

Multihazard Loss Estimation Methodology

HAZUS®MH MR4

Software Design Description for the Earthquake Model and the Shell

Developed by:

PBS&J

For:

Department of Homeland Security

Emergency Preparedness and Response Directorate

FEMA

Mitigation Division

Washington, D.C.

Under a contract with:

National Institute of Building Sciences

Washington, D.C.

©2007, Federal Emergency Management Agency

(Secured by Assignment)

HAZUS® is a trademark of the Federal Emergency Management Agency

Foreword

The research, development, and studies that provided the basis for this publication were conducted pursuant to a contract with the Federal Emergency Management Agency (FEMA) by the:



The National Institute of Building Sciences (NIBS), located in Washington, D.C., is a non-governmental, non-profit organization, authorized by Congress to encourage a more rational building regulatory environment, to accelerate the introduction of existing and new technology into the building process, and to disseminate technical information.

For information on obtaining copies of this report, contact FEMA at:

<http://www.fema.gov/plan/prevent/hazus/index.shtm>

or:

FEMA Distribution Center
P.O. Box 2012
Jessup, MD 20794-2012
Tel.: 800-480-2520
Fax: 301-362-5335

HAZUS® is a trademark of the Federal Emergency Management Agency.

Contents

1.	Introduction	1
1.1.	Purpose	1
1.2.	Scope	1
1.3.	Definitions, Acronyms, and Abbreviations	1
1.3.1.	Definitions	1
1.3.2.	Acronyms and Abbreviations	2
2.	References	4
3.	Document Overview	5
4.	Decomposition Description	6
4.1.	Database Design	6
4.1.1.	Conceptual Data Model	6
4.1.2.	Reading a CDM/PDM Diagram	10
4.1.3.	Database Structure	12
4.1.4.	Aggregated Data	15
4.1.5.	Essential Facilities	21
4.1.6.	High Potential Loss Facilities	23
4.1.7.	Bridges	25
4.1.8.	Segments	27
4.1.9.	Transportation Facilities	29
4.1.10.	Utility Facilities	32
4.1.11.	Pipelines	34
4.1.12.	Tunnels	36
4.1.13.	Guidelines for Group Development of the HAZUS MH Database	37
4.2.	Interface Design (Presentation Services Layer)	40
4.2.1.	The Study Region Aggregation Wizard	40
5.	Dependency Description	51
5.1	Mapping Engine	51
5.1.1.	Inventory Mapping	51
5.1.2.	Mapping General Building Stock Inventory	55
5.1.3.	Inventory Editing	56
5.1.4.	Mapping General Building Stock Results	59
5.1.5.	Mapping Inventory Results	63

5.1.6.	Scenario Definition	66
6.	Interface Description	68
6.1.	Database Access Layer	68
6.2.	Application Logic Layer	73
6.2.1.	Aggregation Layer	73
6.2.2.	The Analysis Engine	74
6.3.	Presentation Services Layer	81
6.3.1.	Occupancy Mapping Dialog Interface	81
6.4.	Foundation Type Dialog Interface	84
6.5.	The Data Browser	84
6.6.	The Mapping Engine	87
6.6.1.	The Mapping Engine Components	88
7.	Detailed Design Description	109
7.1.	Database Access Layer (DAL)	109
7.1.1.	Security Model for the HAZUS-MH Database	109
7.1.2.	Database Schemas and SQL Scripts	111
7.1.3.	Data Access Methods	111
7.1.4.	Aggregation Process in DAL	112
7.2.	Application Logic Layer (ALL)	114
7.2.1.	The Study Region Aggregation Engine	114
7.2.2.	The Analysis Engine	119
7.3.	Presentation Services Layer (PSL)	133
7.3.1.	Study Region Aggregation Wizard	134
7.3.2.	Inventory Menu	134
7.3.3.	Hazard Menu	134
7.3.4.	Analysis Menu	134
7.3.5.	Results Menu	134
7.4.	The Custom UI Objects	134
7.5.	The Data Browser	134
7.5.1.	Displaying Data in the Spreadsheet Grid on the Dialog	137
7.5.2.	Start Editing	138
7.5.3.	Add New Record	138
7.5.4.	Delete Selected Records	139
7.5.5.	Editing Records	140

7.5.6.	Import	140
7.5.7.	Stop Editing	141
7.5.8.	Export	141
7.5.9.	Filter	142
7.5.10.	Calculate Statistics	143
7.6.	The Mapping Engine	143
Appendix A:	HAZUS-MH System Database Schema	
Appendix B:	HAZUS-MH Template Database Schema	

1. Introduction

The HAZUS-MH Earthquake Model Software Design Description (SDD) presents in detail the different design concepts to be implemented in the Earthquake Model of HAZUS-MH. It also covers what is termed the “Shell API.” The Shell API implements mainly the study region aggregation feature and the common utility API like viewing the common inventory.

1.1. Purpose

The purpose of this document is to provide the precise design information needed for planning, analysis, and implementation of the HAZUS-MH Earthquake Model.

1.2. Scope

The HAZUS-MH Earthquake (referred to from now on as the Eq. model) is the component which gets launched in HAZUS-MH when the user selects an earthquake hazard and opens a study region.

The HAZUS-MH software is partitioned into three main design entities: the Data Access Layer, Application Logic Layer and the Presentation Services Layer. This document provides the attributes for each design entity, the relationships among entities, interface descriptions and other necessary design details to satisfy the Software Requirements Specifications and provide the necessary information for implementation of the software.

1.3. Definitions, Acronyms and Abbreviations

1.3.1. Definitions

The following terms from Section 3 of the IEEE Standard 1016-1998 are used to establish meaning in the context of this document.

Design Entity – An element (component) of a design that is structurally and functionally distinct from other elements and that is separately named and referenced.

Design View – A subset of design entity attribute information that is specifically suited to the needs of a software project activity.

Entity Attribute – A named characteristic or property of a design entity. It provides a statement of fact about the entity.

Software Design Description (SDD) – A representation of a software system created to facilitate analysis, planning, implementation, and decision making. A blueprint or model of the software system. The SDD is used as the primary medium for communicating software design information.

1.3.2. Acronyms and Abbreviations

The following table lists a definition of acronyms and abbreviations used throughout this document.

Table 1-1 Acronyms and Abbreviations

Acronym/Abbreviation	Definition
ADO	ActiveX Data Objects
AEBM	Advanced Engineering Building Model
AEM	Analysis Engine Module
ALL	Application Logic Layer
AML	Aggregation Map Layer
ARA	Applied Research Associates
ATL	Active Template Library
BIT	Building Import Tool
CD	Compact Disk (also see CD-ROM)
CDM	Conceptual Data Model
CD-ROM	Compact Disk – Read Only Memory
CEUS	Central and Eastern United States
DAL	Data Access Layer
DDS	Detailed Design Specification
DTI	Durham Technologies Inc.
DTS	Data Transformation Services
EF	Essential Facilities

Acronym/Abbreviation	Definition
Eq.	Earthquake
ERD	Entity Relationship Diagram
ESRI	Environmental Systems Research Institute
FEMA	Federal Emergency Management Agency
FFE	Fire Following Earthquake
GBS	General Building Stock
GIS	Geographic Information Systems
GUI	Graphical User Interface
HAZUS	Hazards U.S. (natural hazards loss estimation software program)
HAZUS-MH	Multihazard HAZUS
HFT	Hazard Factor Tables
HPLF	High Potential Loss Facilities
IEEE	Institute of Electrical and Electronics Engineers
INCAST	Inventory Collection and Survey Tool
MFC	Microsoft Foundation Classes
MSDE	Microsoft Database Engine
NIBS	National Institute of Building Sciences
OCX	ActiveX File Extension
PESH	Potential Earth Science Hazards
PGA	Peak Ground Acceleration
PGD	Permanent Ground Deformation
PGV	Peak Ground Velocity
PSL	Presentation Services Layer
RAM	Random Access Memory
RDBMS	Relational Data Base Management System
SDD	Software Design Descriptions
SRS	Software Requirements Specifications
TBD	To Be Determined
UI	User Interface

Acronym/Abbreviation	Definition
WFL	Workflow Layer
WUS	Western United States

2. References

The following documents are referenced throughout the SDD and contain important information related to the SDD.

1. "HAZUS-MH registry Structure": outlines the detailed structure of the Windows registry used by HAZUS-MH
2. "HAZUS-MH CD Packaging and Distribution Layout": outlines the packaging and CD layout format of HAZUS-MH shipped to the user.
3. "HAZUS-MH Setup Design Document": outlines the steps and features implemented by the HAZUS-MH setup program.
4. "BIT (Building Import Tool) Design Document": documents the new features implemented in the upcoming version of BIT.

Also of importance is the HAZUS-MH Software Requirement Specifications.

The following documents distributed previously have been integrated in this SDD and are therefore obsolete:

1. Data Model Description Version 5 (latest version).
2. Data Packaging document.
3. Earthquake Module Menu Specification
4. Addendum to the Earthquake Module Menu Specification

3. Document Overview

This document generally follows the example provided in Annex A of the IEEE Recommended Practice for Software Design Descriptions. After providing a list of references in Section 2, the remainder of the document is divided into six main sections: Decomposition Description (Section 4), Dependency Description (Section 5), Interface Description (Section 6) and Detailed Design (Section 7). The principal design entities described in Sections 4 and 5 are the Data Layer, Application Layer, and Presentation Layer.

The Data Layer includes the conceptual and physical data models for the database and the Data Services Package, which will be used to access data from the underlying RDBMS using ActiveX Data Objects (ADO). The Application Layer implements the business logic components. This layer includes the Analysis Engine and will also provide a layer of abstraction between the user interface and underlying database. Finally, the Presentation Layer implements the Graphical User Interface (GUI).

These design entities support the Shell and the three main hazard models: Earthquake, Hurricane and Flood. The HAZUS-MH Shell is the launching application that initiates an interface common across all models that enables the user to choose the study region and the type of hazard (earthquake, flood or hurricane) that the user wants to work with. The Shell launches one of the three hazard-specific models. The Shell composed of common object classes enables the three-hazard specific models to access common functionality and provide users with a consistent and integrated user interface. Each model will utilize the Shell's common objects initially before initiating the hazard-specific functions and interfaces. The common objects contained in The Shell are included in the Software Design Document (SDD) for each hazard model in order to provide a comprehensive design specification for each model that will describe how the Shell interacts with the hazard model components. The HAZUS-MH Earthquake Model is the application that the user will use to evaluate the effect of an earthquake on a particular study region. This SDD focuses exclusively on the Earthquake Model and its interaction with the Shell.

4. Decomposition Description

This section describes the three-tier layering implemented in the Earthquake Model of HAZUS-MH which matches also the same layering in the overall HAZUS-MH models: Flood Model, Hurricane Model, and the Shell.

The three-tier architecture relies on a typical data access layer, application logic layer, and a presentation services layer.

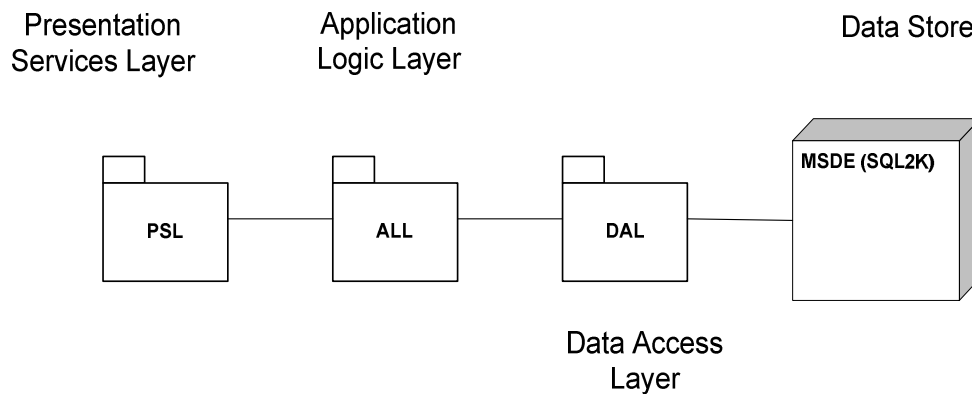


Figure 4-1. Layer Architecture in HAZUS-MH

The following sections will provide detail on the structure of each layer.

4.1. Database Design

The database design in HAZUS-MH is the cornerstone for the overall global design of the application. The purpose of this section is to explain the concepts used to develop the HAZUS-MH conceptual data model, the naming convention used, and the steps required to finalize the model.

4.1.1. Conceptual Data Model

The main goal in developing the Conceptual Data Model (CDM) was to design an architecture to account for the multihazard nature of HAZUS-MH.

The relational feature of the target platform for HAZUS-MH (MSDE/SQL2K) lends itself to this approach. The CDM implements the following:

- The common attributes to all three hazards are stored in a common-attribute table, which we will refer to as the hz table. That derives from its naming convention hzxxxx, where xxxx is the descriptive name for the table (1-40 characters), and hz is simply an abbreviation for HaZus
- The specific attributes for each hazard are stored in corresponding tables named eqxxxx, huxxxx, flxxxx, which respectively correspond to the earthquake, hurricane, and flood attributes. These will be referred to as the eq, hu, and fl tables. A one-to-one relationship links the fl, eq, hu tables to the hz table
- The eq, hu, and fl tables store both the attributes for the corresponding hazard and all the attributes/columns for the analysis results. Each contractor¹ is responsible for defining and implementing those attributes/columns. Past experience has shown that it is difficult to finalize the list at this stage, but the proposed architecture for the CDM makes this a non-issue for other contractors. In other words, each contractor can change its table as many times as required without affecting the development process of the other contractors, as long as the integrity rules are respected as per the proposed CDM
- Most of the CDM tables are implemented through inheritance recognizing the commonality of the attributes between different tables
- The general naming convention relies on predefining the first 2 characters (lower-case), and following the “C++ Source Code Guidelines” for the remaining characters (mixture of upper/lower case)

Table 4-1. Naming Convention of Database Objects

Table	Naming convention
Classification tables	cl
Flood Tables	fl
Hurricane Tables	hu

¹ Contractors include DTI (working with the HAZUS Shell and Earthquake Models), ABS (working with the Flood Model) and ARA (working with the Hurricane Model)

Earthquake Tables	eq
Multi-hazard Tables	hz
System Tables	sy
Attributes (fields)	AaaaAaaa Alphanumeric, mix upper/lower, no underscore, 1st char is upper. Any length but <20 recommended.
References/Relationships	rlnaaaa_aaaa 1st 3 chars are rln, Alphanumeric word before underscore is 1st entity of the relationship, alpha word after the underscore is the 2nd entity in the relationship. All lowercase. Any length but <20 recommended. Use acronyms if needed.
Identifier	idt<table name>
Primary Key	<table name>Id Exception #1: for hz tables, the hz is dropped. Exception #2: for some lifeline tables, instead of using the convention <table name>Id which generates a very long name, the abbreviated version based on the name in Error! Reference source not found. is used.
Views	veqXXXXXX, vhuXXXXXX, vflXXXXXX
Stored Procedures	peqXXXXXX, pflXXXXXXXX, phuXXXXX
Triggers	tyhzXXXXX where: y is 'i', 'u', 'd' to reflect type of trigger: insert, update, and delete. hz is hazard module (eq, fl, or hu) XXXXXX is the descriptive name

User-defined Functions	feqXXXX, fflXXXX, fhuXXXX
------------------------	---------------------------

- The challenge in implementing a naming convention is to strike a balance between having the right length for a given identifier. Usually, a fully descriptive identifier can end up being 40 to 60 characters long. Programmers prefer shorter names but they tend to be cryptic. The following acronyms are used in the CDM, mainly for naming the relationships.

Table 4-2. Naming Convention for Lifeline Systems Tables

Category	Table	Acronym
Lifelines Bridges	Highway	hbr
	Railways	rbr
	Light rail	lbr
Lifelines Utilities Facilities	Potable water	pfa
	Waste water	wfa
	Oil	ofa
	Natural gas	nfa
	Electric power	efa
	Communication	cfa
Lifeline Transportation Facilities	Ferry systems	ffa
	Light rail systems	lfa
	Railways	rfa
	Bus systems	bfa
	Airports	afa
	Ports	tfa
	Runways	yfa
Lifeline Segments	Highways	hsg
	Light rail	lsg
	Railways	rsg
Pipelines	Natural gas	npl
	Oil	opl
	Waste water	wpl
	Potable water	ppl
Tunnels	Highways	htu

	Railways	rtu
	Light rail	ltu

To allow flexibility in defining the data types, the CDM makes use of domains definition. These are predefined data types to which the tables' attributes are mapped later. The advantage of using domains instead of setting data types explicitly is flexibility: if at a later stage, there is a need for changing the data type for a given field -say char(40) instead of char(20)- then instead of replacing each occurrence of that attribute, it is easier to just change the domain definition.

Appendix A "The CDM Report" lists the domains currently defined in HAZUS-MH. It is recommended that all contractors use the same list and avoid duplicate definitions.

4.1.2. Reading a CDM/PDM Diagram

The CDM and PDM (Physical Data Model) diagrams pack a lot of information, which is lost if a reader does not understand the different symbols used. Below is a brief description of the diagrams, Refer to the software² user manuals or any ERD (Entity Relationship Diagram) textbook for additional information.

