BRIDG HCT Physical Model – A Model Driven Architecture for Stem Cell Data Exchange

Shyam Sundar Somasundaram
MS – Computer Science
University of Minnesota, Twin Cities

Advisor: Dr. John Carlis

(i) Abstract

A Physical Data Model (PDM) [1] provides a visual representation of the structure of a database design. It highlights the required tables, how they are related to each other and includes artifacts such as indexes, constraint definitions of a database management system. PDMs find application in an industrial setting when relational database management systems are used to store data. Project teams create PDMs to model their database schema and derive the database schema automatically from the model.

The Biomedical Research Integrated Domain Group (BRIDG) HCT Physical Model is one such application. The BRIDG model [2] is a structured information model that represents a shared view of the concepts of protocol-driven clinical research. It is used to support development of data interchange standards and technology solutions that will enable semantic interoperability within the biomedical/clinical research arena and between research and the healthcare arena. This project aims at creating a physical model from the Hematopoietic Stem Cell Transplant (HCT) domain of the BRIDG Model that will help remove barriers that the transplant centers experience while transferring HCT data electronically. It provides a foundation for transplant centers to develop their own data systems and facilitate the process of submission of transplant data to The Center for International Blood and Marrow Transplant Research (CIBMTR) [3]. CIBMTR is charged with collecting data on all allogeneic (related and unrelated) hematopoietic cell transplantations performed in the United States, and on all HCTs done with products procured through the Be The Match Program but performed outside of the United States.

The physical model created supports and works well with the existent Stem Cell Therapeutic Outcomes Database (SCTOD). This database allows analysis of Program use, center - specific outcomes, size of donor registry, cord blood inventory and patient access to HCT. It is well documented and adheres to industry standards on Unified Modeling Language (UML) and Physical Modeling Practices thus allowing easier usage and implementation by external parties and better maintenance by CIBMTR.

The implementation of the project began by first understanding and analyzing the BRIDG model and its contents. Instance diagrams were created for each mapping path that provided a visual representation of the interaction between entities. The entities relevant to the HCT domain were identified and a Conceptual Data Model (CDM) was created using these entities. The CDM was then converted into a Logical Data Model (LDM) that handled the complex data types, different types of relationships, and referential integrity constraints. A Physical Data Model (PDM) was created from the LDM to facilitate the generation of Data Definition Language (DDL) and the necessary abbreviations were applied to make the model American National Standards Institute (ANSI) compliant. All design decisions made throughout the process were documented with proper justification.

(ii) Table of Contents

(iii) List of Abbreviations:

ABBREVIATION	DESCRIPTION
ANSI	American National Standards Institute
BRIDG	Biomedical Research Integrated Domain Group
caDSR	Cancer Data Standards Registry
CDE	Common Data Element
CDISC	Clinical Data Interchange Standards Consortium
CDM	Conceptual Data Model
CIBMTR	Center for International Blood and Marrow Transplant Research
DAM	Domain Analysis Model
DDL	Data Definition Language
DOC	Degree of Complexity
EA	Enterprise Architect
FDA	Food and Drug Administration
HCT	Hematopoietic Stem Cell Transplant
HL7	Health Level Seven International
HLA	Human Leukocyte Antigens
HRSA	Health Resources and Services Administration
LDM	Logical Data Model
LSDAM	Life Sciences Domain Analysis Model
NCI	National Cancer Institute
NMDP	National Marrow Donor Program
NOR	Number of Relationships
OMG	Object Management Group
OOM	Object Oriented Model
OOP	Object Oriented Programming
PD	Power Designer
PDM	Physical Data Model
RCRIM	Regulated Clinical Research Information Management Technical Committee
SCC	Semantic Coordination Committee
SCTOD	Stem Cell Therapeutic Outcomes Database
SME	Subject Matter Experts
UML	Unified Modeling Language

(iv) List of Figures:

1. Introduction

1.1 Stem Cell Therapeutic Outcomes Database

The Center for International Blood and Marrow Transplant Research [3] (CIBMTR) is a combined research program of the National Marrow Donor Program and the Medical College of Wisconsin which facilitates critical research on transplantation outcomes that leads to increased survival and an enriched quality of life for thousands of patients through collaborations with the global scientific community to advance hematopoietic cell transplantation and cellular therapy research worldwide.

CIBMTR holds the contract for the Stem Cell Therapeutic Outcomes Database (SCTOD), awarded by the Health Resources and Services Administration (HRSA) of the U.S. Department of Health and Human Services. As the contract holder, the CIBMTR is charged with collecting data on all allogeneic (related and unrelated) hematopoietic stem cell transplantations (HCTs) performed in the United States, and on all HCTs done with products procured through the Program but performed outside of the United States. The SCTOD database allows analysis of Program use, center- specific outcomes, size of donor registry and cord blood inventory, and patient access to HCT.

Hematopoietic Stem Cell Transplantation [4] is the transplantation of hematopoietic stem cells usually derived from the bone marrow, peripheral blood cells or the umbilical cord blood. It is autologous if the patients own cells are used or allogeneic if the stem cells come from a donor.

CIBMTR collects the outcomes data of the transplantation for research and analysis purposes. The analysis has helped for faster and improved matching to find the best donor and consequently earlier transplantation has led to better outcomes.

Because of the collaboration of physicians and data managers at transplant centers, the field of transplantation already had in place a tremendous voluntary data submission mechanism. The SCTOD builds upon the existing CIBMTR infrastructure and expertise for data collection, management and analysis.

1.2 Existing Clinical Domain Models

The Biomedical Research Integrated Domain Group (BRIDG) [2] Model is a collaborative effort engaging stakeholders from the Clinical Data Interchange Standards Consortium (CDISC), Health Level Seven International (HL7), the US National Cancer Institute (NCI), and the US Food and Drug Administration (FDA). The BRIDG Model is an information model representing a shared view of the concepts of protocol-driven clinical research. This structured information model is being used to support development of data interchange standards and technology solutions that will enable semantic (meaning-based) interoperability within the biomedical/clinical research arena and between research and the healthcare arena.

The Life Sciences Domain Analysis Model (LSDAM) [5] provides a framework to support biomedical research; it produces a shared view of static data and semantics of life

sciences domain. The LSDAM is aligned, where appropriate, with the BRIDG model. The points of alignment highlight the touch points between life sciences and clinical research, and through these touch points, a combined use of the two models could support semantic interoperability across the translational research continuum.

For instance, the "PerformedHistopathologyResult" belongs to Life Sciences Domain and talks about tissue abnormalities. It is not directly related to HSCT domain, but could cause certain types of breast cancer and hence is aligned with the BRIDG.

The LSDAM model was developed as a peer to the BRIDG model from its inception. BRIDG model 4.0, the latest version that was released a few days includes the integration of LSDAM. The semantics currently represented in the LSDAM will be represented within the BRIDG model and will be managed by the BRIDG Semantic Coordination Committee (SCC).

The Bioinformatics Research Department at the NMDP is working closely with the CIBMTR to build on the existing BRIDG model, design and implement the architecture to use the model for storing and moving data from the SCTOD. The SCTOD is a much larger database containing outcomes data extracted by biostatisticians that might not be or will never be in the BRIDG model. Hence NMDP and CIBMTR need to maintain both separately.

2. Motivation

This project gives me an opportunity to work on an impactful problem that directly contributes to the society. An interdisciplinary project such as the BRIDG project not only allows me to apply my computer skills in an industrial setting but also explores certain aspects of the Bioinformatics domain like Human Leukocyte Antigens (HLA) typing and their role in organ transplantation. Working on this project helps me understand the good work done by organizations such as NMDP and CIBMTR and gives me immense satisfaction of being a part of it. Lastly, having aspirations to become a Data Architect in the future, being successful in this project would be like taking the first step in that direction.

3. Connection with Gene Ontology

Gene ontology [6] is a significant effort made by the bioinformatics community to unify the representation of gene and gene product attributes. It is an ongoing initiative to develop and maintain a controlled vocabulary of gene and gene products. Both gene ontology and BRIDG model are similar in the fact that they attempt to establish a common vocabulary system to facilitate a shared understanding of concepts within a domain.

4. Problem Statement

To propose a new architecture to use the existing BRIDG model and create a physical database model that

- Supports and works in compliance with the SCTOD such that any patient/donor data inserted into the physical model can also be moved into the SCTOD without any modification.
- Can be easily maintained by CIBMTR such that any patient/donor data sent by any transplant center fits into the model.
- Can be easily used and implemented by external parties such that they can use the model as a basis and develop their own data systems.
- Is well documented and adheres to industry standards on UML and Physical Modeling Practices

5. Implementation

5.1 Input

5.1.1 BRIDG Model v3.2

The Object Management Group [7] (OMG) is an international technology standards consortium. OMG's modeling standards include the Unified Modeling Language (UML) and enable powerful visual design, execution and maintenance of software and other processes.

A UML [8] model is a general-purpose model that provides a visual representation of the structure of the system, the objects in the system and their interaction. UML consists of structural diagrams that are static and represent the structure of the system and behavioral diagrams that are dynamic and describes the behavior of the system.

The BRIDG model contains static data semantics of the protocol driven clinical research domain. Hence most of its semantics are represented as UML class diagrams, which are structural diagrams.

