR2(50)	

DE_XTRIAL_PARENT_NAMES		
PARENT_CD	NUMBER	<pk>
ACROSS_PATH	VARCHAR2(500)	<ak>
MANUALLY_CREATED	NUMBER	

DE_SAVED_COMPARISON	
COMPARISON_ID	NUMBER
QUERY_ID1	NUMBER
QUERY_ID2	NUMBER