² Sybase PowerDesigner 7.5

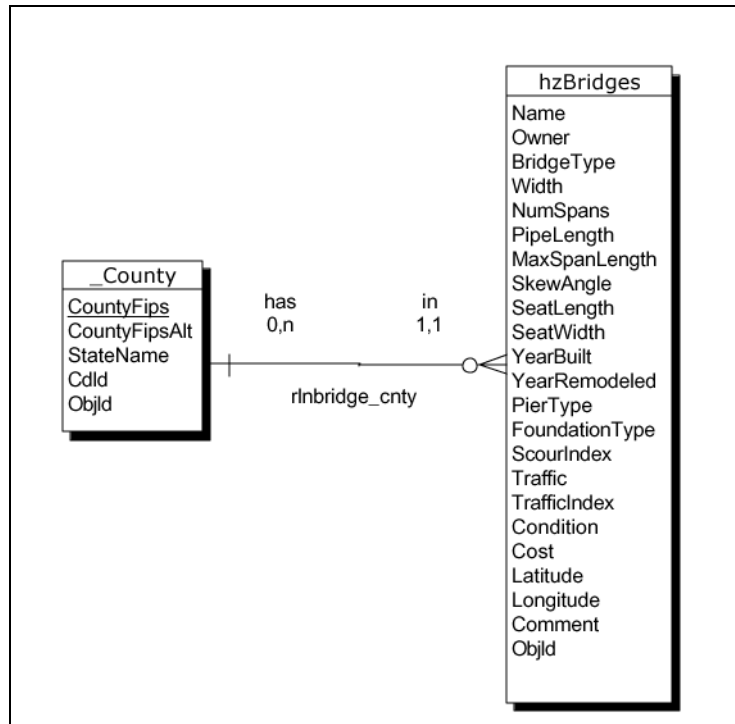


Figure 4-2. Tables Relationships Sample




In the diagram above, there is a relation between the table hzBridges and County (the name of the relationship is rlnbridge_cnty). The type of the relationship is one-to-many. What this means is that a bridge can be located within only one county (a bridge cannot be located in 2+ counties at the same time) and this is noted on the diagram as 1,1. A county on the other hand can have as many bridges located in it (or none), and this is noted as 0,n. The word “in” and “has” are the roles of the relationship.

The fact that a bridge has to be located in a county (it cannot be located nowhere) implies a mandatory relationship which is noted on the diagram by the tickmark.

The fact that a county may have or may not have bridges implies an optional relationship that is noted on the diagram with a circle. Table 4-3 shows other possible combinations of relationships.

Table 4-3. Symbols for Different Types of Relationships

Termination Point	Existence	Cardinality	Description
	Mandatory	One	Must exist one and

			only one
	Mandatory	Many	Must exist one or more
	Optional	One	May exist one, or none
	Optional	Many	May exist one or more, or none

Example 2: Inheritance

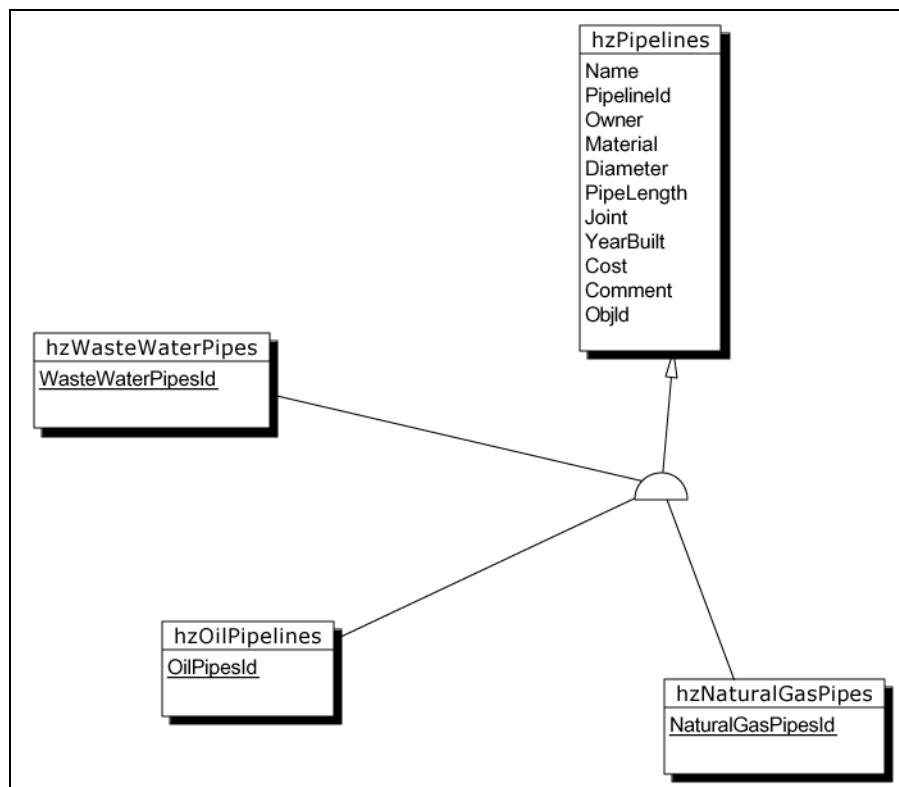


Figure 4-3. Inheritance Symbol Sample.

In HAZUS-MH, few tables have common attributes between them and differ only to some degree. To model these relationships, the concept of (table) inheritance is used. Figure 3-3, depicts this through the half-circle symbol with the arrow pointing to the parent table. In this example, the tables hzWasteWaterPipes, hzOilPipelines, and hzNaturalGasPipes all inherit the attributes of hzPipelines, but each table adds its own primary index (shown underlined). Inheritance parameters control the generation options in the PDM.

4.1.3. Database Structure

The CDM developed in version 1 (and its corresponding PDM) will reflect the structure of the study region database. It is intended that the databases structure implemented in HAZUS-MH will mirror the database structure in HAZUS99, mainly:

- There will be a database for each study region created by the user
- There will be a database, syHAZUS, which is a system database. It will store internal tables used by the program (among other things, it will store the list of study regions created by user, and thus the list of study region databases). This implementation mirrors the data folder in HAZUS99
- In HAZUS99, a new region is created by making a copy of the template folder³. In MS SQL Server, a new database is always created by making a copy of the MODEL database, so it is possible to replicate the HAZUS99 functionality by simply modifying the MODEL database to reflect an equivalent template structure. In the short term⁴, this is fine, but in the long term when running on a full version of SQL Server, that would interfere with using SQL Server for anything other than HAZUS-MH. To avoid difficulties, the model database is planned to stay as is, and implement the equivalent functionality through a custom SQL scripts

A typical SQL Server database structure would be like shown in Figure 4-4.

³ Template folder in HAZUS99 is simply an empty study region.

⁴ The short term is defined here as being the initial version of HAZUS-MH implemented using MSDE. The long term is the web enabled version of HAZUS-MH.

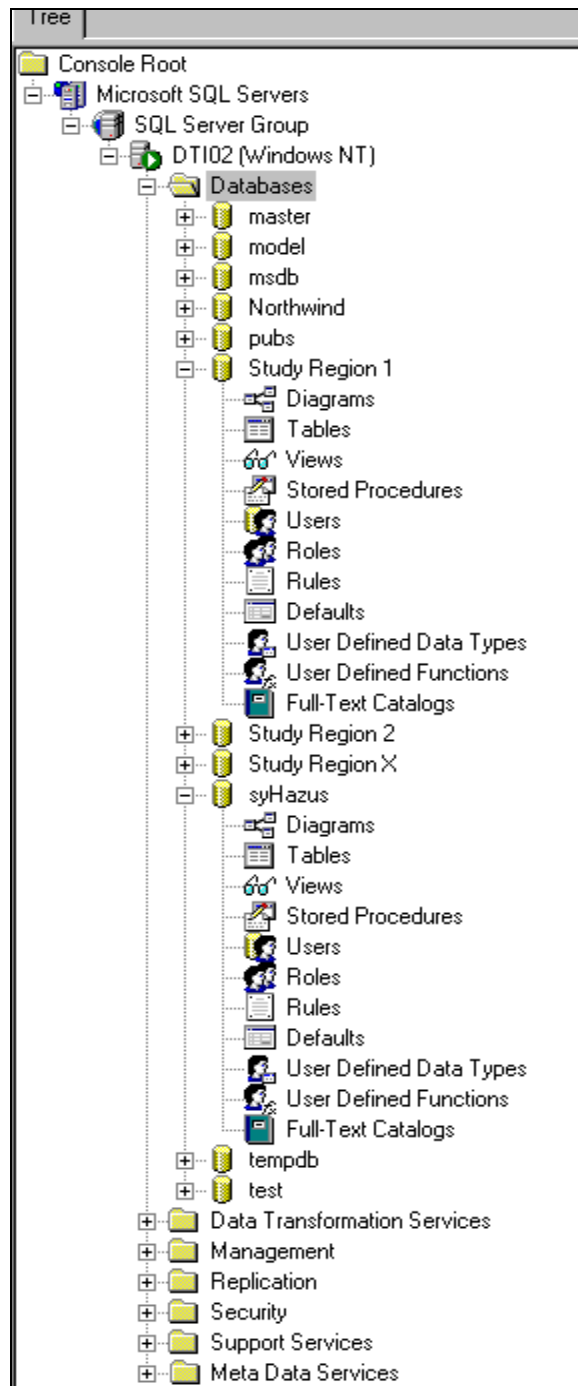


Figure 4-4. HAZUS-MH Databases Hierarchy in MSDE / SQL2K

At a specific inventory level, the HAZUS-MH data model has been designed such that the interaction between the three-hazard related attributes (earthquake, flood, and hurricane) is self-contained. This can be reflected in the following diagram where each inventory object is split into:

- The main table labeled hzXxxxx
- The earthquake attributes plus earthquake results table labeled eqXxxxx
- The flood attributes plus the flood results table labeled flXxxxx
- The hurricane attributes plus hurricane results table labeled huXxxxx
- The relationship between the main table and the hazard-specific tables is one-to-one relationship
- The primary key in the main table is of type char(9) with the format AA999999 where: AA is the two-char state-code (CA, DC, GA, NC, ...) and 999999 is a six-digit sequential number. This key is the foreign key and also the primary key in all the three-hazard-specific tables. This key value also matches the value used in the geodatabase if the table is mappable

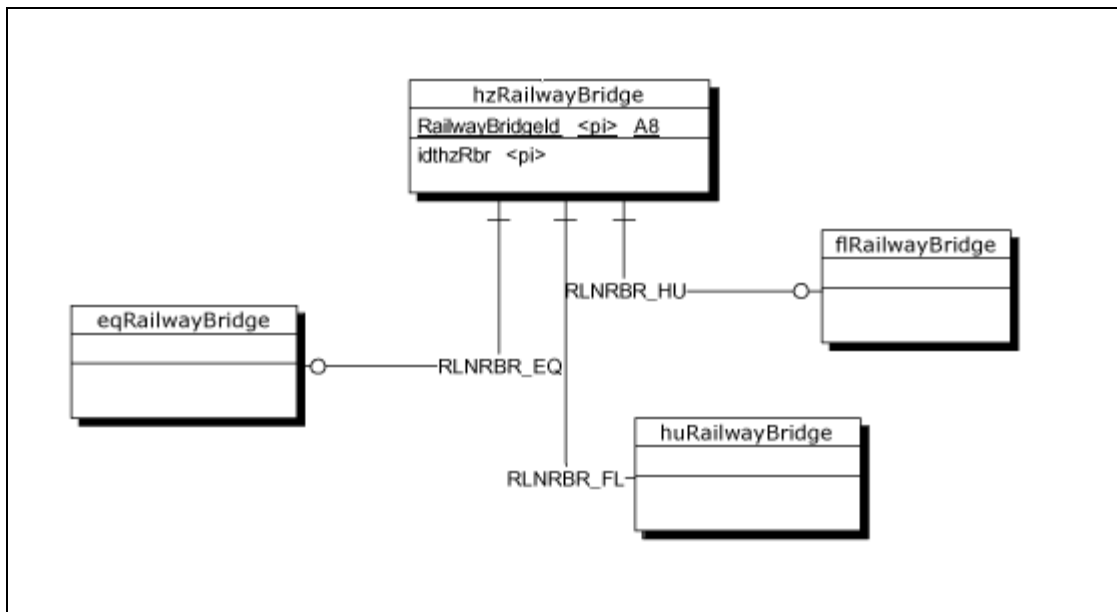


Figure 4-5. Data Model Hierarchy Concept

4.1.4. Aggregated Data

The aggregated data covers all of the attributes which are grouped by census tract or census block. This is mostly what is termed General Building Stock (GBS). The complexity in the GBS tables stems from the fact that most of the attributes have to be shown under different angles, namely by occupancy and/or by building type, and within each, the attributes are grouped by general occupancy/building type and specific occupancy/building type.

Table 4-4. Main GBS Tables Editing Modes

Study Region Hazard/Table	Category	Earthquake only or Earthquake + Hurricane	Earthquake + Flood
Square Footage	Specific Occupancy	Editable at tract level	Editable at block level & then aggregated automatically to the tract.
	General Occupancy.	Non-editable. Computed automatically	Non-editable. Computed automatically
	Specific Building Type	Non-editable. Computed automatically	Non-editable. Computed automatically
	General Building Type	Non-editable. Computed automatically	Non-editable. Computed automatically
Building Count	Specific Occupancy.	Editable at tract level	Editable at block level & then aggregated automatically to the tract.
	General Occupancy.	Non-editable. Computed automatically	Non-editable. Computed automatically
	Specific Building Type	Non-editable. Computed automatically	Non-editable. Computed automatically
	General Building Type	Non-editable. Computed automatically	Non-editable. Computed automatically
Exposure	Specific Occupancy.	Editable at tract level	Editable at block level & then aggregated automatically to the

Study Region Hazard/Table	Category	Earthquake only or Earthquake + Hurricane	Earthquake + Flood
			tract.
	General Occupancy.	Non-editable. Computed automatically	Non-editable. Computed automatically
	Specific Building Type	Non-editable. Computed automatically	Non-editable. Computed automatically
	General Building Type	Non-editable. Computed automatically	Non-editable. Computed automatically
Demographics		Editable at tract level	Editable at block level & then aggregated automatically to the tract.

An important difference between HAZUS-MH and HAZUS99 is that all four tables above are independent; in other words, changes in the square footage do not affect the building count or the dollar exposure (like they did in HAZUS99). As a result, all four tables are editable by the user at any time.

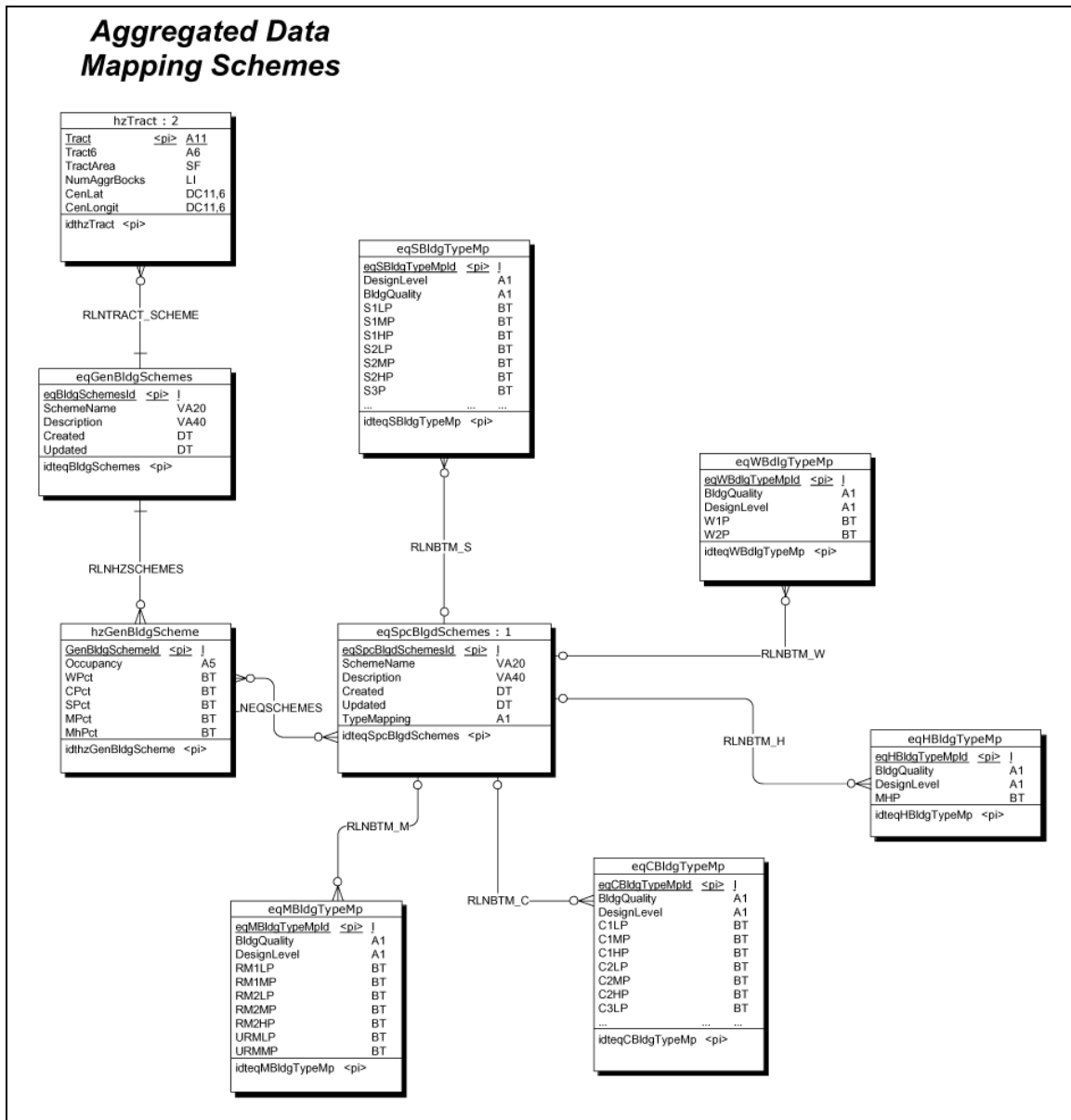


Figure 4-8. CDM for the Occupancy Mapping Scheme Tables

The major change is the split of the occupancy mapping scheme into two levels:

- Mapping of specific occupancy classes to general building type: that is the mapping which is common to all three hazards
- Mapping of general building type to specific building type: which is specific to each hazard (since the specific building type classification varies from one hazard to another)

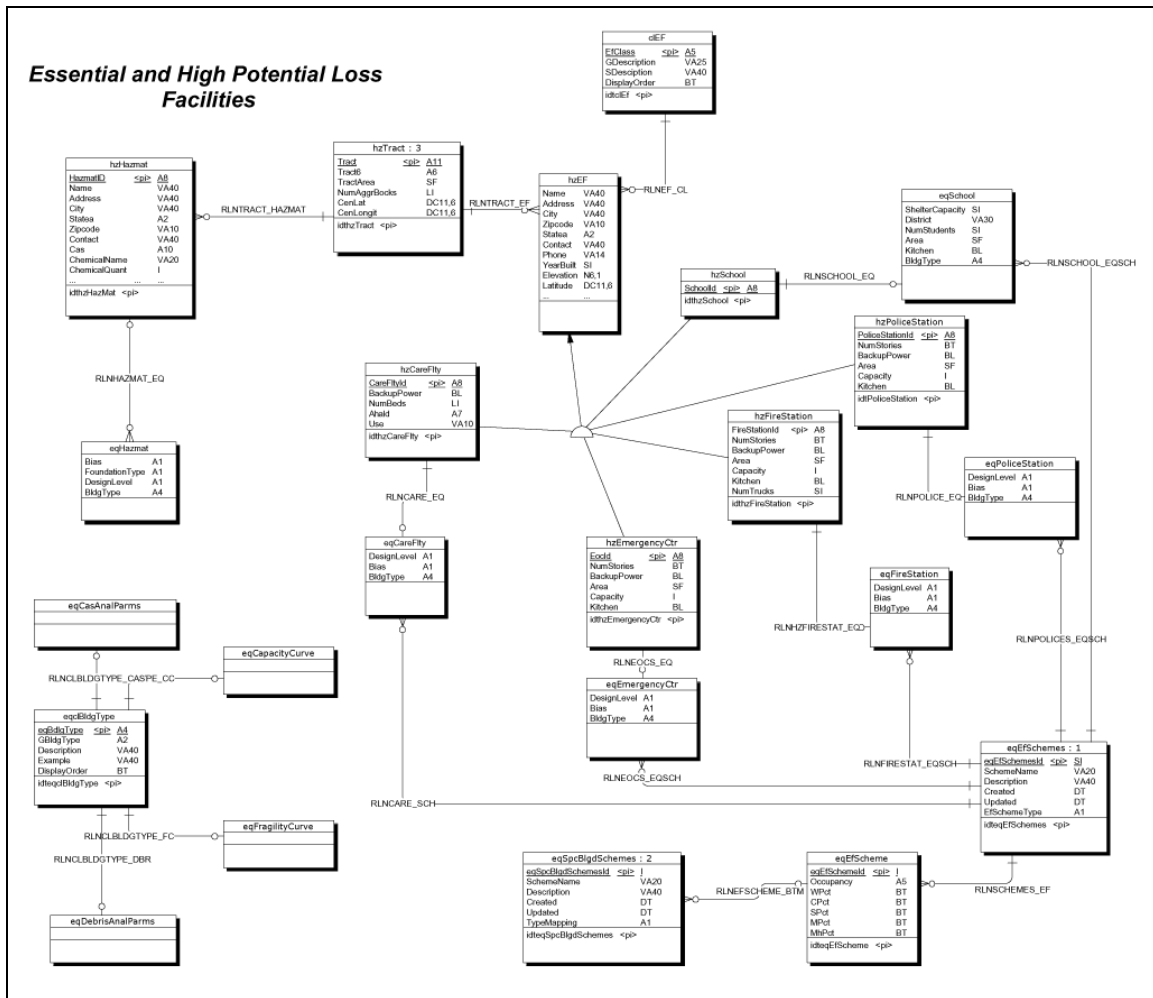


Figure 4-10. CDM for the Essential Facilities

The essential facilities rely on the same occupancy mapping concept as the GBS and re-uses specific mapping schemes (i.e. those which map the general building types to the specific building type by design level and building quality).

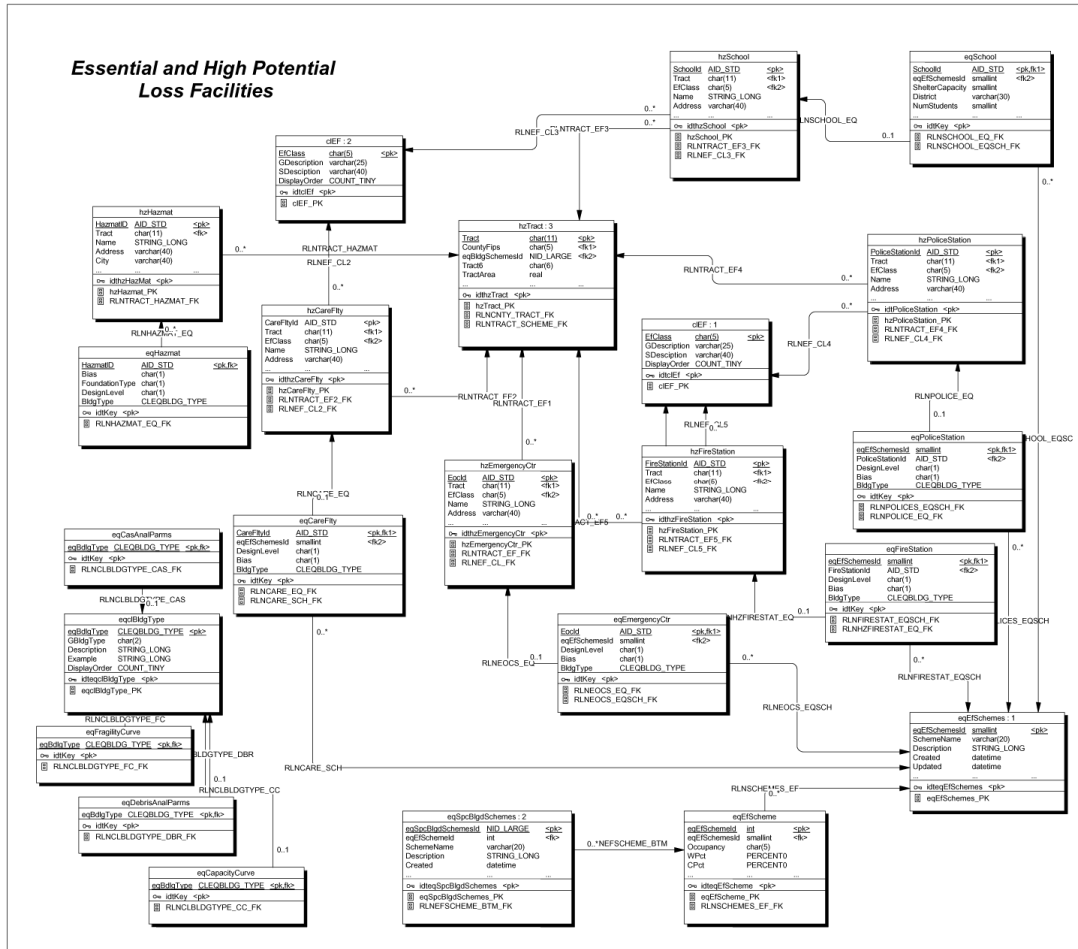


Figure 4-11. PDM for the Essential Facilities

4.1.6. High Potential Loss Facilities

High potential loss facilities (HPLF) are comprised of military installations, nuclear facilities, dams, and levees. HAZUS-MH (like HAZUS99-SR2) implements an analysis engine only for the military installations.

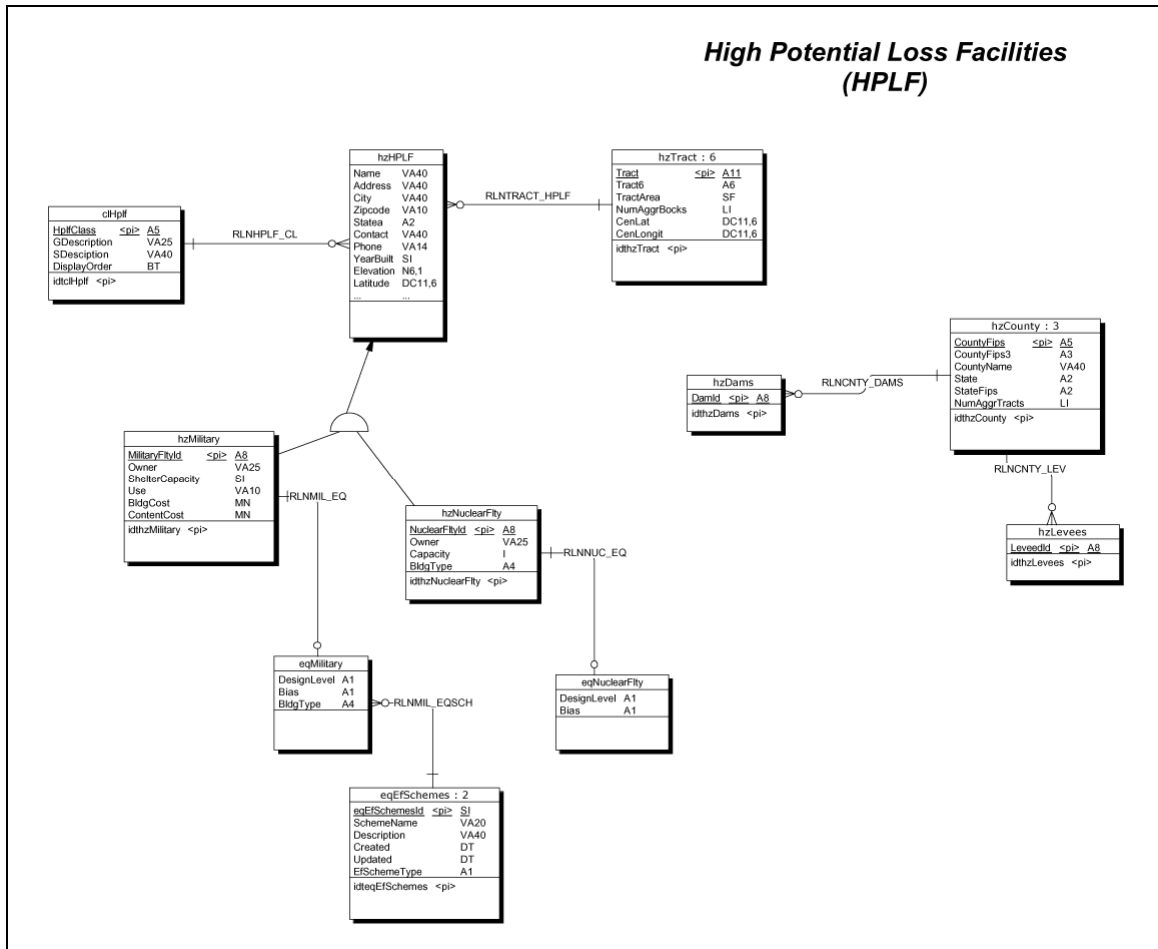


Figure 4-12. CDM for the High Potential Loss Facilities

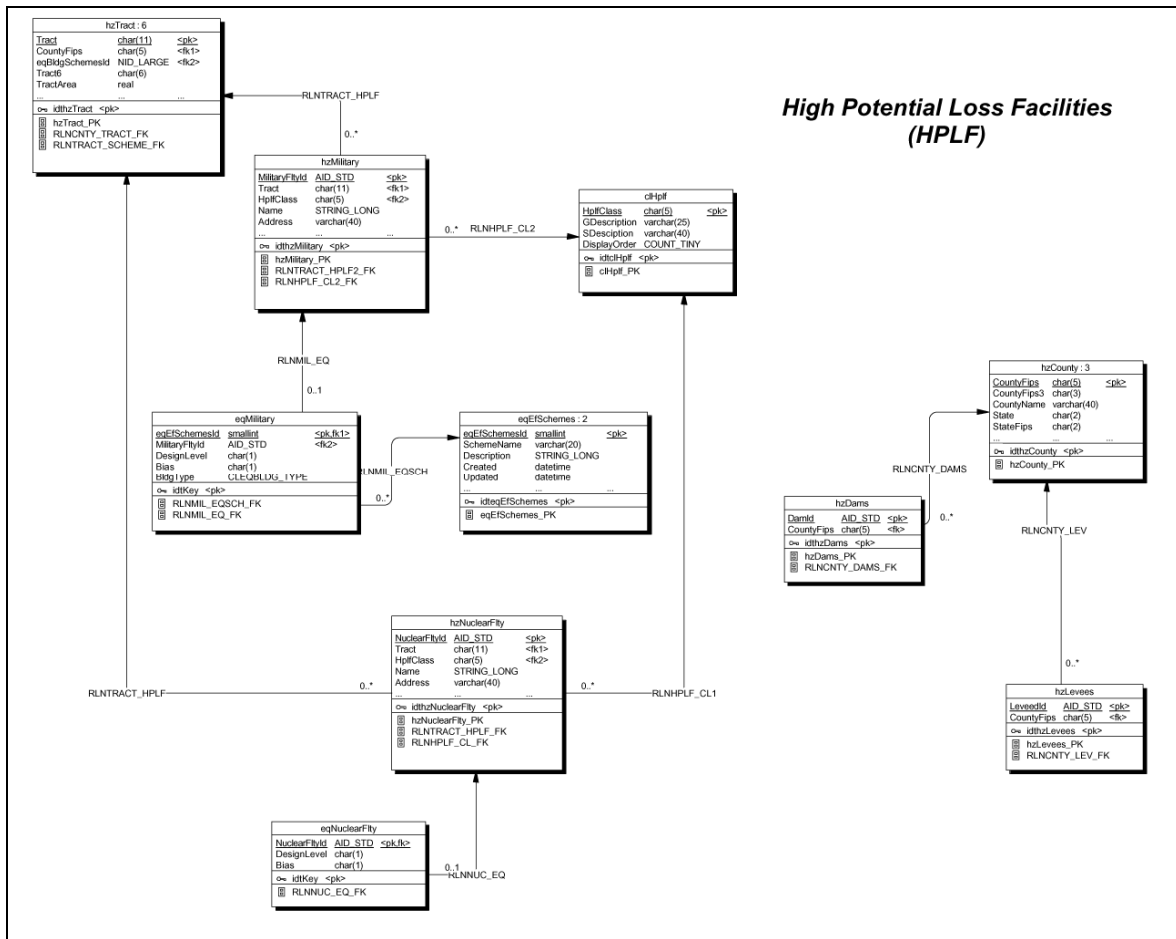


Figure 4-13. PDM for the High Potential Loss Facilities

4.1.7. Bridges

The transportation bridges are classified into highway bridges, railway bridges, or light rail bridges. All three classifications share the same attributes and analysis parameters.