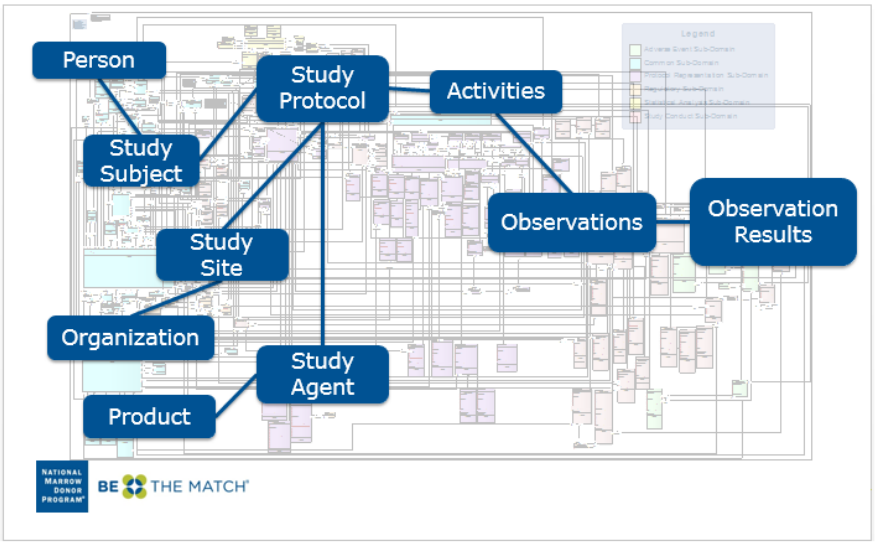


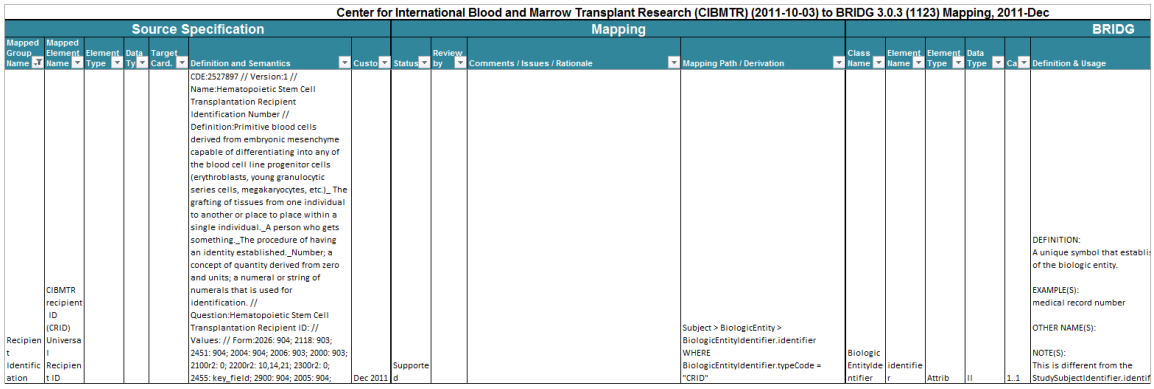
Fig 1. BRIDG Model v3.2 [Courtesy of NMDP]

5.1.2 BRIDG Mapping Spreadsheet

A Common Data Element (CDE) is a representation of a data point collected on CIBMTR mandated forms. Each CDE describes the entity in focus and its relation with other entities required to represent the data point. All CDEs collected from the CIBMTR forms are registered in the National Cancer Institute's (NCI) Cancer Data Standards Registry (caDSR).

The caDSR [9] contains information describing data elements and information models known as metadata. Along with its associated applications, the caDSR supports the research community by providing means to manage detailed description of data held in publically accessible data sets and shared information models.

Each row in the BRIDG mapping spreadsheet represents a CDE. A CDE consists of a unique number, definition, domain, class, element, data type, comments and a mapping path. The mapping path describes the entity and its interaction with other required entities to represent the data point collected from the form.



Center for International Blood and Marrow Transplant Research (CIBMTR) (2011-10-03) to BRIDG 3.0.3 (1123) Mapping, 2011-Dec

Source Specification							Mapping				BRIDG					
Mapped Group Name	Mapped Element Name	Element Type	Data Ty	Target Card.	Definition and Semantics	Custo	Status	Review by	Comments / Issues / Rationale	Mapping Path / Derivation	Class Name	Element Name	Element Type	Data Type	Ca	Definition & Usage
Recipient Identification	CIBMTR recipient ID (CRID) Universal Recipient ID				CDE:2527897 // Version:1 // Name:Hematopoietic Stem Cell Transplantation Recipient Identification Number // Definition:Primitive blood cells derived from embryonic mesenchyme capable of differentiating into any of the blood cell line progenitor cells (erythroblasts, young granulocytic series cells, megakaryocytes, etc.)_ The grafting of tissues from one individual to another or place to place within a single individual._A person who gets something._The procedure of having an identity established._Number; a concept of quantity derived from zero and units; a numeral or string of numerals that is used for identification. // Question:Hematopoietic Stem Cell Transplantation Recipient ID: // Values: // Form:2026: 904; 2118: 903; 2451: 904; 2004: 904; 2006: 903; 2000: 903; 2100r2: 0; 2200r2: 10,14,21; 2300r2: 0; 2455: key_field; 2900: 904; 2005: 904;	Dec 2011	Supported			Subject > BiologicEntity > BiologicEntityIdentifier.identifier WHERE BiologicEntityIdentifier.typeCode = "CRID"	BiologicEntityIdentifier	identifier	Attrib	II	1..1	DEFINITION: A unique symbol that establi of the biologic entity. EXAMPLE(S): medical record number OTHER NAME(S): NOTE(S): This is different from the StudySubjectIdentifier.identif

Fig 2. BRIDG Mapping spreadsheet

Since the project started out as a collection of CDEs, it was essential to map these CDEs to the BRIDG model. The mapping of CDEs was done through the mapping spreadsheet. The mapping spreadsheet also contained some useful information about each CDE like the form in which the CDE is used, the question in the form it answered, the class name, the mapping path, etc. Consequently it proved to be very useful in the creation of instance diagrams for each CDE from its mapping path.

5.2 Instance Diagrams

An instance diagram [10] provides a point in time representation of a state of a system. Since in Object Oriented Programming (OOP) paradigm, an object represents an instance of a class, instance diagrams are sometimes referred to as object diagrams as they represent specific instances of a class. It is a static diagram that helps to visualize the structure of the system at a particular time and highlights a particular group of classes, their attributes and how they interact with each other.

Each CDE in the mapping spreadsheet is associated with an instance diagram. Instance diagrams are preferred to other diagrams due to their simplicity and ease of construction. They provide a visual representation of the mapping paths and hence are easier to read, interpret and analyze. In order to reduce complexity of mapping paths, some implicit classes are omitted from them. These classes, which are required to explain certain concepts in BRIDG, are captured in instance diagrams.

Instance diagrams were created using Sparx Enterprise Architect (EA) v9.3 tool which is the default tool used to navigate through the BRIDG model. Hence creating the instance diagrams using EA allowed easier incorporation into the model as opposed to creating them in another tool and transferring them to EA using import/export operations.

CDE	Mapping Path
2866037	PerformedSubstanceAdministration.dateRange WHERE PerformedProcedure.nameCode = Administer Hematopoetic Stem Cell Product {Transplantation}

Fig 3. Sample CDE and its mapping path

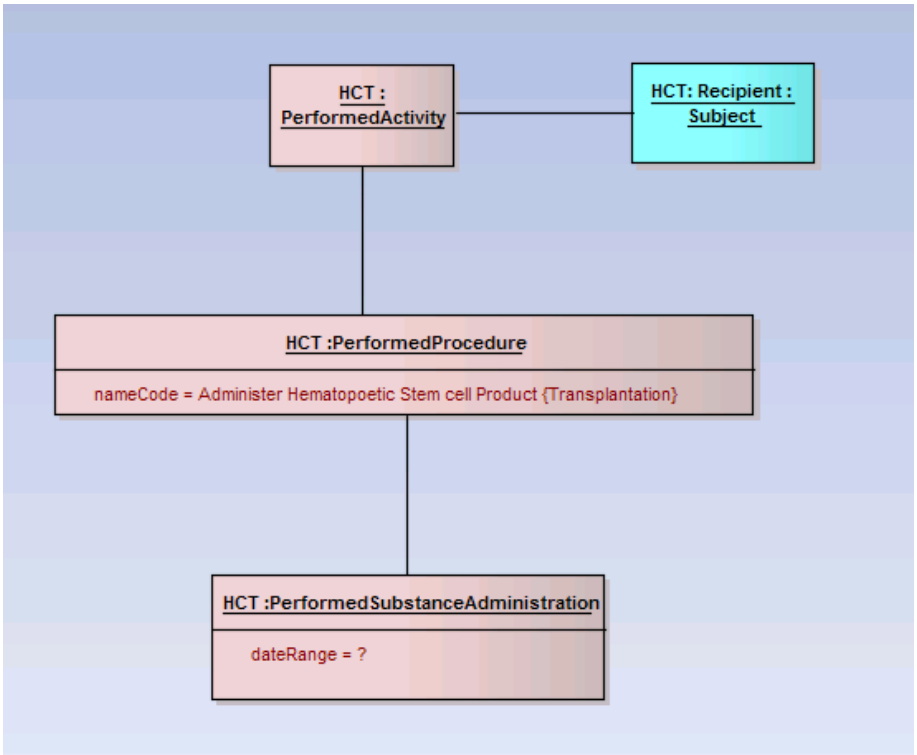


Fig 4. Sample instance diagram corresponding to CDE shown in Fig 1.

5.2.1 Managing Complexity

5.2.1.1 Degree of Complexity (DOC)

The complexity of instance diagrams is measured using a metric called Degree of Complexity (DOC). The DOC is defined as the number of unique classes present in the mapping path of a CDE. The minimum value of DOC is 1 and the current maximum value is 25. The DOC metric helps to divide the work across sprints in an agile/scrum development framework.

Instance diagrams for lower DOCs are pretty straightforward to create, as they don't involve many implicit classes. But higher DOCs involve more classes and consequently more implicit classes and relationships leading to greater complexity. However many but not all instance diagrams with higher DOCs need to be created from scratch and can be created by putting together pieces of instance diagrams of lower DOCs.

CDE	Mapping Path
2866037	PerformedSubstanceAdministration.dateRange WHERE PerformedProcedure.nameCode = Administer Hematopoetic Stem Cell Product {Transplantation}

Fig 5. CDE described in Fig 1 with DOC = 2

5.2.1.2 Number of Relationships (NOR)

Yet another metric to manage the complexity of instance diagrams is the number of relationships. The number of relationships of a CDE is defined as the sum of the DOC and the number of relationships of each class in the mapping path.

The mapping path is sometimes broken or does not contain all necessary information to explain the CDE for simplicity purposes. In such cases NOR gives a better estimate of how complex the diagram is going to be compared to the DOC which could be misleading sometimes. For example the CDE: 2770870 with mapping path Organization.name has DOC = 1 which implies low complexity. However its NOR is 43 since the Organization class has many relationships. This means that if we want to explain the complete scenario where a patient/donor, receives/donates in a transplant center (organization), then the diagram involves many implicit classes and relationships leading to much higher complexity.

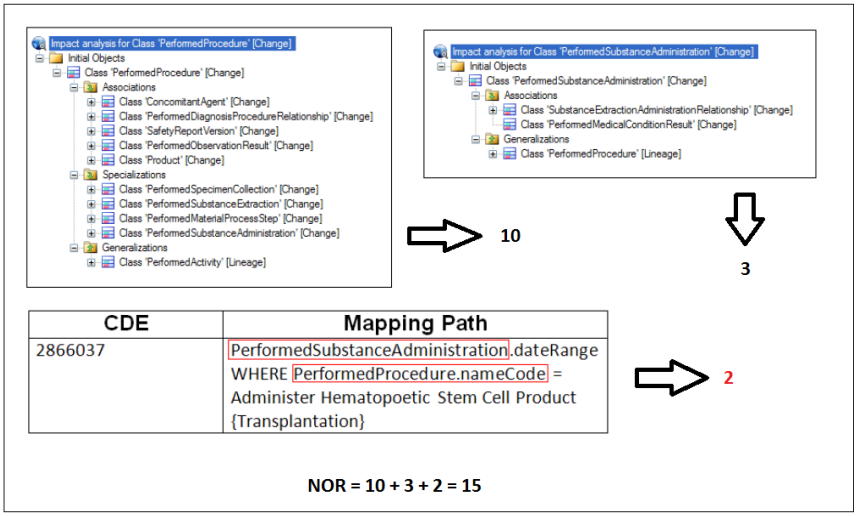


Fig 6. Number of relationships for CDE described in Fig 1 with DOC = 2

The number of relationships calculation is automated using a script that takes as input the mapping path of a CDE and generates the number of relationships of that CDE as output. Python v2.7.8 was used to generate the script.

1. Read the cde number, DOC and mapping path line by line from an input file.

2. For each mapping path:
    - 2.1. Extract the unique classes present in the mapping path.
    - 2.2 Get the number of relationships for each extracted class present in a separate excel spreadsheet
    - 2.3 Calculate the sum of the number of relationships of each extracted class.
    - 2.4 Add the DOC of the CDE to the sum calculated in 2.3

3. Output the final sum computed in 2.4 as the NOR of the CDE

Fig 7. Pseudocode of python script

❑ Input: Mapping Path

PerformedSubstanceAdministration.dateRange WHERE PerformedProcedure.nameCode = "Administer Hematopoetic Stem cell Product {Transplantation}“

❑ Output: Number of Relationships

['PerformedProcedure', 'PerformedSubstanceAdministration']
Row:151 CDE:2866037 DOC:2 Num_of_Rel:15

Fig 8. Working of the python script on the CDE described in Fig 1

5.2.2 Extracting Required Classes/Attributes

All classes, including implicit classes that are used in the instance diagrams to explain the mapping paths are included in the conceptual model. All attributes of the extracted classes are required and included in the conceptual model.

5.3 Tool Transformation

The required entities from the EA model are moved into Sybase Power Designer (PD) using import/export option. The models created by both tools are stored in XML Metadata Interchange (XMI) format. Hence the models can be transferred between the two tools using this format.

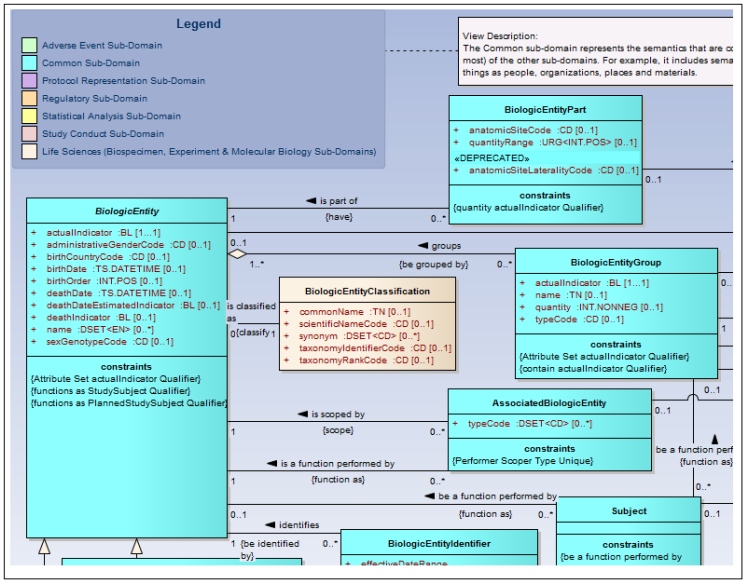


Fig 9. EA Model

Power Designer is NMDP's standard technology for modeling and hence it is essential to make this transformation. PD also used provides a clear distinction between Generalization and Specialization relationships making the modeling process easier.

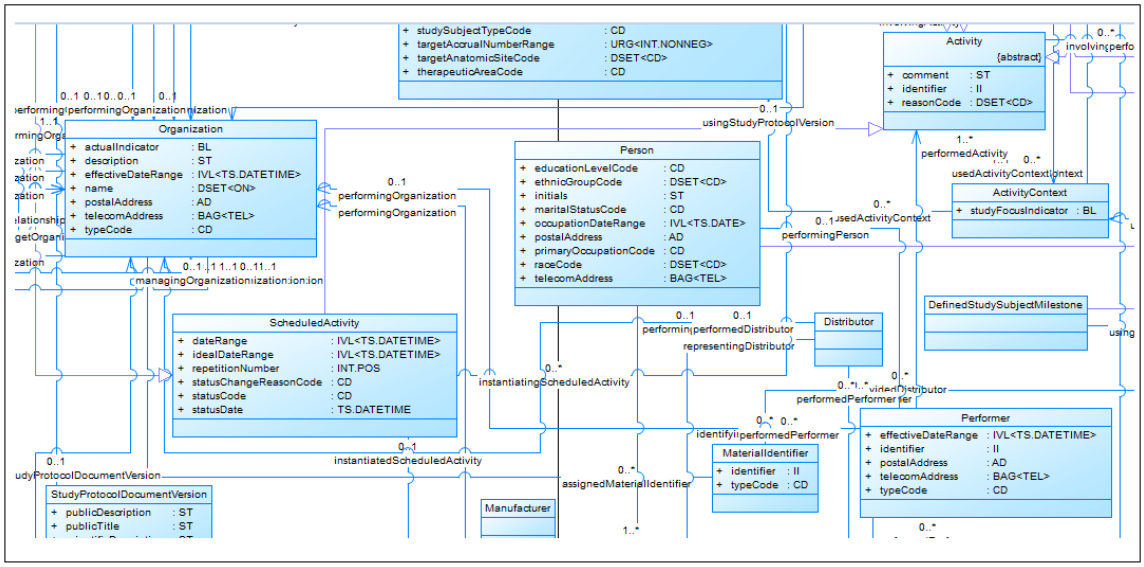


Fig 10. PD Model

5.3.1 Generalization/Specialization relationships

Inheritance is the process by which a class extracts attributes and methods from another class. The class from which the attributes and methods are extracted is known as the superclass and the class, which extracts the attributes and methods, is known as the subclass. In common terms, the superclass is said to be a generalization of the subclass and the subclass is said to be a specialization of the superclass. Eg. The "Activity" class is a generalization of the "PerformedActivity" class and the "PerformedActivity" class is a specialization of the "Activity" class.

In the EA model, there is no clear distinction between a generalization and a specialization relationship. All generalization/specialization relationships are represented as Generalization. Hence it is difficult to know if the target class is a parent or a child of the focal class. Power Designer however provides a clear distinction between the relationships and provides better readability to the user.

Relationships

Element	Element Stereotype	Type	Connection	Stereotype
ActivityContext		Class	Association	
DefinedCompositionRelationship		Class	Association	
DefinedContingentOnRelationship		Class	Association	
DefinedObservation		Class	Association	
Activity		Class	Generalization	
DefinedAdministrativeActivity		Class	Generalization	
DefinedObservation		Class	Generalization	
DefinedProcedure		Class	Generalization	
DefinedSubjectActivityGroup		Class	Generalization	

Fig 11. Relationships in EA Model

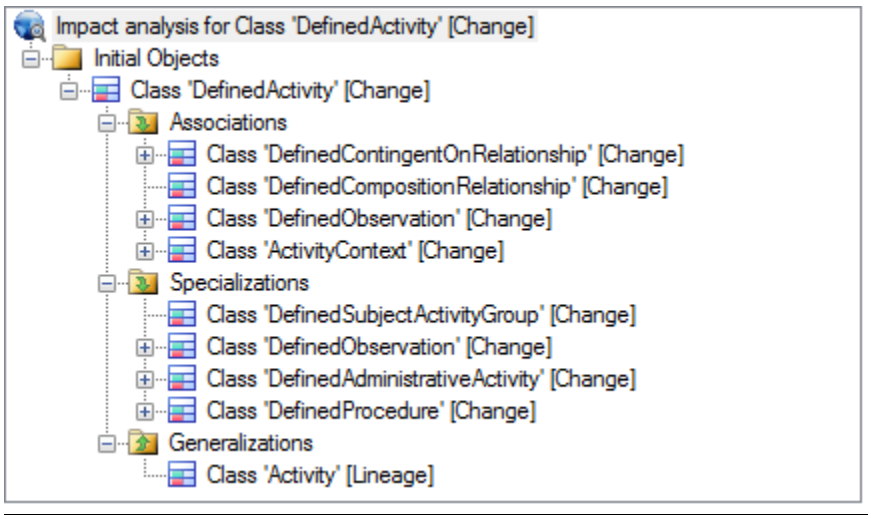


Fig 12. Relationships PD Model

5.3.2 Missing relationships

During transformation from one model to another, a few relationships are lost due to the order of extraction of entities. Since the entities were extracted sub-domain by sub-domain, all relationships of entities outside its sub-domain were lost. This is a limitation of the EA tool. For example the "DefinedActivity" class has relationships only with other entties from its sub-domain (Protocol Representation Sub – Domain) like "DefinedObservation" and "DefinedProcedure" and loses its relationship with "Activity" and "ActivityContext" classes that belong to Common Sub – Domain.