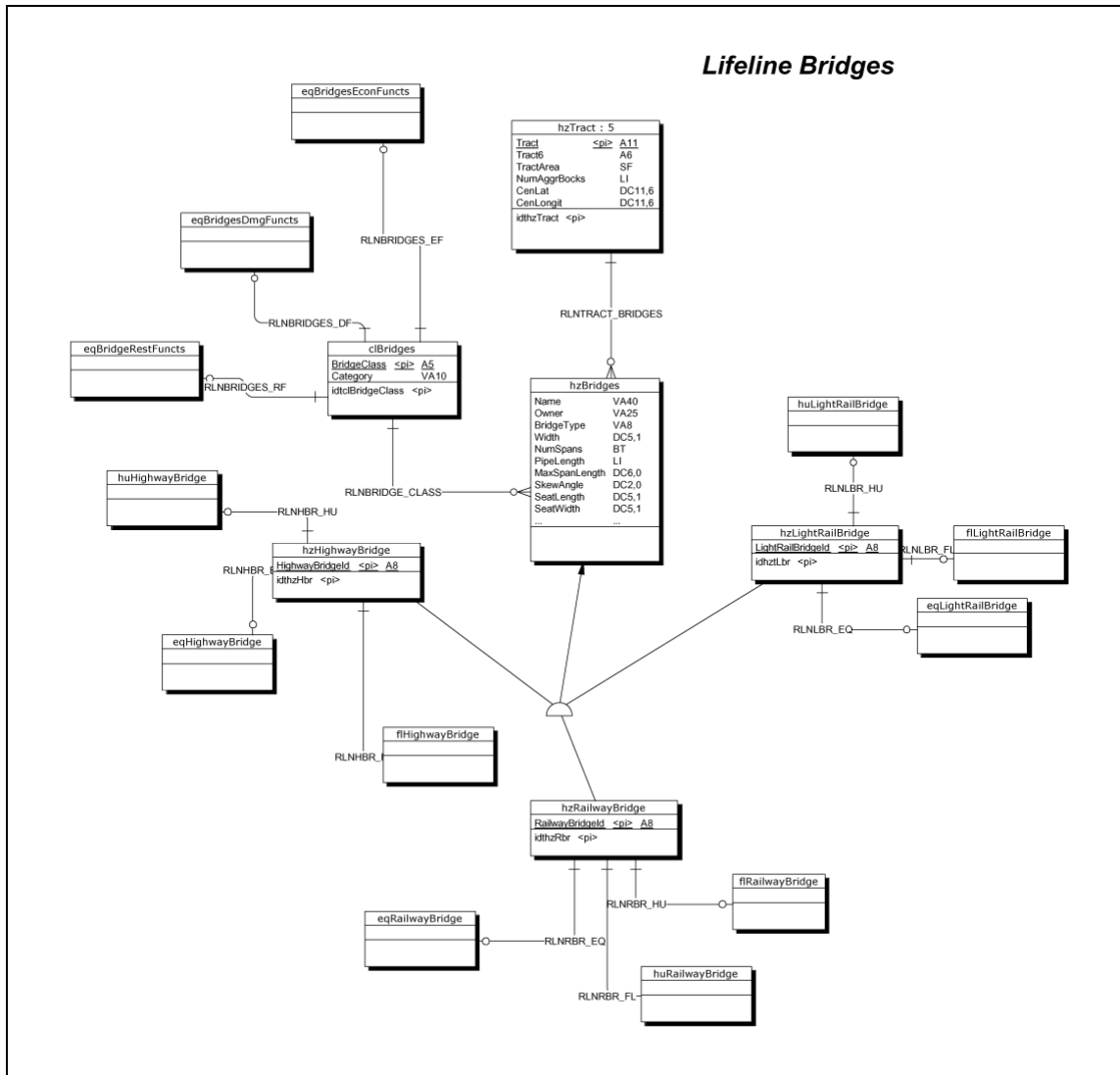


Figure 4-14. CDM for the Bridges Tables

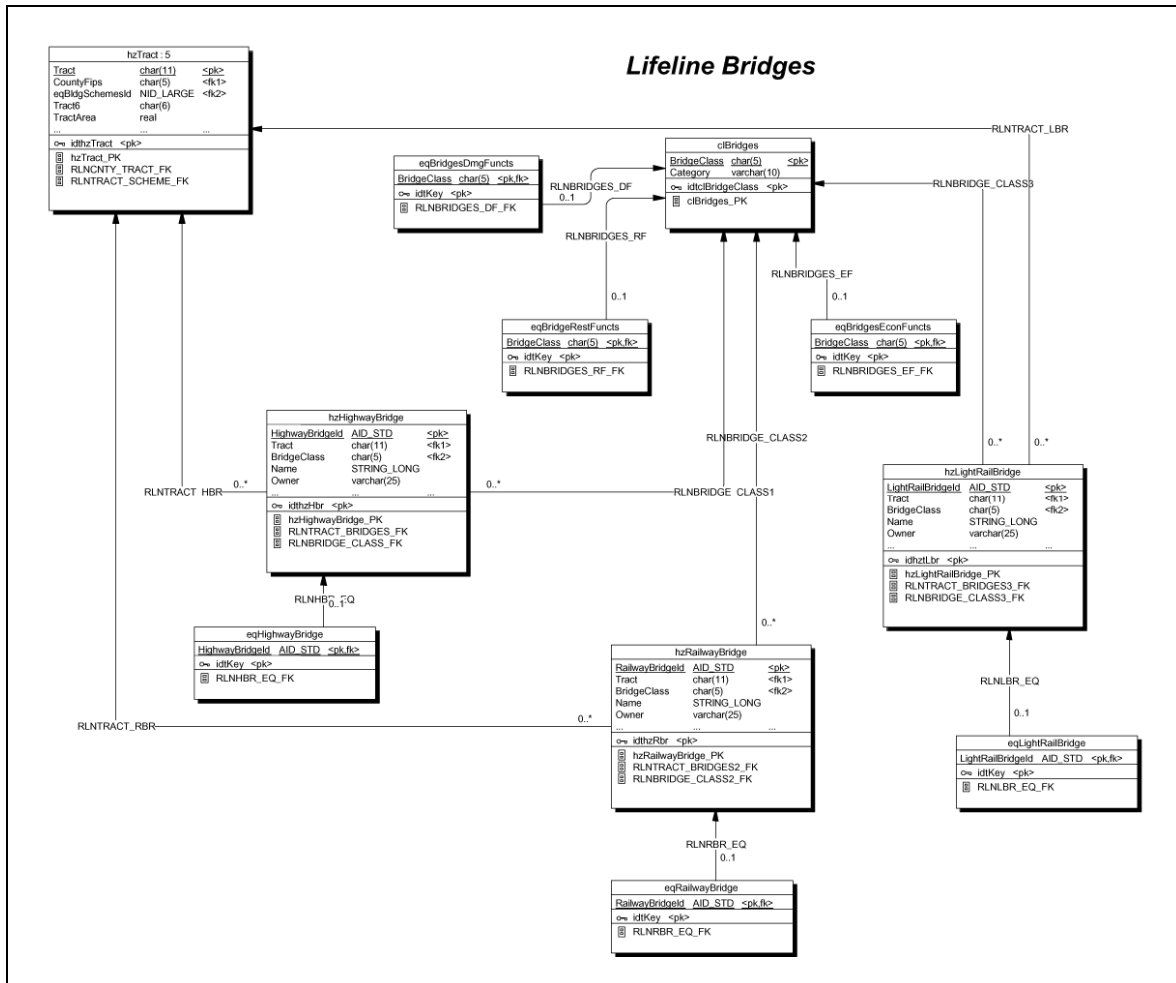


Figure 4-15. PDM for the Bridges Tables

4.1.8. Segments

The lifeline segments are the highways, railways, and the light rail segments. All the segments are split at the county level. In other words, if a given highway crosses two or more counties, then it is packaged as two or more records, one for each county. This is consistent with the approach used in HAZUS99-SR2.

Lifelines Segments

```

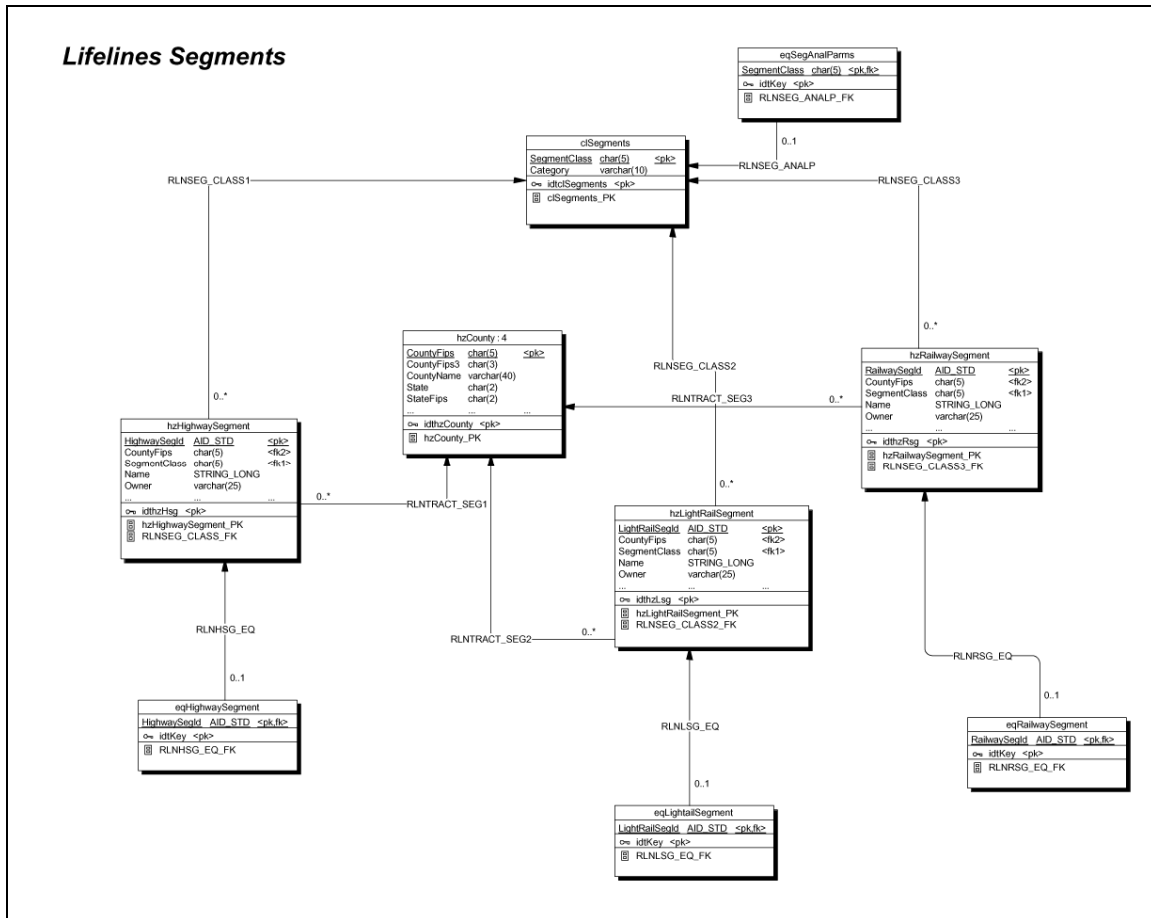
classDiagram
    class hzCounty {
        CountyFips <pi> A5
        CountyFips3 A3
        CountyName VA40
        State A2
        StateFips A2
        NumAggrTracts LI
        idhzCounty <pi>
    }
    class hzSegments {
        Name VA40
        Owner VA25
        PipeLength LI
        Traffic LI
        Cost MN
        Comment VA40
    }
    class cSegments {
        SegmentClass <pi> A5
        Category VA10
        idcSegments <pi>
    }
    class hzHighwaySegment {
        HighwaySegId <pi> A8
        NumLanes BT
        Pavement VA10
        Width DC5,1
        Capacity I
        idhzHsg <pi>
    }
    class hzRailwaySegment {
        RailwaySegId <pi> A8
        NumTracks BT
        idhzRsg <pi>
    }
    class hzLightRailSegment {
        LightRailSegId <pi> A8
        NumTracks BT
        idhzLsg <pi>
    }
    class eqHighwaySegment
    class huHighwaySegment
    class flHighwaySegment
    class eqLighttailSegment
    class flLighttailSegment
    class huLighttailSegment
    class eqRailwaySegment
    class flRailwaySegment
    class huRailwaySegment
    class eqSegAnalParms

    hzCounty -- hzSegments : RLNTRACT_SEG
    hzSegments -- cSegments : RLNSEG_CLASS
    hzHighwaySegment -- hzSegments
    hzRailwaySegment -- hzSegments
    hzLightRailSegment -- hzSegments
    eqHighwaySegment -- hzHighwaySegment : RLNHSG_EQ
    huHighwaySegment -- hzHighwaySegment : RLNHSG_HU
    flHighwaySegment -- hzHighwaySegment : RLNHSG_FL
    eqLighttailSegment -- hzLightRailSegment : RLNLSG_EQ
    flLighttailSegment -- hzLightRailSegment : RLNLSG_FL
    huLighttailSegment -- hzLightRailSegment : RLNLSG_HU
    eqRailwaySegment -- hzRailwaySegment : RLNRSG_EQ
    flRailwaySegment -- hzRailwaySegment : RLNRSG_FL
    huRailwaySegment -- hzRailwaySegment : RLNRSG_HU
    eqSegAnalParms -- cSegments : RLNSEG_ANALF
  
```

The diagram illustrates the structure of Lifelines Segments, showing the relationships between various segment classes and their parent/child classes. The classes are represented as boxes with their attributes and relationships as lines with labels.

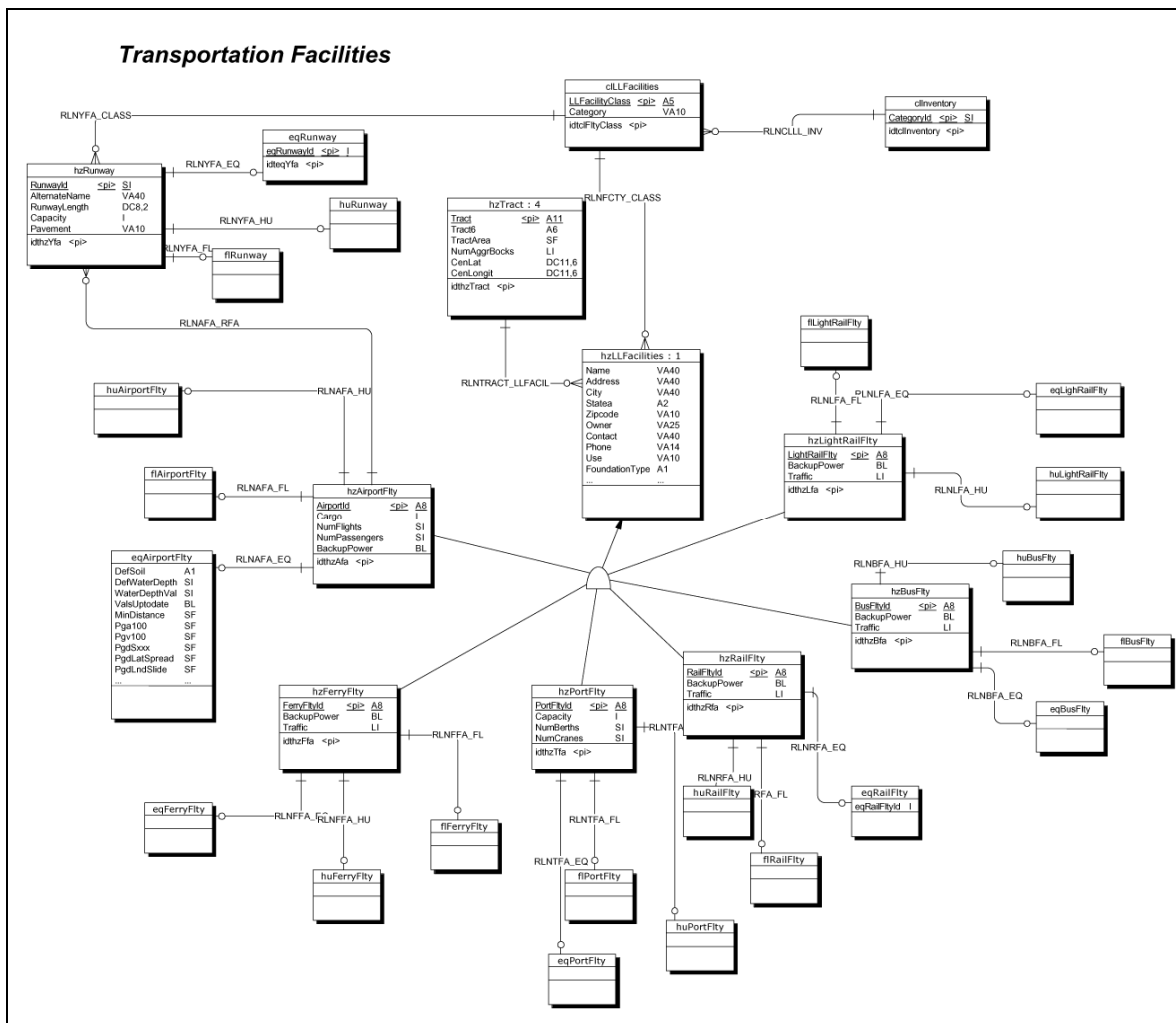
- hzCounty** (Class): Attributes include CountyFips, CountyFips3, CountyName, State, StateFips, NumAggrTracts, and idhzCounty. It is associated with **hzSegments** via the relationship **RLNTRACT_SEG**.
- hzSegments** (Class): Attributes include Name, Owner, PipeLength, Traffic, Cost, and Comment. It is associated with **cSegments** via the relationship **RLNSEG_CLASS**.
- cSegments** (Class): Attributes include SegmentClass, Category, and idcSegments. It is associated with **eqSegAnalParms** via the relationship **RLNSEG_ANALF**.
- hzHighwaySegment** (Class): Attributes include HighwaySegId, NumLanes, Pavement, Width, Capacity, and idhzHsg. It is associated with **hzSegments** via a generalization relationship (indicated by a solid line with an open arrowhead).
- hzRailwaySegment** (Class): Attributes include RailwaySegId, NumTracks, and idhzRsg. It is associated with **hzSegments** via a generalization relationship (indicated by a solid line with an open arrowhead).
- hzLightRailSegment** (Class): Attributes include LightRailSegId, NumTracks, and idhzLsg. It is associated with **hzSegments** via a generalization relationship (indicated by a solid line with an open arrowhead).
- eqHighwaySegment** (Class): Associated with **hzHighwaySegment** via the relationship **RLNHSG_EQ**.
- huHighwaySegment** (Class): Associated with **hzHighwaySegment** via the relationship **RLNHSG_HU**.
- flHighwaySegment** (Class): Associated with **hzHighwaySegment** via the relationship **RLNHSG_FL**.
- eqLighttailSegment** (Class): Associated with **hzLightRailSegment** via the relationship **RLNLSG_EQ**.
- flLighttailSegment** (Class): Associated with **hzLightRailSegment** via the relationship **RLNLSG_FL**.
- huLighttailSegment** (Class): Associated with **hzLightRailSegment** via the relationship **RLNLSG_HU**.
- eqRailwaySegment** (Class): Associated with **hzRailwaySegment** via the relationship **RLNRSG_EQ**.
- flRailwaySegment** (Class): Associated with **hzRailwaySegment** via the relationship **RLNRSG_FL**.
- huRailwaySegment** (Class): Associated with **hzRailwaySegment** via the relationship **RLNRSG_HU**.

Figure 4-16. CDM for Segments Tables



4.1.9. Transportation Facilities

There are six types of transportation facilities: airports, ferries, ports, railway facilities, bus facilities, and light rail facilities.