Relationships

Element	Element St...	Type	Connection	Stereotype
Activity		Class	Generaliza...	
ActivityContext		Class	Association	
DefinedAdministrativeActivity		Class	Generaliza...	
DefinedCompositionRelationship		Class	Association	
DefinedContingentOnRelationship		Class	Association	
DefinedObservation		Class	Association	
DefinedProcedure		Class	Generaliza...	
DefinedSubjectActivityGroup		Class	Generaliza...	

Fig 13. All relationships of DefinedActivity class

Relationships

Element	Element St...	Type	Connection	Stereotype
DefinedAdministrativeActivity		Class	Generaliza...	
DefinedCompositionRelationship		Class	Association	
DefinedContingentOnRelationship		Class	Association	
DefinedObservation		Class	Association	
DefinedProcedure		Class	Generaliza...	
DefinedSubjectActivityGroup		Class	Generaliza...	

Fig 14. Relationships of DefinedActivity class after extraction

The missing relationships were identified and added into the PD model as new relationships as it is essential to preserve any interactions between classes and consequently foreign key relationships. An excel sheet was used to keep track of the number of relationships for each class. This excel sheet was very useful in identifying the missing relationships by counting the number of relationships for each class in the EA model and the PD model and comparing them.

5.3.3 Transferring Diagrams

Diagrams could not be transferred from the EA model to the PD model. This is because each class has a corresponding Globally Unique Identifier (GUID) associated with it. While creating diagrams, every class is identified with its GUID. EA and PD use their own GUIDs to identify their classes. Hence the GUID information is not preserved during the tool transformation.

Due to timing constraints, the task of automation of matching the GUIDs of classes in EA with the GUIDs of classes in PD was not carried out. The comprehensive diagram was recreated in PD because it is essential to know all the classes present in the model and how they interact with each other.

5.3.4 Transferring Metadata

While transferring the classes and attributes from EA to PD, the metadata is not transferred and this has to be automated using code. The metadata consists of definitions, usage and comments about the classes and attributes. This metadata is pushed down to logical and physical models and it provides useful information about the class and attributes to a user who is using the model for the first time. Also in order to make the table and column names ANSI compliant, we had to preserve the original names and this was stored in the metadata.

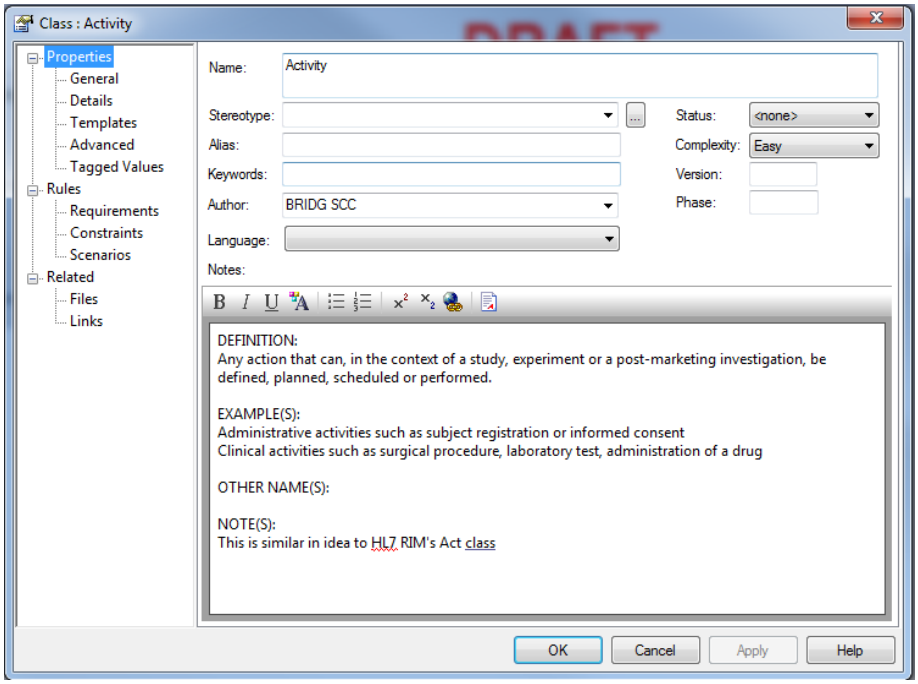


Fig 15. Activity class in EA model containing metadata

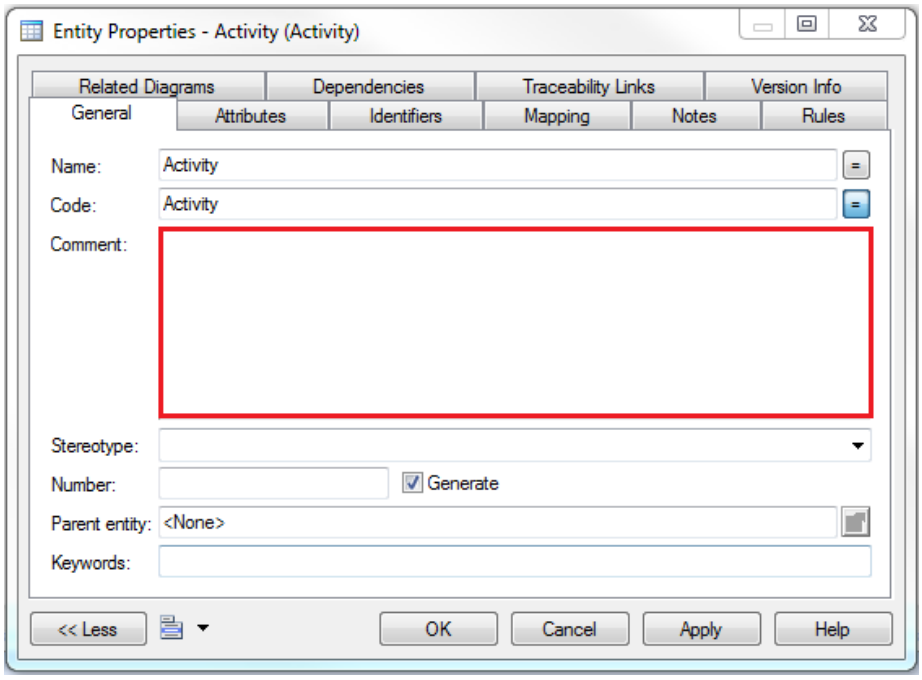


Fig 16. Activity class in PD model without metadata highlighted by empty red rectangle

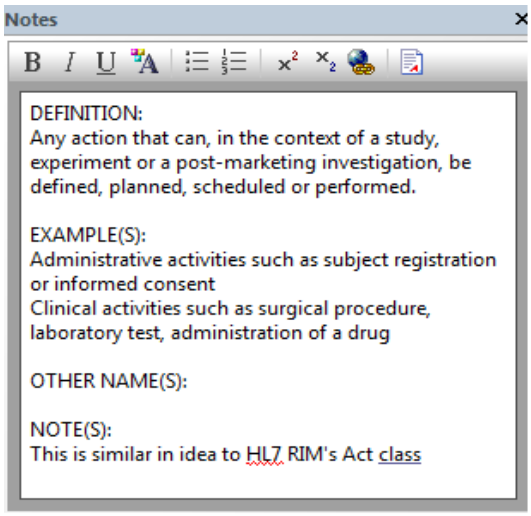


Fig 17. Metadata

An automation script developed using Python carried out the metadata transfer. The required metadata was first extracted from the xml file of the EA model. Some manipulations/modifications were performed on the extracted data to add some more data useful for analysis purposes like the list of CDEs in which each class and attribute occurs and the modified data were inserted into the xml file of the PD model.

Computing List of CDEs

1. Read the cde number and mapping path line by line from an input file.

2. For each mapping path:
    - 2.1. Extract the unique classes present in the mapping path.
    - 2.2. Extract the unique combination of class.attribute present in the mapping path.
    - 2.3. Read the unique class name and combination of class.attribute name from a file.
    - 2.3. Append the cde number to the end of the classname and combination of class.attribute name.
    - 2.4 Write it back to the file.

Fig 18. Pseudocode for computing the list of CDEs

Extracting Metadata and Moving into PD Model

1. Export the EA model as an XML file
2. Export the PD model as an XML file
3. Extract the necessary metadata from the EA model that are present between specific tags in the XML document.
4. Add the list of CDEs obtained to the extracted metadata.
5. Get the GUID of every class and combination of class.attribute from the PD model XML file.
6. Match the content from 4 to the corresponding GUID from 5.
7. Create a new XML <Comment> tag to add the metadata as a comment in the PD model and add each matched content from 6 as a separate comment.
8. Open Power Designer and import the modified PD XML file.

Fig 19. Pseudocode for extracting and moving metadata into PD model

5.4 Conceptual Data Model

A Conceptual Data Model (CDM) [11] is a high level description of the components of a database system. It includes only the main elements of the system, the relationships between them and abstracts the underlying details, which is essential to build the actual database system. The conceptual model helps Subject Matter Experts (SME) and business analysts in an organization to gain a quick understanding of the entire model usually within a week.

5.4.1 Inheritance Structure

The conceptual model is represented in the form of an inheritance structure for better visualization and analysis. The HCT domain of the BRIDG model consists of certain key classes. All of the other classes are either descendants or are directly related to these key classes. The key classes and their descendants are represented in RED and the associated classes are represented as ORANGE. For instance, the “Activity” class can be considered a key class and all activities that are performed, defined, planned or scheduled such as a diagnosis, observation, procedure, drug administration, etc. can be considered as a descendant of the “Activity” class.

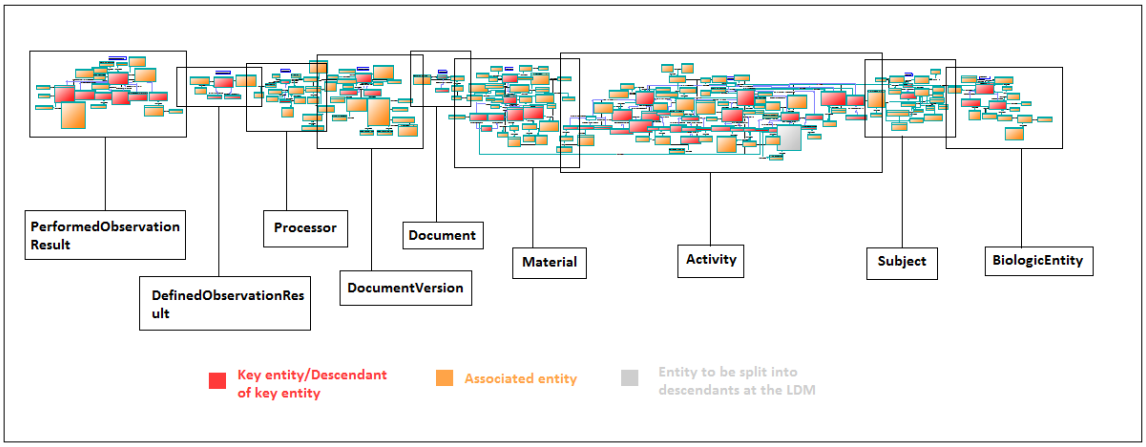


Fig 20. Conceptual Data Model (Inheritance Structure)

5.4.2 Handling Complex Data Types

The data types of attributes of classes in the BRIDG model are represented using Health Level Seven International [12] (HL7) standards. HL7 is an ANSI-accredited standards developing organization that provides a framework and related standards for the exchange, integration, sharing, and retrieval of electronic health information.

Some of these data types are complex and can't be mapped directly to an ANSI compliant data type. For instance, an attribute with the Interval (IVL) data type of HL7, that specifies the begin and end time of an activity is converted to two attributes, one denoting the begin time and the other denoting the end time. Each of the two new attributes has the simple "Timestamp" data type.

Fig 21. Handling the complex IVL data type

5.4.3 Creating Identifiers

All classes in the CDM will be converted to a table in the PDM. Some classes do not have an identifier or a primary key since they do not have a meaning as a stand-alone class but are used to complement other classes. Hence a default primary key is created for these classes. The name of the default primary key is given as "classnameID".

For instance the "DefinedImaging" class represents the activity of obtaining an image of the interior or the exterior of the body like X-ray, MRI scans, etc. However this activity is never performed alone but as part of an observation on the patient. Hence it does not have a primary key of its known and a "DefinedImagingID" primary key is created for it. This scenario also highlights one key difference between a conceptual model and a physical model. At the conceptual level the DefinedImaging entity does not have a unique identifier, but at the physical level, every table needs to have a primary key. Hence the primary key is created for before the transformation to the physical model occurs.

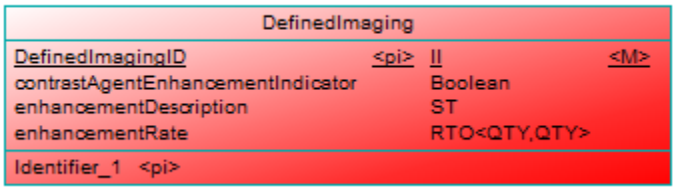


Fig 22. Creating primary key for DefinedImaging

5.4.3.1 Data type for Identifiers

For simplicity, in the first release of the BRIDG HCT model the data type of identifiers are set as integers. However this might not be a practical solution. Several organizations might have their own integers to uniquely identify a donor/patient. When a donor/patient data is transferred between transplant centers, the integer used as identifier by the sending organization might represent a different donor/patient in the receiving organization.

A better solution for the data types of identifiers is to use Globally Unique Identifiers (GUID) or Object Identifiers (OID). Handling data types for integers will be implemented in one of the future releases of BRIDG HCT model.

5.5 Logical Data Model

A Logical Data Model (LDM) [13] provides a detailed description about the components of the database system. It includes all entities, relationships between entities required by the database system. Each entity contains a primary key, all necessary attributes and the foreign keys required to identify relationships with other entities. Normalization is usually performed at the LDM.

A LDM is independent of technology and focuses on the business requirements highlighting what kind of data is stored by the organization. Consequently when an organization switches from one technology to another, the LDM allows for quick transition thus aiding in better maintenance and change management.

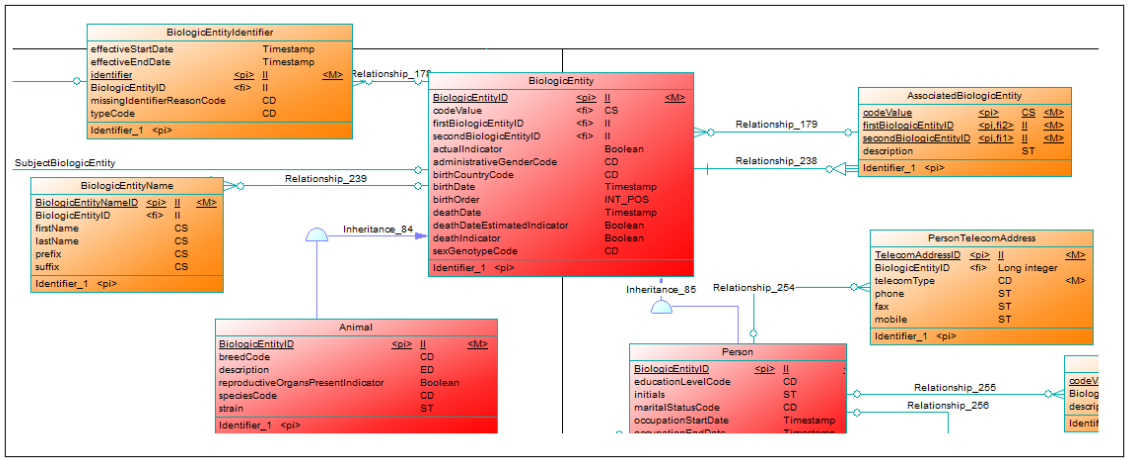


Fig 23. Sample LDM highlighting foreign key relationships

5.5.1 Handling Identifiers for Different Relationship Types

5.5.1.1 One - to - Many

The primary key of the entity with cardinality ONE is a foreign key to the entity with cardinality MANY. As an example, a place could have several identifiers. It could mean the license plate of an ambulance or the bed number in a hospital. Hence the primary key of the entity “Place” (PlaceID) is a foreign key to the entity “PlaceIdentifier”.

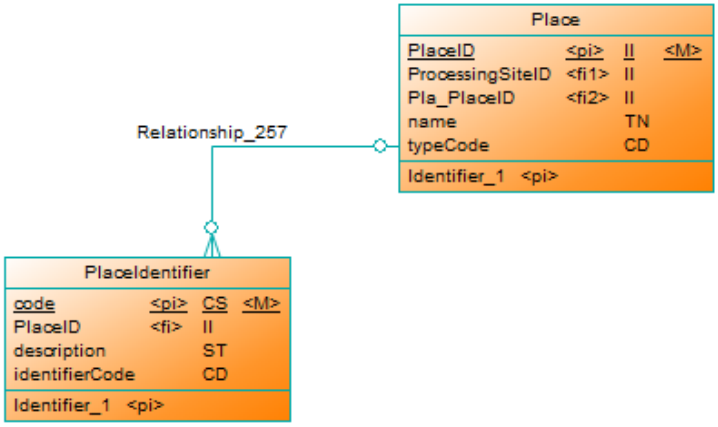


Fig 24. Handling One – to – Many relationship

5.5.1.2 Many - to - Many

A new entity is created with the combination of the primary keys of the participating entities as its primary key. As an example, "AssessedActivityRelationship" entity specifies a link between an assessment (PerformedObservation) and the activity (PerformedActivity) on which the observation is based on. Hence the primary key of AssessedActivityRelationship is a combination of PerformedActivityID and PerformedObservationID highlighting which assessment is based on which activity on which the assessment was performed.