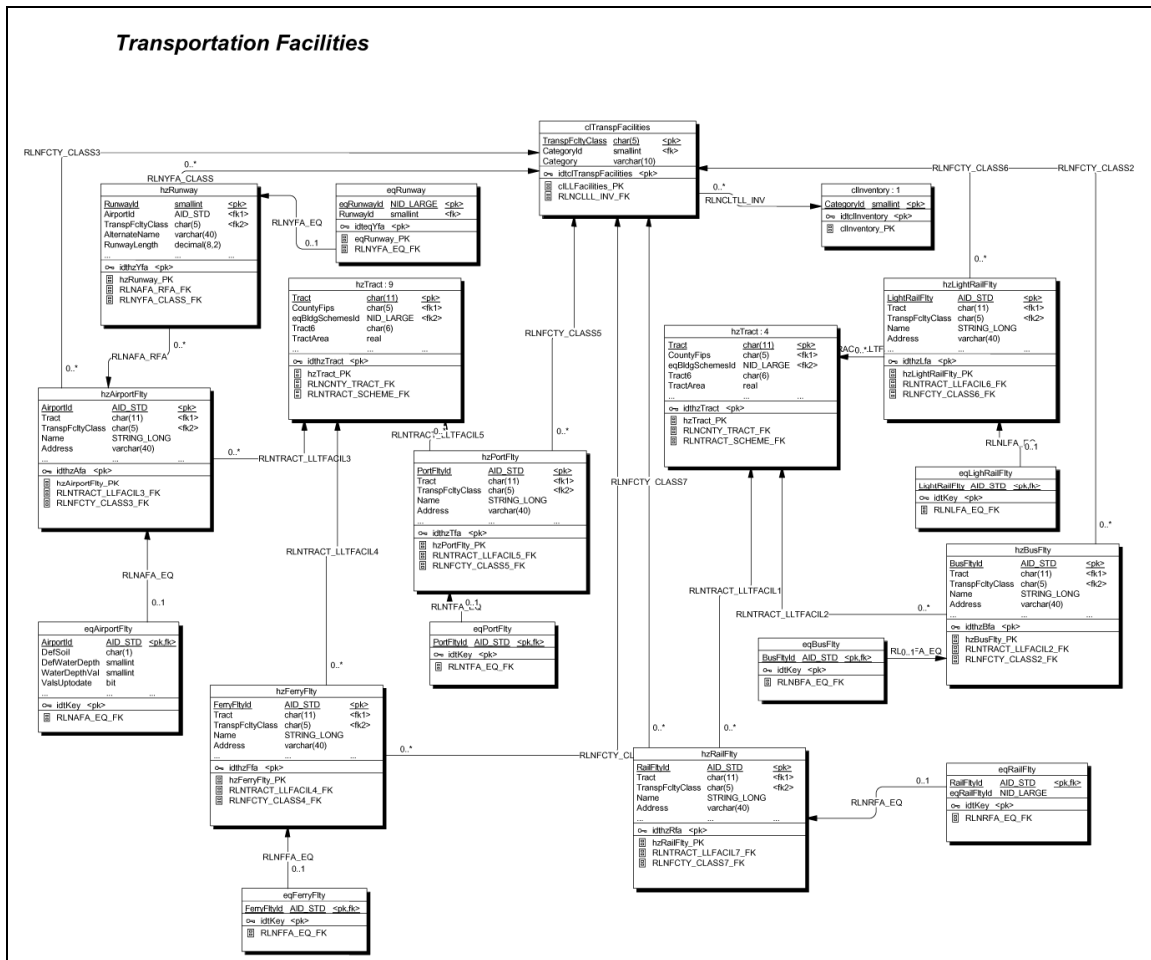


Figure 4-19. PDM for the Transportation System Facilities

4.1.10. Utility Facilities

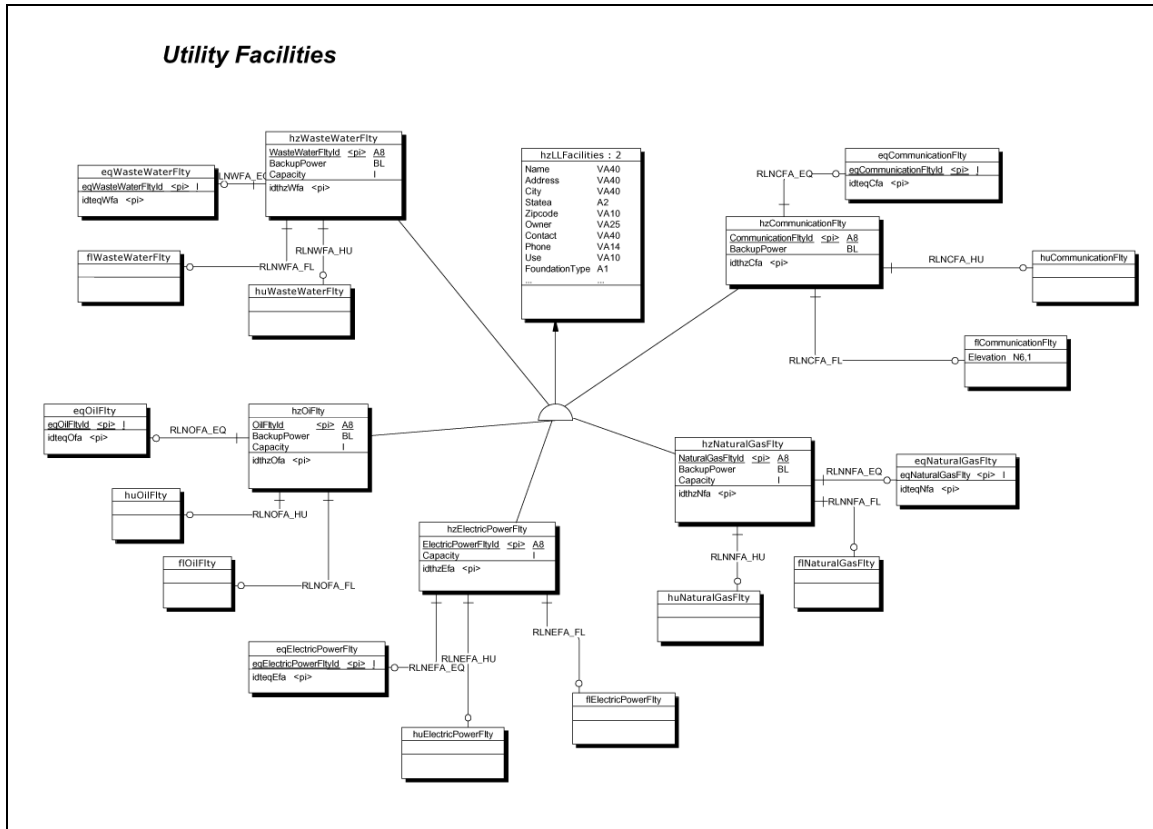
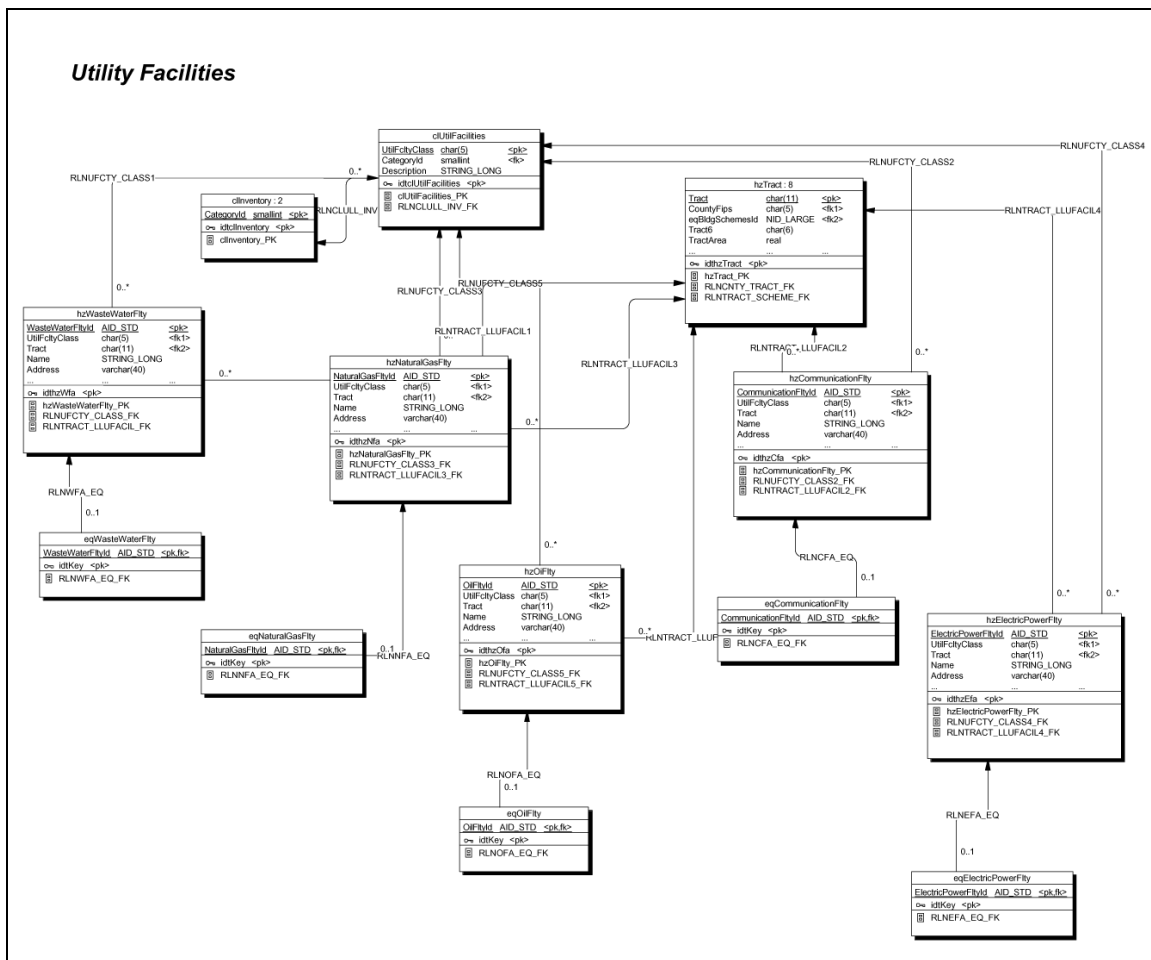


Figure 4-20. CDM for the Utility System Facilities



4.1.11. Pipelines

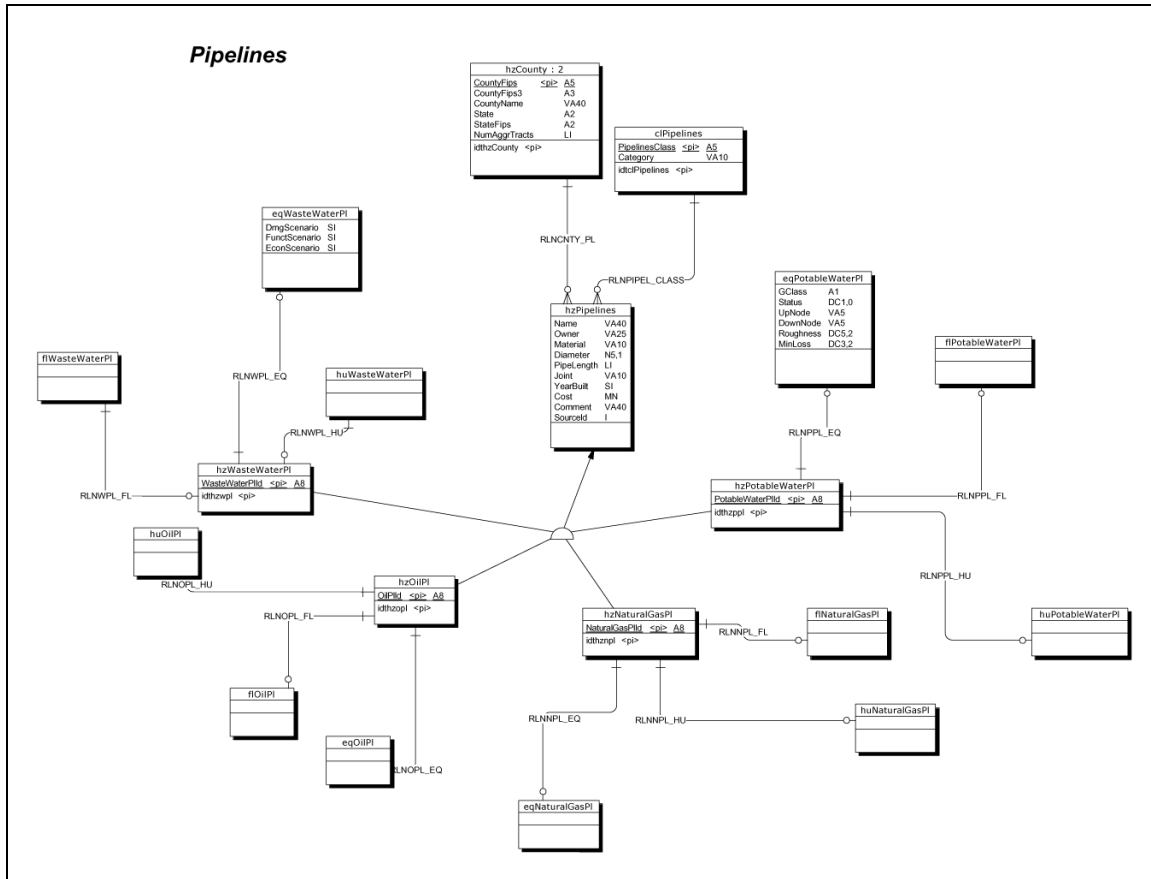
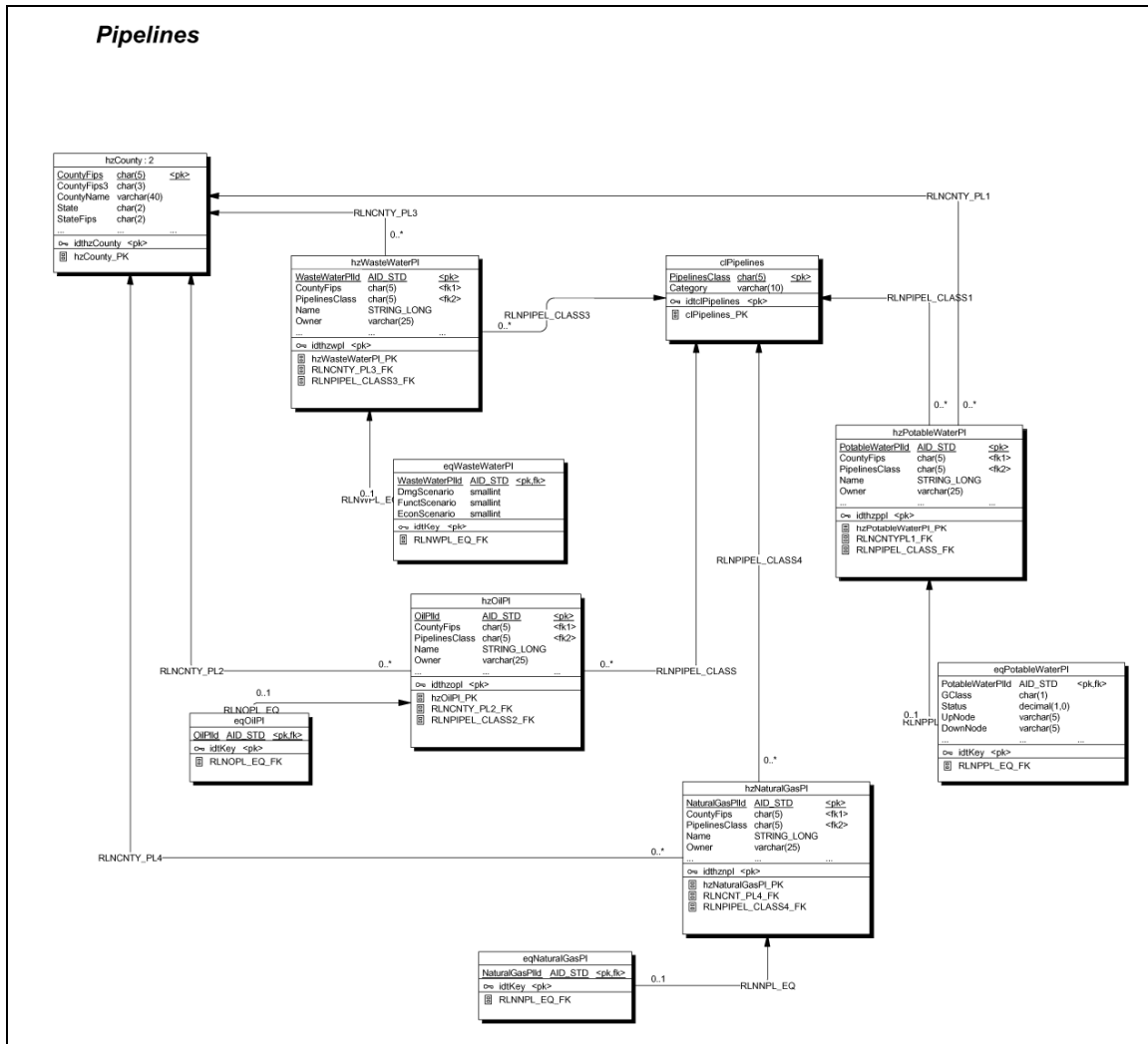