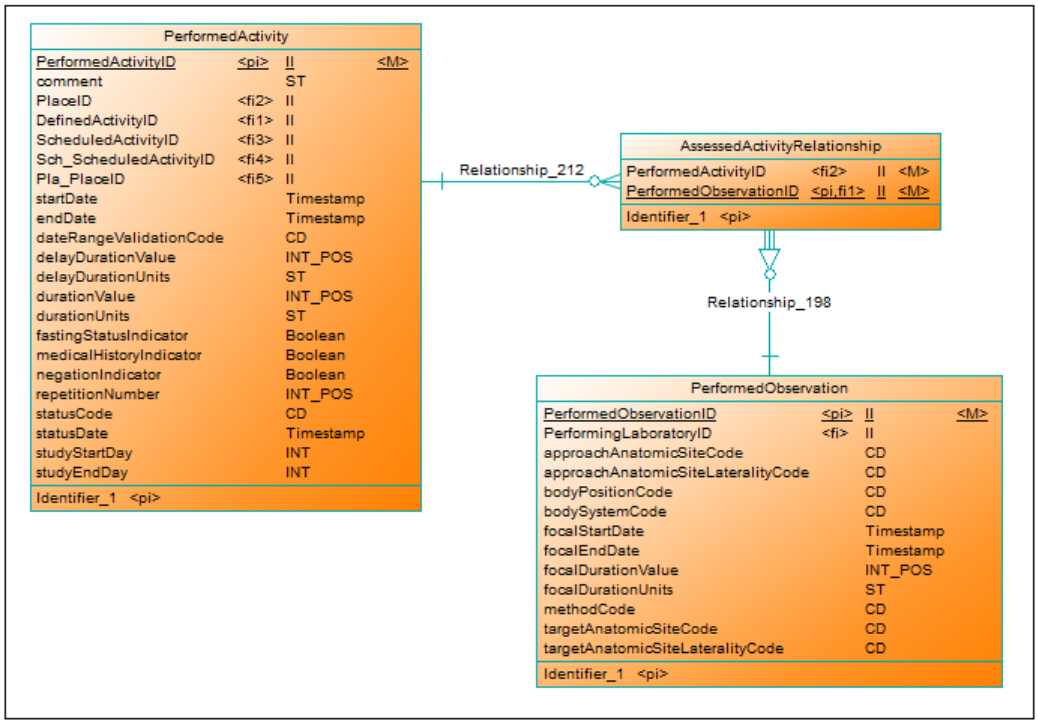


Fig 25. Handling Many – to – Many relationship

5.5.2 Creating Additional Entities

5.5.2.1 PerformedSubstanceAdministration Descendants

"PerformedSubstanceAdministration" describes the introduction of medications or other substances to a subject or experimental unit. However it is also essential to note at what point during the lifetime of the transplant the substance administration occurred and whether the substance administration process was a cellular therapy or transplantation. Hence the "PerformedSubstanceAdministration" class is broken down into 5 different entities denoting the time point at which the substance administration occurred. These 5 entities are "PerformedCellularTherapy", "PerformedTransplantation", "PerformedBeforeCoreActivity", "PerformedDuringCoreActivity" and "PerformedAfterCoreActivity". This information is useful for clinicians and biostatisticians who need to know when a certain drug was administered and the dosage.

As an example an anti – inflammatory drug like aspirin can be administered to the patient before the transplantation, during the transplantation and after the transplantation to reduce inflammation and swelling. Hence it is important for the clinicians to know what drug was administered and when in the lifecycle of the transplant was it administered to the patient.

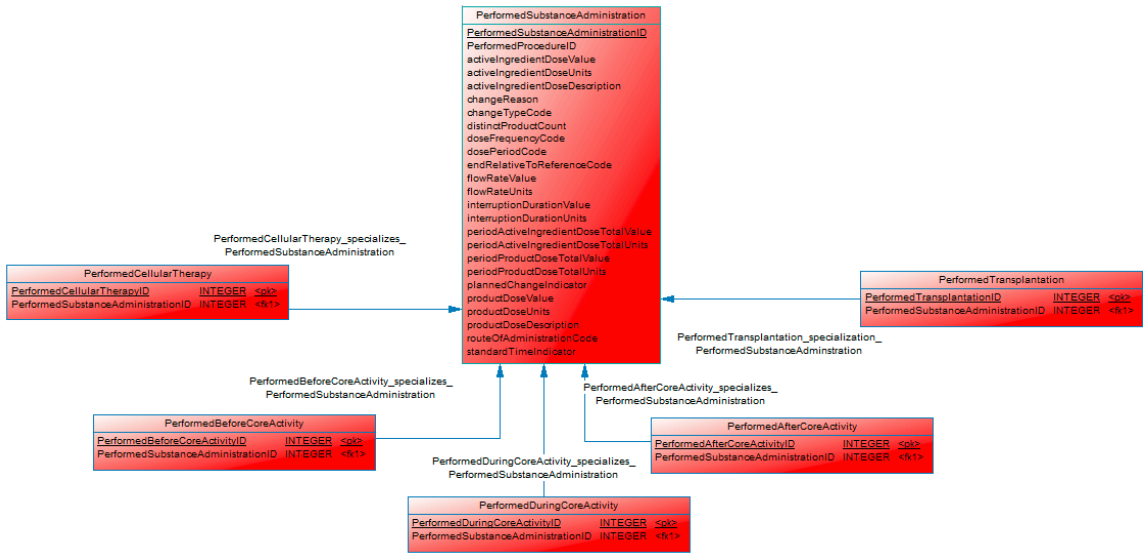


Fig 26. Descendants of PerformedSubstanceAdministration

5.5.2.2 BiologicDrug

The entity "Biologic" refers to substances obtained from living organisms or anything produced by the living organisms. The entity "Drug" refers to the substances produced by pharmaceutical companies to cure diseases. There are certain substances that are biotechnology derived pharmaceutical agents obtained from living organisms or their products. Such substances are described by a new entity called "BiologicDrug" which is a descendant of both the "Biologic" and the "Drug" entity.

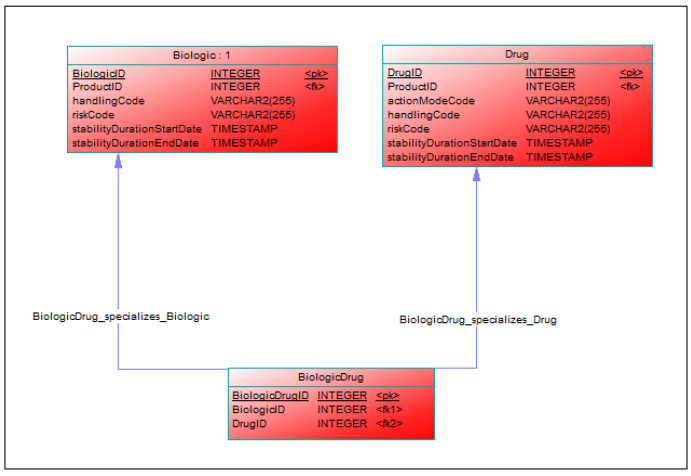


Fig 27. Creation of BiologicDrug entity

5.5.2.3 AdverseEventCausalRelationship

An adverse event might be caused by an observation result such as diagnosis of Diabetes or an action such as administration of cold medicine. If the adverse event is caused by an observation it is captured by the "EvaluatedResultRelationship" entity. Similarly if an adverse event is caused by an action it is captured by the "EvaluatedActivityRelationship" entity. However there arises an ambiguity when the cause of the adverse event is unknown.

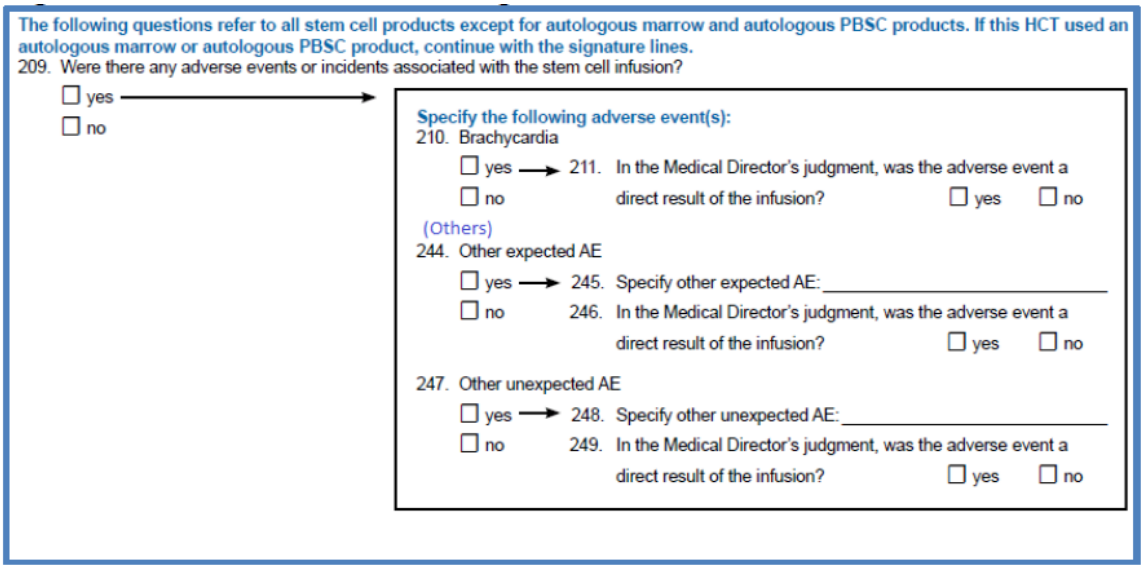


Fig 28. Form 2006 for capturing AdverseEvent information [Courtesy of NDMP]

The Form 2006 for capturing AdverseEvent information is shown in Fig. The options "Other Expected/Unexpected AE" causes the ambiguity. If the person filling out the form decides to leave the cause as blank, then person entering the data might not know if the cause was due to an activity or a result.

To handle this ambiguity, the "EvaluatedResultRelationship" and the "EvaluatedActivityRelationship" is combined into a single entity "AdverseEventCausalRelationship" with an extra attribute "typeCode" which indicates whether the cause of the adverse event is an activity or a result. This gives the user the flexibility to leave the "typeCode" as NULL when the cause is unknown but still enter the other details.

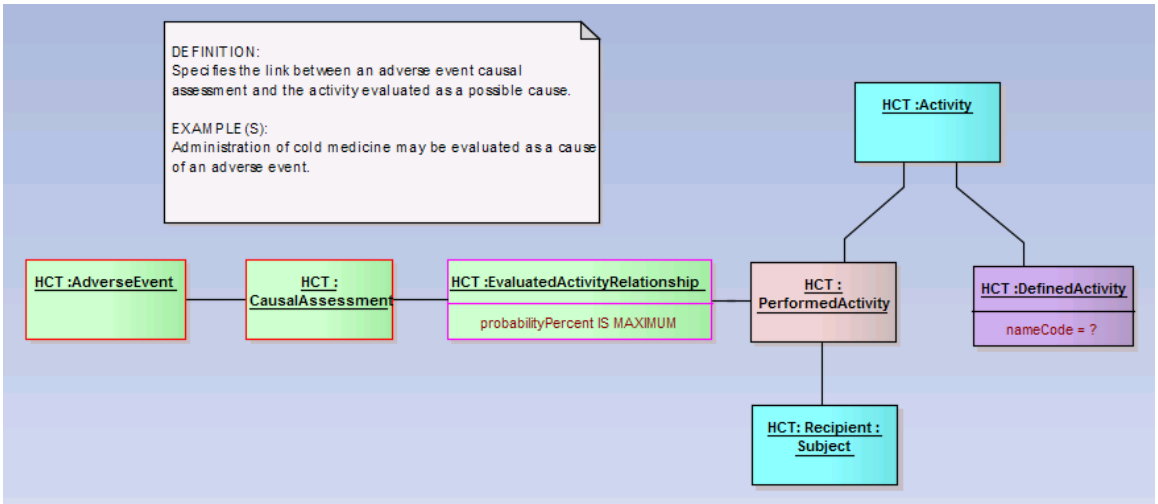


Fig 29. EvaluatedActivityRelationship entity

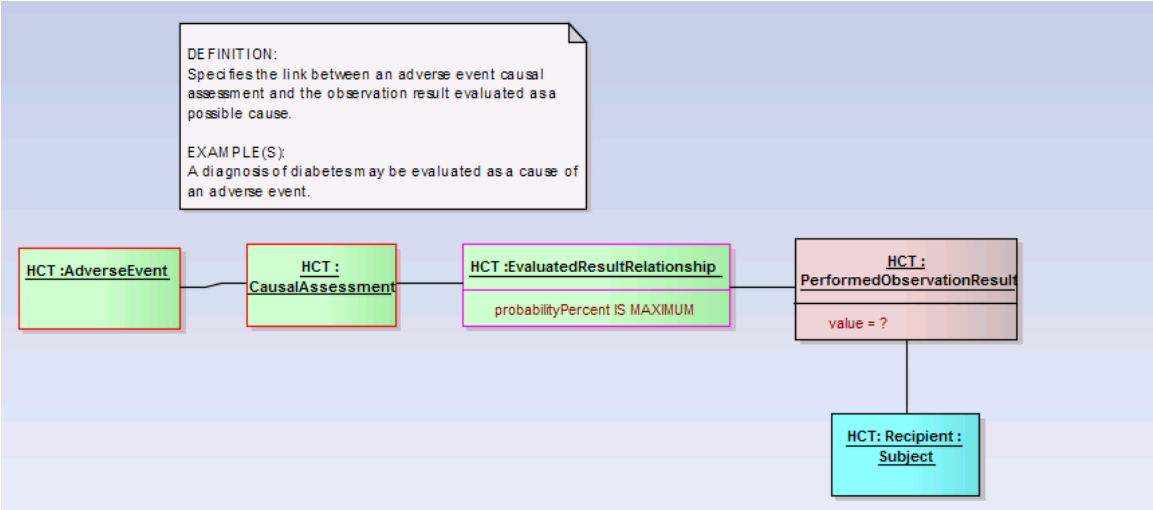


Fig 30. EvaluatedResultRelationship entity

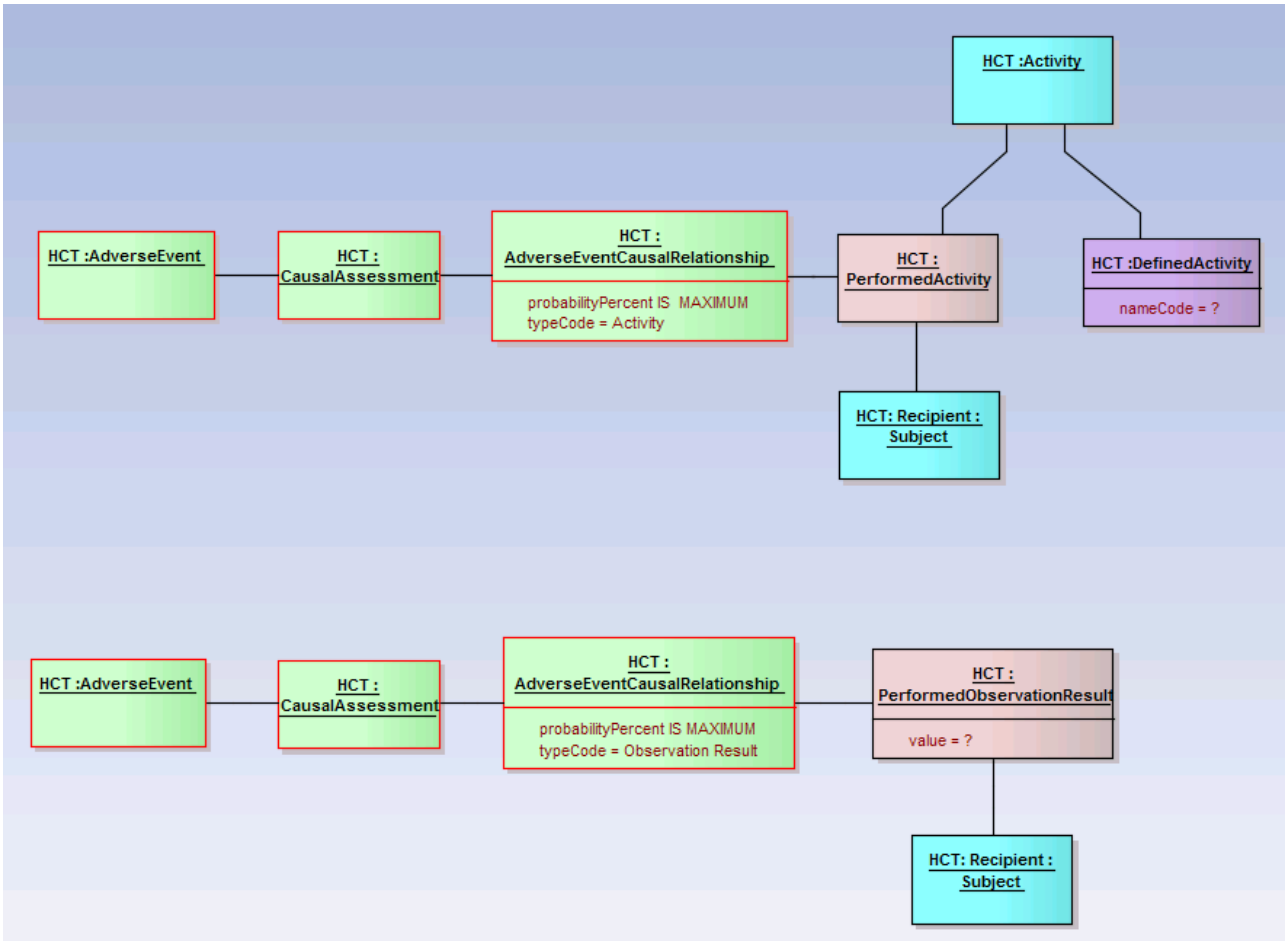


Fig 31. AdverseEventCausalRelationship entity

5.5.2.4 PerformedCompositionRelationship

When several activities are being performed, it is essential to know the chronological relationship between these activities indicating which activity occurred before or after the other activity. This information is captured by the PerformedCompositionRelationship entity, which relates two performed activities.

If we decide to use just dates instead of the "PerformedCompositionRelationship" entity we might need to add the date attribute to all activity entities, which could be tedious. Also we might not know the date sometimes and adding approximate dates would lead to false implications when bio-statisticians and analysts research this data. The quality of the data is also very poor as we go back in time and the chances of knowing the dates are very less.

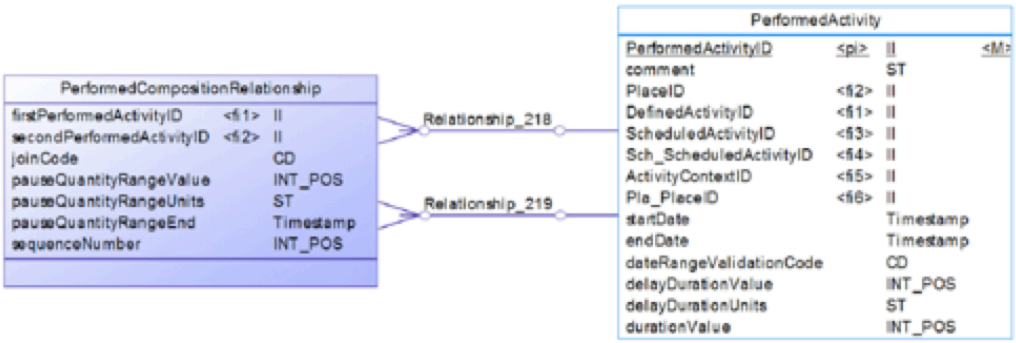


Fig 32. PerformedCompositionRelationship entity

5.5.3 Identifying Unnecessary Entities

Some implicit entities are included in the conceptual model to make the mapping path more meaningful. However some of these entities are not directly used in any of the mapping paths and are removed in the logical model in order to avoid creating an extra table for them. But their attributes might be used by some of its descendants and their relationships need to be preserved to maintain interaction between entities. Therefore the attributes and the relationships of these entities are passed over to its descendants.