Figure 4-22. CDM for the Pipelines Tables



4.1.12. Tunnels

Tunnels

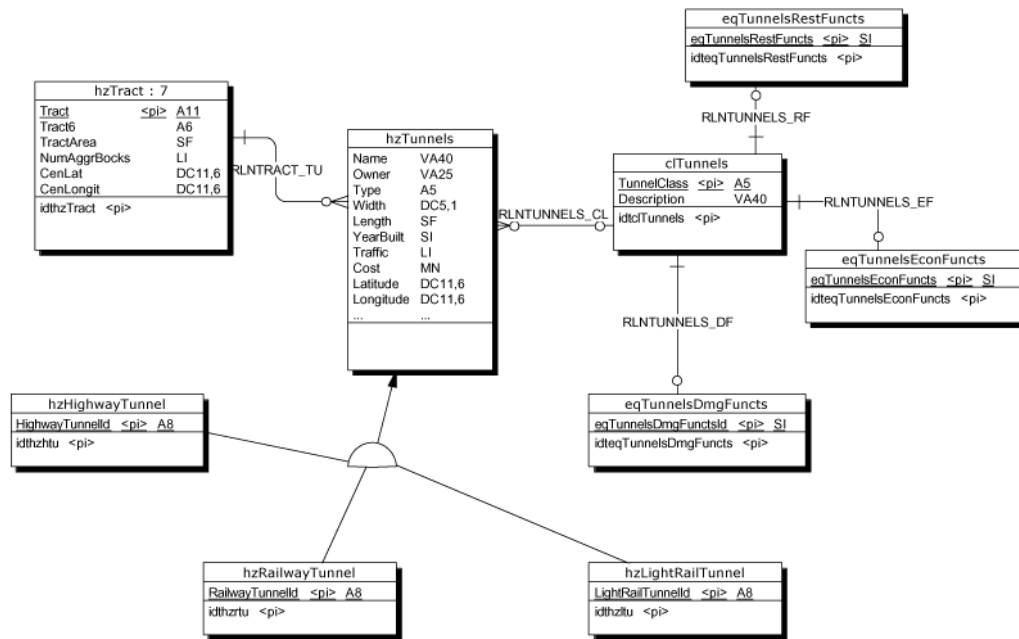


Figure 4-24. CDM for the Tunnels Tables

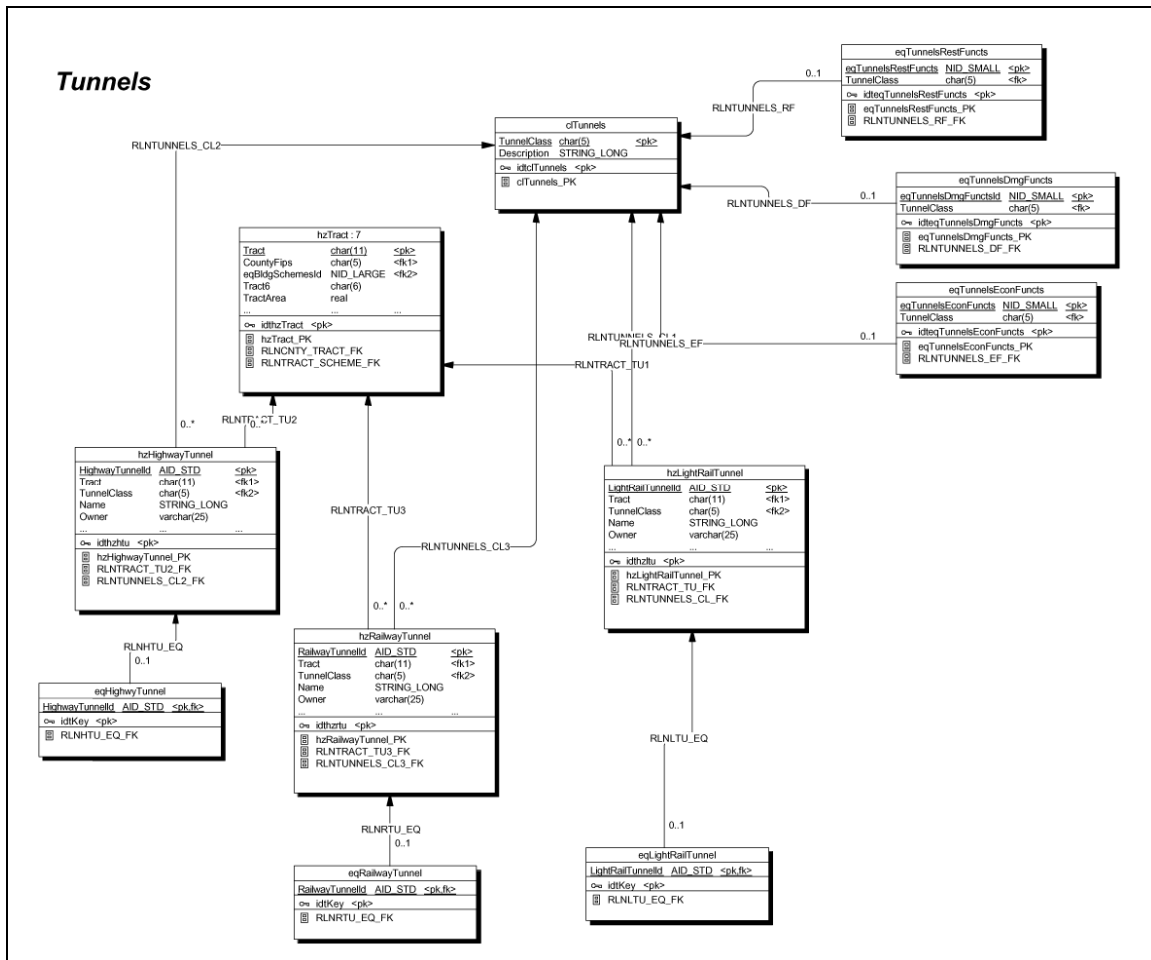


Figure 4-25. PDM for the Tunnels Tables

4.1.13. Guidelines for Group Development of the HAZUS- MH Database

The development of the HAZUS-MH databases (system database and the template database) is a joint effort among the three contractors. As such, rules must be observed so that the development of the data models in parallel by the three parties would not cause clashes --which will be difficult to fix later on.

Below is a set of the guidelines that need to be observed by all parties:

- Follow the generally accepted database modeling practices, specifically the normalization rules
- Fully understand the base data models developed by DTI

- Follow the same naming conventions outlined in this document
- Use the same domains across all the tables. DTI shall add any missing domain definitions. To have the domain definitions propagate across different data models/files, make use of the shortcut feature in PowerDesigner
- Each contractor is responsible for the development of the data model for its hazard. In addition, DTI is responsible for all the common objects. This can be summarized as follows:

Table 4-5. Distribution of Tasks by Table Type

Table Type	Responsibility	Description
hz*	DTI	Common tables
eq*	DTI	Earthquake-related objects
fl*	EQE	Flood-related tables
hu*	ARA	Hurricane-related tables
other (cl*,...)	All	Other tables like classification tables are a joint effort with DTI responsible for final integration in global model.

- The responsibilities for the development of the HAZUS-MH data model can be divided into three broad categories: definition, implementation, and scope.
 - The definition includes the detailed description of all the attributes related to a given object. The attribute types vary based on the object defined; for example, for a table, the attributes will be the column, their types, names, nullability, and so on. For a stored procedure, the attribute will be the source code.
 - The implementation includes making the physical changes to the data model file itself (in PowerDesigner 8), including all inherent data modeling tasks (documentation/comments, domains, integrity rules, CDM vs. PDM merging, triggers implementation, stored procedures, etc.).
 - The scope defines at what level the implementation is carried out. During the development phase, there would be four data models (common + 1 for each hazard), but as a deliverable, all four need to be merged into a single data model. Each contractor is responsible for developing the data

for his relevant hazard. DTI is responsible for merging the three hazard-specific models into a HAZUS-MH global data model.

Table 4-6. Contractors Responsibilities vs. Tables Type.

Database	Table Definition	Responsible Party for Definition	Responsible Party for Implementation
syHAZUS (system database)	sy*	DTI	DTI
	eq*	DTI	DTI
	fl*	EQE	DTI
	hu*	ARA	DTI
tlHAZUS (template database)	hz*	DTI	DTI
	eq*	DTI	DTI
	fl*	EQE	EQE
	hu*	ARA	ARA

- Following data modeling best practices, all HAZUS-MH data models must be self-defined. This implies that all the databases and tables used by the program are visible within the data diagram. A direct consequence is that no table should be added/created at run-time (if the model shows that database x has n tables, then that should be always the case)
- The size/width of all the columns should be set such that it is just the right amount; if only two bytes are needed to hold a given attribute, then two bytes must be used and no more. This is due to the major constraint of MSDE on the size of databases, which cannot exceed 2 GB. In HAZUS-MH, a database is a study region. In all cases, using a double instead of a real or an integer instead of a tinyint means that the maximum possible size of a study region is cut in half
- The medium of exchange between the three contracts should be always the native PowerDesigner files (.pdm & .cdm). If relevant, all associated documentation should be also exchanged. DTI will then merge the local data models with the global data model, and send back the latter to each contractor with the relevant complete installation scripts

- A suggested workflow on the template database for each contractor is as follows:
 - Start from the current DTI global model
 - Take out all the tables except the hz*, and hazard you're working on (for example, ARA would delete eq*, and fl*)
 - Uncheck the generate option on the hz* tables. We need to keep the hz* tables so that the reference/constraint from the hazard specific table can be generated correctly
 - Add in all your objects
- A suggested workflow on the system database for each contractor is as follows:
 - Implement your hazard-specific object in an independent data model
 - Pass the file to DTI for full integration into the global system data model
 - Because the system database has very limited hazard-specific objects, the effort here should be minimal

4.2. Interface Design (Presentation Services Layer)

4.2.1. The Study Region Aggregation Wizard

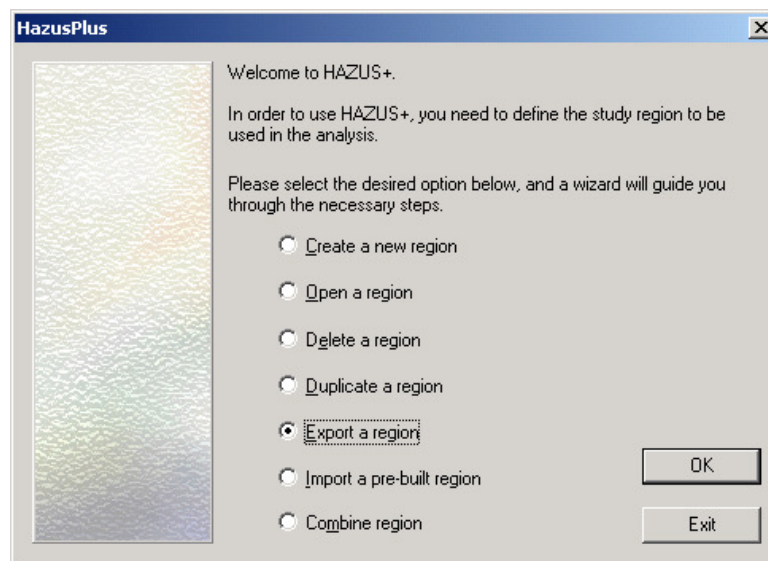


Figure 4-26. Startup Screen for the Study Region Aggregation Wizard

The functionality of the study region aggregation wizard can be grouped into seven major groups:

- Creating new study regions

- Opening an already created study region
- Deleting an existing study region
- Duplicate (making a copy) of an existing study region
- Export/archive an existing study region
- Importing (an exporting) a study region
- Combining two study regions to create a third larger study region (this option is pending budget approval)