As an example the “Activity” entity is deemed unnecessary, as it is not directly used in the mapping path. Hence all of its attributes and relationships are passed on to its descendants. This is done because the “reasonCode” attribute for instance, which indicates why the activity is being performed, is needed. Also some connections between entities go through the “Activity” entity and if the relationships are not passed on to its descendants, these connections could be broken.

Fig 33. Denormalizing Activity entity

5.6 Physical Data Model

A Physical Data Model [14] specifies all tables, columns and relationships between tables. It is created from LDM by converting all entities to tables, attributes to columns and all relationships to foreign keys. The data types of the columns are represented as ANSI compliant data types that will be used when the actual database is built using a database server such as Oracle SQL or MySQL.

The actual database server to be used is dependent on the requirements of the organization. The organization can use a database server that satisfies its price, performance and scalability requirements. For instance an organization can opt for Oracle server if it is going to use a lot of Java, XML, etc. to complement its database code or it can use MySQL server if it has budget constraints as most of the features in MySQL is free.

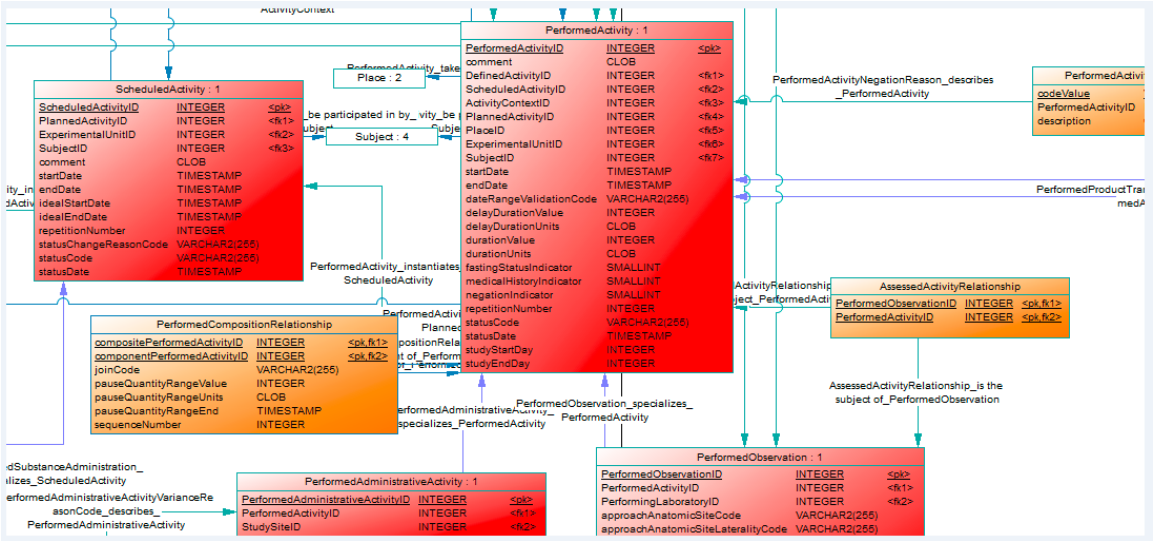


Fig 34. Sample PDM highlighting ANSI data types for columns

5.7 ANSI Compliance (Applying Abbreviations)

The ANSI standards allow for up to 30 characters for table and column names. Many table and column names in the PDM (originally from BRIDG) exceed the 30-character limit. If we don't reduce the table and column names to less than 30 characters, the database server being used might remove the extra characters, which might lead to name collisions. Hence it is necessary to apply abbreviations to trim the names of these table and column names before generating the DDL from the PDM.

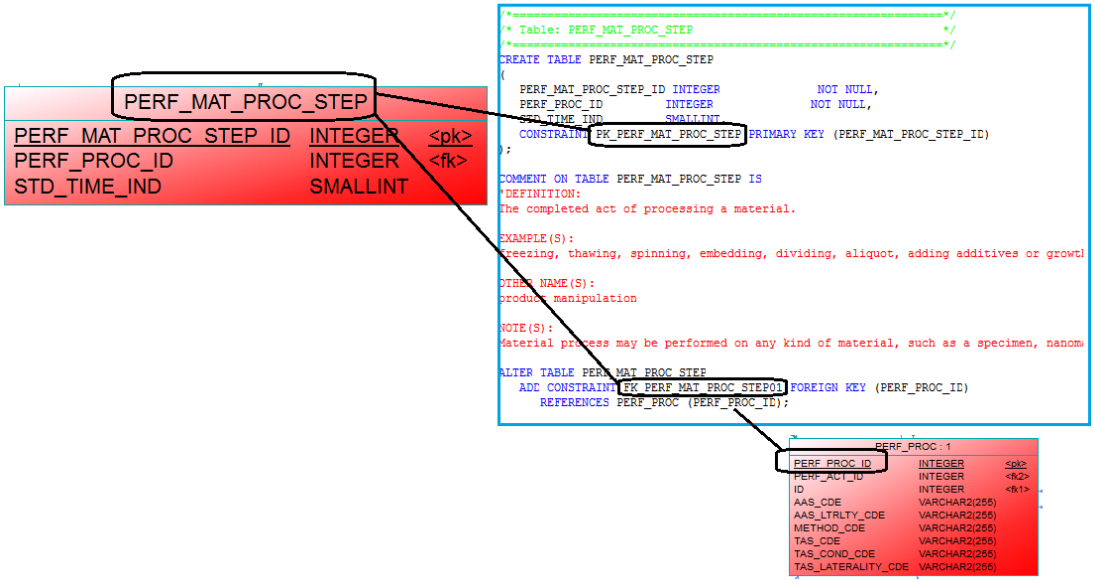


Fig 35. Applying abbreviation to PerformedMaterialProcessStep entity [Courtest of NDMP]

5.8 Generating DDL

Once the abbreviations are applied, the DDL script is generated from the PDM. This is done using Power Designer. A sample DDL for the "AssociatedBiologicEntity" table is shown in figure

```
*==============================================================*/
/* Table: ASSD_BIOL_ENT
*/
/
*==============================================================*/
CREATE TABLE ASSD_BIOL_ENT
(
   CDE_VAL              VARCHAR2(254)         NOT NULL,
   FST_BIOL_ENT_ID      INTEGER               NOT NULL,
   SECOND_BIOL_ENT_ID   INTEGER               NOT NULL,
   "DESC"               CLOB,
   CONSTRAINT PK_ASSD_BIOL_ENT PRIMARY KEY (CDE_VAL,
SECOND_BIOL_ENT_ID, FST_BIOL_ENT_ID)
);

COMMENT ON TABLE ASSD_BIOL_ENT IS
'DEFINITION:
An individual biologic entity connected/linked to another
biologic entity.

EXAMPLE(S):
family member, roommate, nursing home attendant

OTHER NAME(S):

NOTE(S):
';

/
```

Fig 36. Sample DDL script for creating the "AssociatedBiologicEntity" table

6. Conclusions and Future Work

The BRIDG HCT physical model was developed to facilitate transplant data exchange and submission into the CIBMTR by providing a means for organization to develop their own data systems. A model driven architecture was implemented in order to manage the inclusion of new domains, entities and other changes in an efficient manner.

The physical model is ANSI compliant and is not implementation specific to any particular database server. The choice of the database server is left for the organization to decide depending on their budget, performance and scalability requirements.

With the recent release of BRIDG v4.0 that includes the integration of LSDAM, some new CDEs will be created and consequently new entities will be added to the model to explain the mappings.

The identifiers of tables that are INTEGERs will be changed to either GUIDs or OIDs in one of the future releases to provide a better practical solution.

Also error-handling mechanisms for data types that were not done due to timing constraints will be implemented in the future releases. For instance, the “CD” data type is implemented as a key-value pair table. But we need to check if the key value entered is a valid one. This could be done using one of several solutions. One solution is to use triggers to check if the key value entered is valid after every insertion and update. While this solution is easy to maintain we need to keep in mind the performance degradation that comes with using a lot of triggers.

7. Acknowledgment

S No.	Name	Affiliation
1	John Carlis	Advisor/Mentor from the UofM
2	Bob Milius	Mentor from NMDP
3	Jane Pollack	Mentor/Supervisor from NMDP
4	Robinette Renner	Mentor from CIBMTR
5	Harry Vassilev	Mentor from CIBMTR
6	Sandra Sorensen	Mentor from CIBMTR
7	Kirt Schaper	Mentor from CIBMTR
8	Demetria Wiley	Project Manager from CIBMTR
9	Charles Martinez	Mentor from MD Anderson
10	Shrimi Sharma	Teammate – Spring 2014
11	Sameera Shah	Teammate – Spring 2014
12	Yumeng Wang	Teammate – Spring 2014
13	Angel Villahoz-Baleta	Teammate – Summer 2014
14	Emily Cherkassky	Teammate – Summer 2014
15	Jasmine Joseph	Teammate – Summer 2014

8. References

1. http://en.wikipedia.org/wiki/Physical_data_model
2. http://www.bridgmodel.org/
3. http://www.cibmtr.org/
4. http://en.wikipedia.org/wiki/Hematopoietic_stem_cell_transplantation
5. https://wiki.nci.nih.gov/display/LS/Life+Sciences+Domain+Analysis+Model+(LS+DAM)
6. http://en.wikipedia.org/wiki/Gene_ontology
7. http://www.omg.org/gettingstarted/gettingstartedindex.htm
8. http://en.wikipedia.org/wiki/Unified_Modeling_Language
9. https://cbiit.nci.nih.gov/ncip/biomedical-informatics-resources/interoperability-and-semantics/metadata-and-models
10. http://en.wikipedia.org/wiki/Object_diagram
11. http://en.wikipedia.org/wiki/Conceptual_schema
12. http://www.hl7.org
13. http://www.1keydata.com/datawarehousing/logical-data-model.html
14. http://www.1keydata.com/datawarehousing/physical-data-model.html