4.2.1.1. Creating a Region

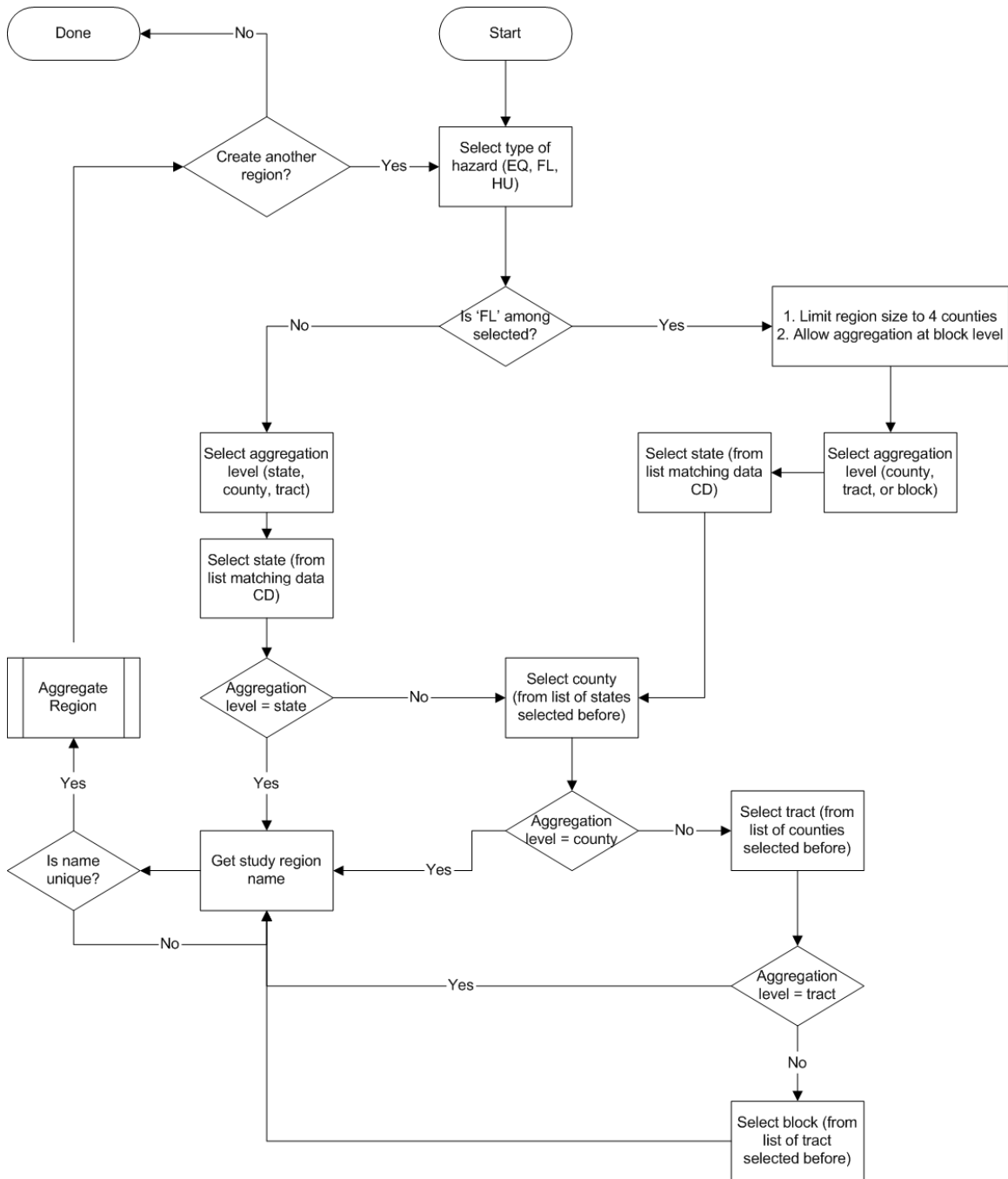


Figure 4-27. Creating a Region Process Flowchart

4.2.1.2. Open a Region

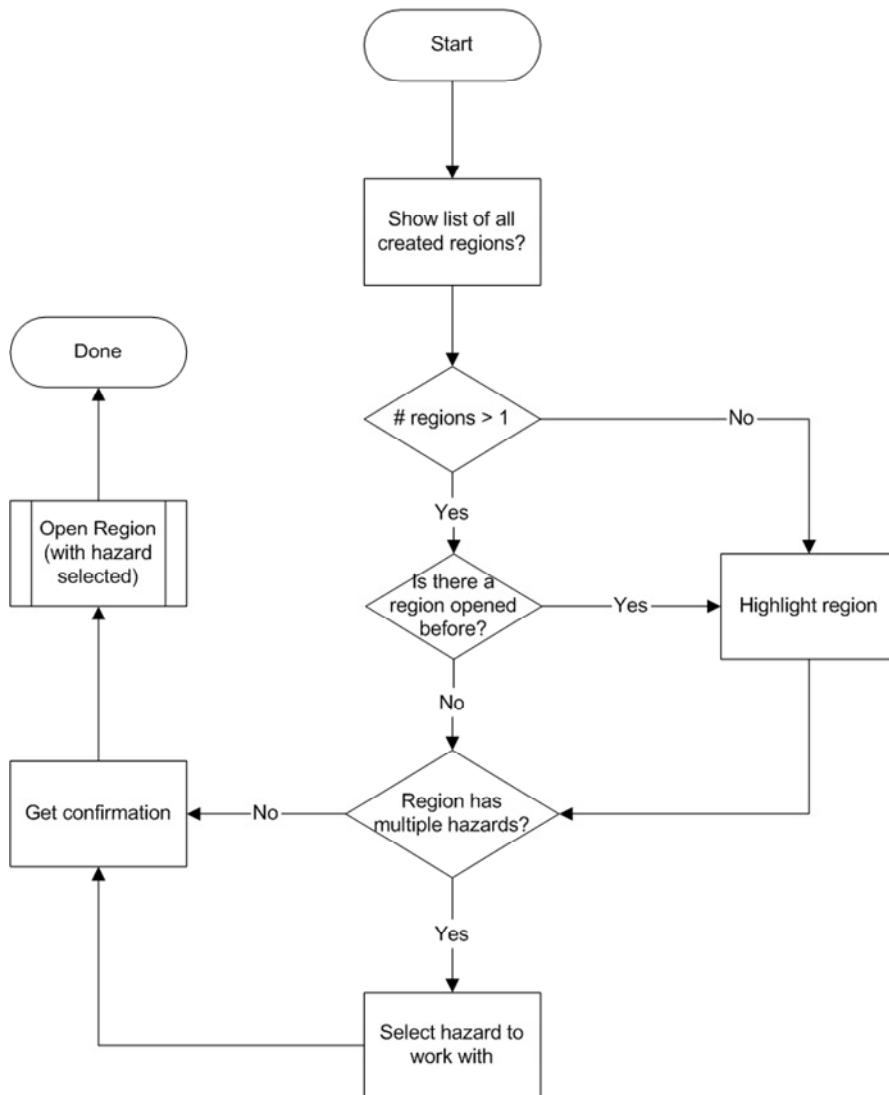


Figure 4-28. Opening a Region Process Flowchart

4.2.1.3. Delete a Region

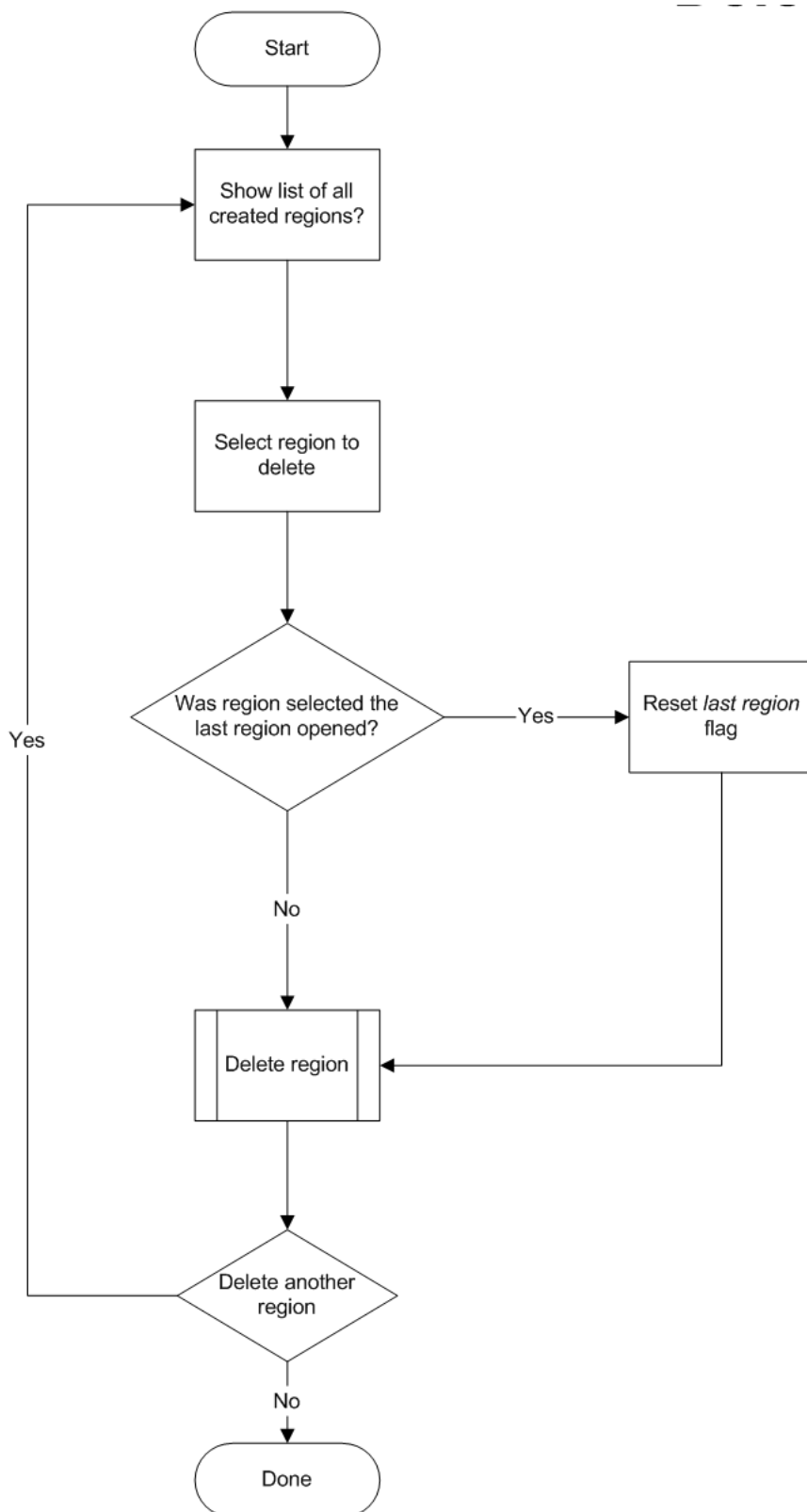


Figure 4-29. Deleting a Region Process Flowchart

4.2.1.4. Duplicate a Region

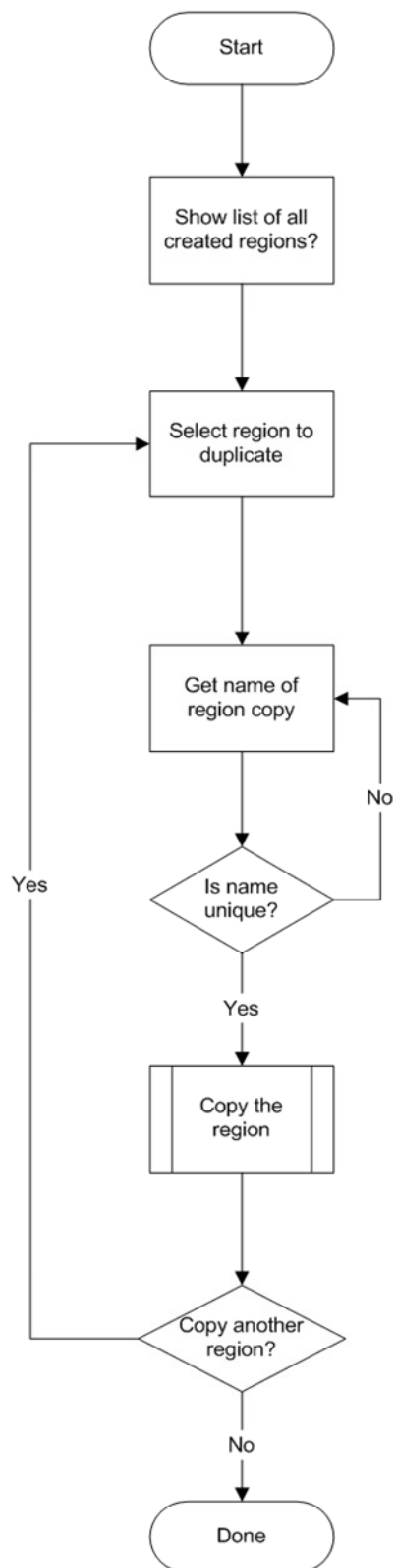


Figure 4-30. Duplicating a Region Process Flowchart

4.2.1.5. Export a Region

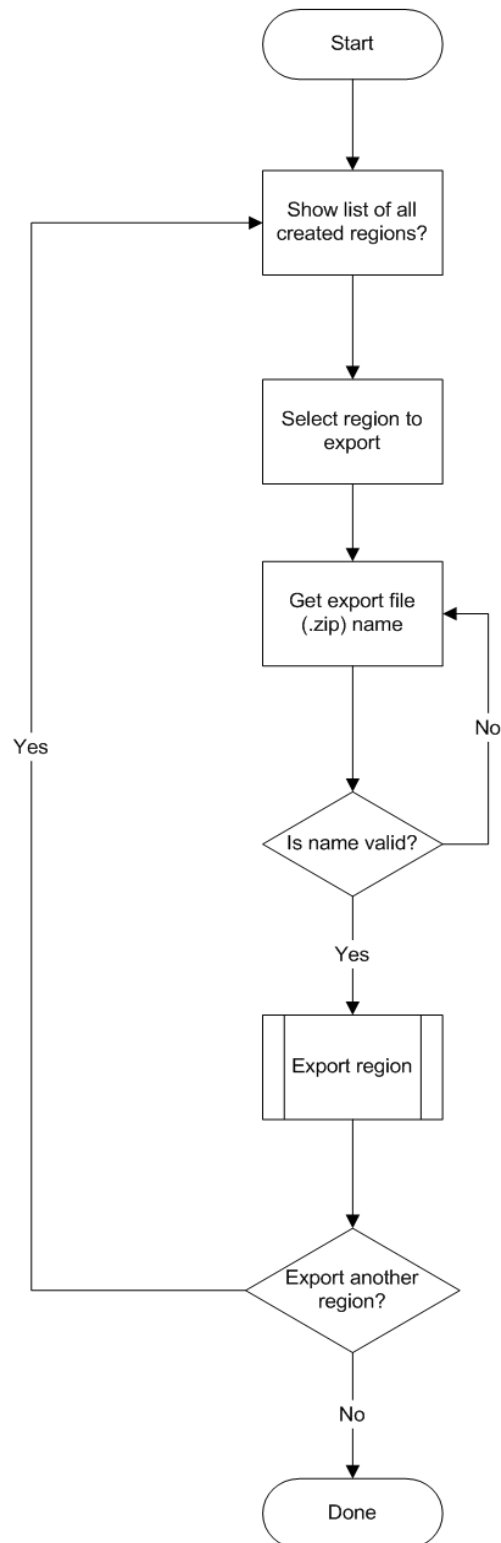


Figure 4-31. Exporting a Region Process Flowchart.

4.2.1.6. Import a Region

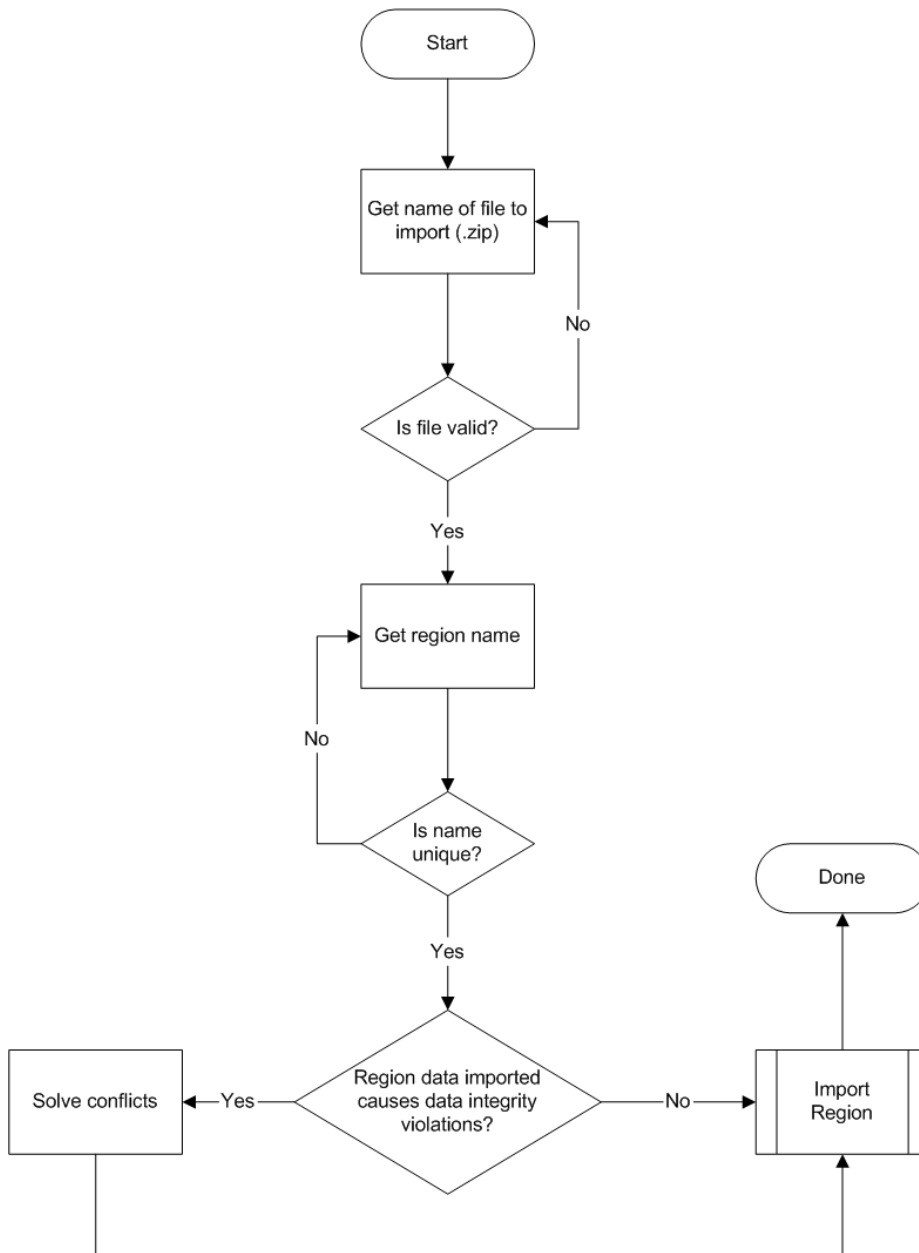


Figure 4-32. Importing a Region Process Flowchart.

4.2.1.7. Combine a Region

This option is pending budget consideration.

NIBS has expressed an interest in expanding the aggregation capability by allowing the HAZUS-MH user to change the boundaries of a previous created region. To control expenses, DTI proposes limiting the region changes to additions only. (i.e. the user can only add blocks/tracts to an existing region but cannot subtract, thus the concept of region combination.)

The feature will work as follows:

1. User creates a region (Region 1)
2. At a later time, user wants to expand Region 1 to include additional blocks, tracts, counties, or states, then the user creates a second region with the additional area (Region 2)
3. From the region wizard, user selects a newly added option 'Combine Regions' which prompts for 2 regions, and creates a third region which is the sum of the two.

The combining process described in Step C can be repeated as many times as needed (i.e. the combination process could be done at any time for any two previously created regions.)

There is a limitation that only regions with same hazard type can be combined. For example if Region 1 was aggregated initially with earthquake data, and Region 2 was aggregated with hurricane data, then the two regions cannot be combined⁶.

This could also be an advantage for a very large region where the analyses are run on several computers and then the sub-regions merged after analysis.

⁶ This limitation is as per the current HAZUS03 requirements.

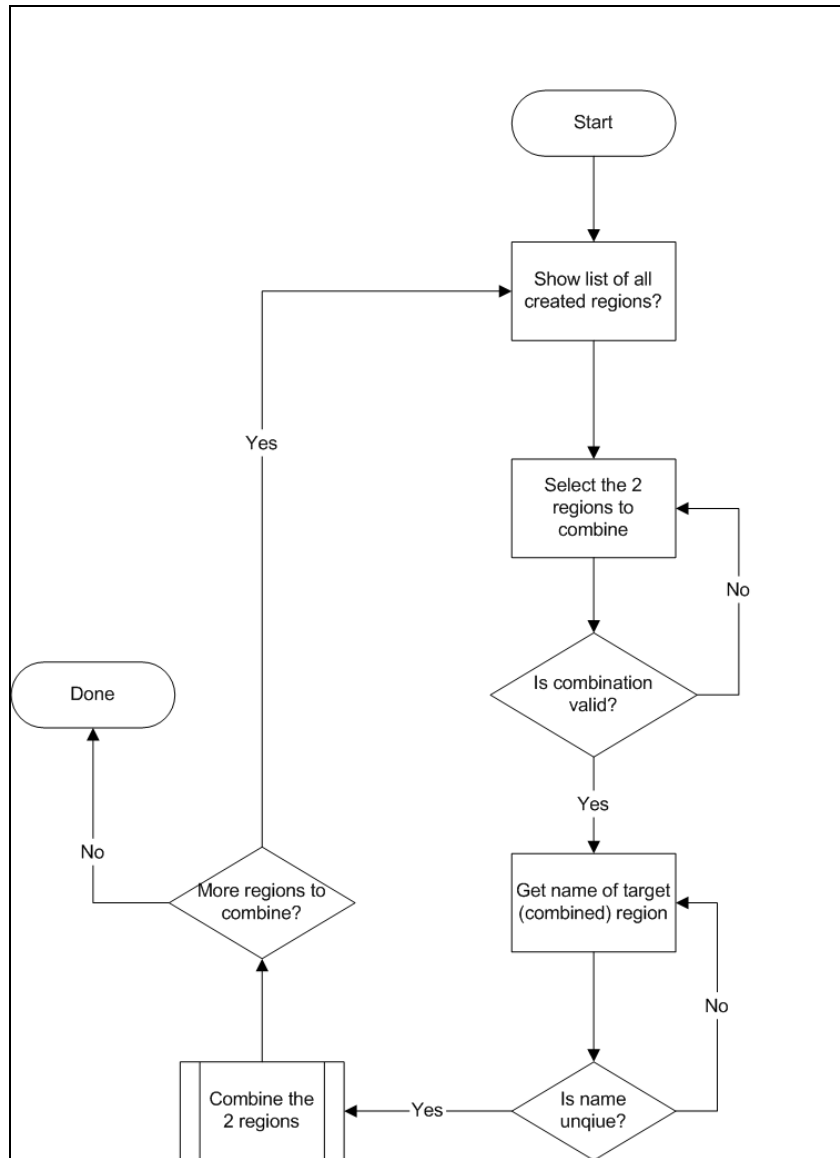


Figure 4-33. Combine Region Process Flowchart.

5. Dependency Description

5.1. Mapping Engine

The following diagrams show relationships among object roles such as the set of messages exchanged among the objects to achieve an operation or result thus describing the control flows among various classifiers involved in the usage of the mapping engine components. Every diagram has to be read as follows:

- The flow is always from left to right as indicated by the direction arrows on the association roles.
- The leftmost classifier is always the starting point for the flow and the rightmost classifier is always the end point.
- Every association starting at the left most classifier represents the flow for one complete operation involving the components at all the application tiers.

5.1.1. Inventory Mapping

The Inventory Mapping collaboration diagram shows how the various objects interact with each other for every mapping operation. The diagram also indicates major methods that are used in the process.

The various objects shown in the diagram are the instances of various components or interfaces discussed above. This is a collaboration between the HazusEq.mxd, Data Browser, IMapPolygonFeatures, IMapPOintFeatures, IMapPlineFeatures, IMapDetails and IThMapDetails objects. Following table describes the various objects used in the diagram:

Table 5-1. Inventory Mapping Objects

Classifier Object	Instance Of	Description
EqMapView	HazusEq.mxd	Represents the current study region opened for earthquake hazard
InvtdDataBrowser	Data Browser	Represents the dialog box

Classifier Object	Instance Of	Description
		in which the user views the table
PrintLayout	IMapPolygonFeatures	Created by EqMapView this object handles all the print layout operations
BoundaryMap	IMapPolygonFeatures	Created by EqMapView to map various study region boundary maps
DataMap	IMapPolygonFeatures	Created by EqMapView to map various datamap based on corresponding unique value themes
EssentialFacility	IMapPointFeatures	Created by InvtDataBrowser to map various Essential Facilities
HighPotentialLossFacility	IMapPointFeatures	Created by InvtDataBrowser to map various High Potential Loss Facilities
AEBM/UserDefFacility	IMapPointFeatures	Created by InvtDataBrowser to map the AEBM and User Defined structures
Bridges	IMapPointFeatures	Created by InvtDataBrowser to map various types of Bridges
Tunnel	IMapPointFeatures	Created by InvtDataBrowser to map various types of Tunnels
Highway/RailTrack	IMapPlineFeatures	Created by InvtDataBrowser to map various types of Highways and Railtracks
Pipeline	IMapPlineFeatures	Created by InvtDataBrowser to map various types of

Classifier Object	Instance Of	Description
		Pipelines
GeographicData	IMapDetails	Created by the middle tier objects to get geographic data objects
GeographicTheme	IThMapDetails	Created by the middle tier objects to get geographic data objects

5.1.2. Mapping General Building Stock Inventory

This collaboration diagram shows how the various objects interact with each other for mapping of general building stock data. The diagram also indicates major methods that are used in the process.

This is a collaboration between the Data Browser, IMapBuildingData and IThMapDetails objects. Following table describes the various objects used in the diagram:

Table 5-2. Mapping General Building Stock Results

Classifier Object	Instance Of	Description
InvtDataBrowser	Data Browser	Represents the dialog box in which the user views the table
SquareFootage	IMapBuildingData	Created by InvtDataBrowser to create a range theme using the values selected by the user and the census tract boundary map
BuildingCount	IMapBuildingData	Created by InvtDataBrowser to create a range theme using the values selected by the user and the census tract boundary map
DollarExposure	IMapBuildingData	Created by InvtDataBrowser to create a range theme using the values selected by the user and the census tract boundary map
Demographics	IMapBuildingData	Created by InvtDataBrowser to create a range theme using the values selected by the user and the census tract boundary map
GeographicTheme	IThMapDetails	Created by the middle tier objects to get geographic data objects

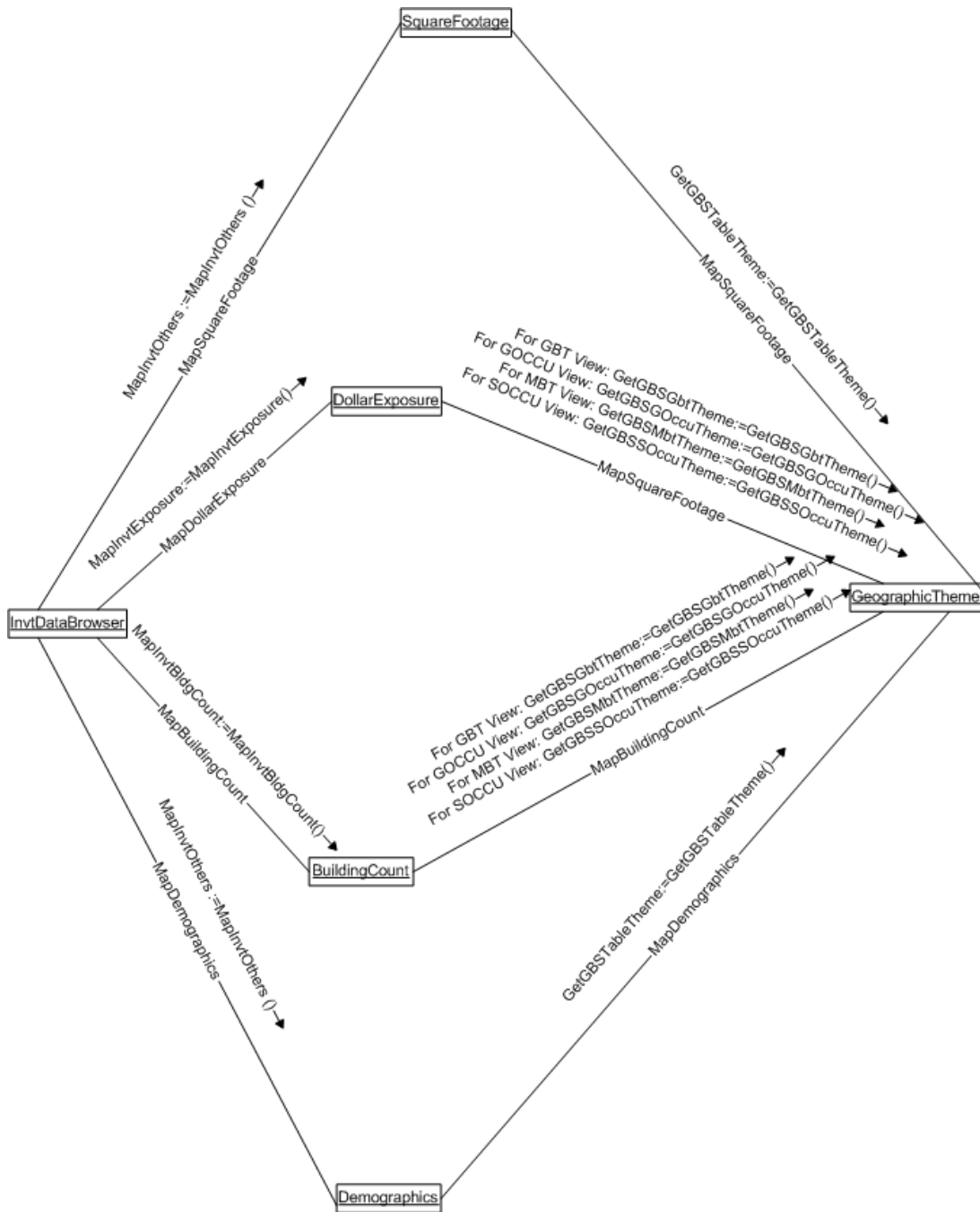


Figure 5-2. Mapping General Building Stock Collaboration Diagram

5.1.3. Inventory Editing

The Inventory Editing collaboration diagram shows how the various objects interact with each other for every editing operation. The diagram also indicates major methods that are used in the process.

The various objects shown in the diagram are the instances of various components or interfaces discussed above. This is a collaboration between the HazusEq.mxd, IMapPolygonFeatures, IMapPOintFeatures, IMapPlineFeatures, and IMapDetails objects. Following table describes the various objects used in the diagram:

Table 5-3. Inventory Editing Objects

Classifier Object	Instance Of	Description
EqMapView	HazusEq.mxd	Represents the current study region opened for earthquake hazard
EssentialFacility	IMapPointFeatures	Created by EqMapView to edit and import various EF
HighPotentialLossFacility	IMapPointFeatures	Created by EqMapView to edit and import various HPLF
AEBM/UserDefFacility	IMapPointFeatures	Created by EqMapView to edit and import the AEBM and User Defined structures
Bridges	IMapPointFeatures	Created by EqMapView to edit and import various types of Bridges
Tunnel	IMapPointFeatures	Created by EqMapView to edit and import various types of Tunnels
Highway/RailTrack	IMapPlineFeatures	Created by EqMapView to import various types of Highways and Railtracks
Pipeline	IMapPlineFeatures	Created by EqMapView to import various types of Pipelines
GeographicData	IMapDetails	Created by the middle tier objects to get geographic data objects

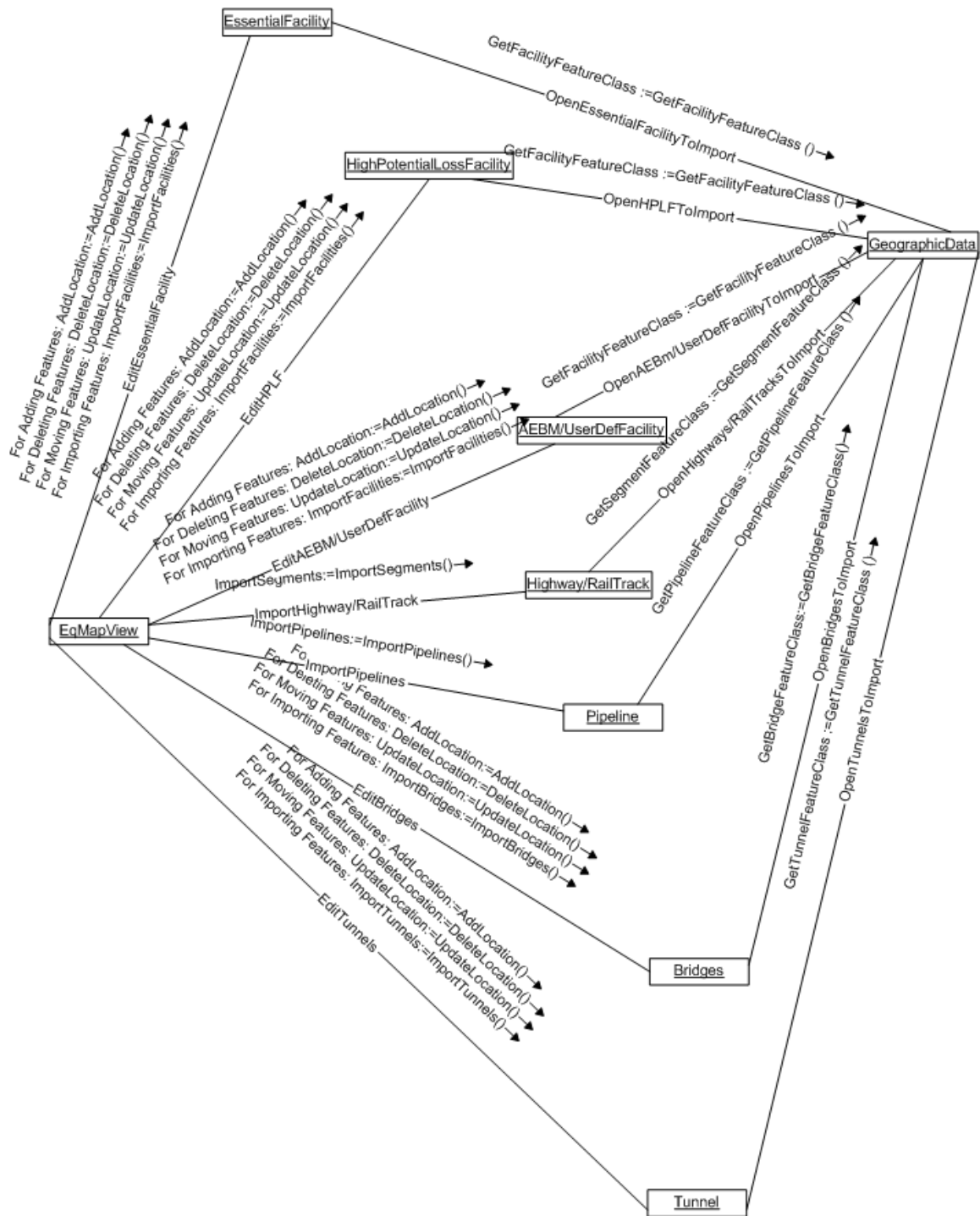


Figure 5-3. Inventory Editing Collaboration Diagram

5.1.4. Mapping General Building Stock Results

This collaboration diagram shows how the various objects interact with each other for mapping of general building stock analysis results. The diagram also indicates major methods that are used in the process.

This is a collaboration between the Data Browser, IMapBuildingData and IThMapDetails objects. Following table describes the various objects used in the diagram:

Table 5-4. Mapping General Building Stock Results Objects (Part 1)

Classifier Object	Instance Of	Description
ResultsBrowser	Data Browser	Represents the dialog box in which the user views the analysis results
EconomicLoss	IMapBuildingData	Created by ResultsBrowser to create a range theme using the values selected by the user and the census tract boundary map
DamagedBuildingCount	IMapBuildingData	Created by ResultsBrowser to create a range theme using the values selected by the user and the census tract boundary map
Casualties	IMapBuildingData	Created by ResultsBrowser to create a range theme using the values selected by the user and the census tract boundary map
GBSDamage	IMapBuildingData	Created by ResultsBrowser to create a range theme using the values selected by the user and the census tract boundary map
GeographicTheme	IThMapDetails	Created by the middle tier

Classifier Object	Instance Of	Description
		objects to get geographic data objects

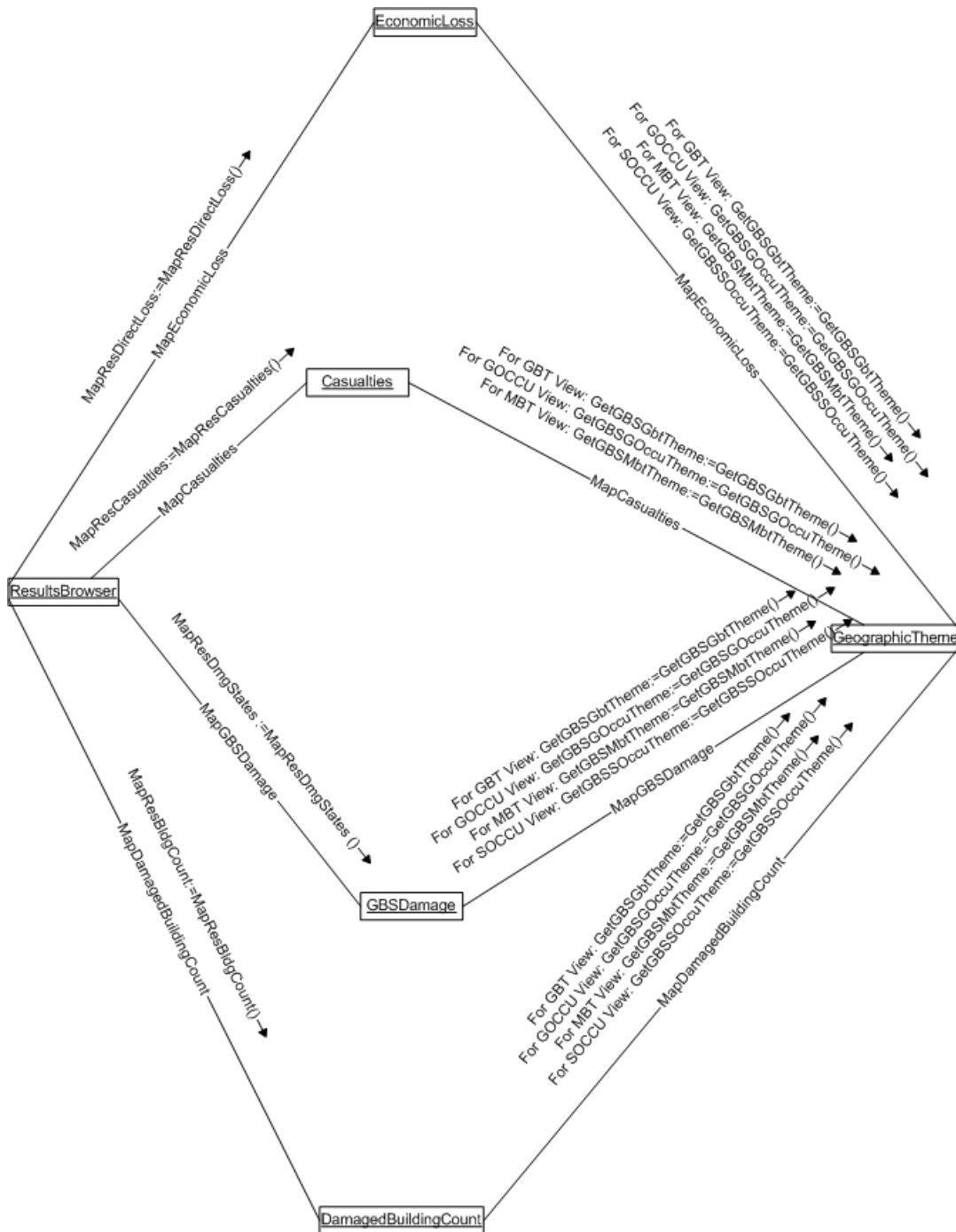


Figure 5-4. Mapping General Building Stock Results Collaboration Diagram (Part 1).

The following table describes the various objects used in the next part of the diagram.

Table 5-5. Mapping General Building Stock Results Objects (Part 2)

Classifier Object	Instance Of	Description
ResultsBrowser	Data Browser	Represents the dialog box in which the user views the analysis results
GroundMotion/Failure	IMapBuildingData	Created by ResultsBrowser to create a range theme using the values selected by the user and the census tract boundary map
GroundMotion/FailureCountours	IMapBuildingData	Created by ResultsBrowser to create a range theme using the values selected by the user and the census tract boundary map
Inundation	IMapBuildingData	Created by ResultsBrowser to create a range theme using the values selected by the user and the census tract boundary map

FireFollowingEarthquake	IMapBuildingData	Created by ResultsBrowser to create a range theme using the values selected by the user and the census tract boundary map
Debris	IMapBuildingData	Created by ResultsBrowser to create a range theme using the values selected by the user and the census tract boundary map
Shelter	IMapBuildingData	Created by ResultsBrowser to create a range theme using the values selected by the user and the census tract boundary map
GeographicTheme	IThMapDetails	Created by the middle tier objects to get geographic data objects

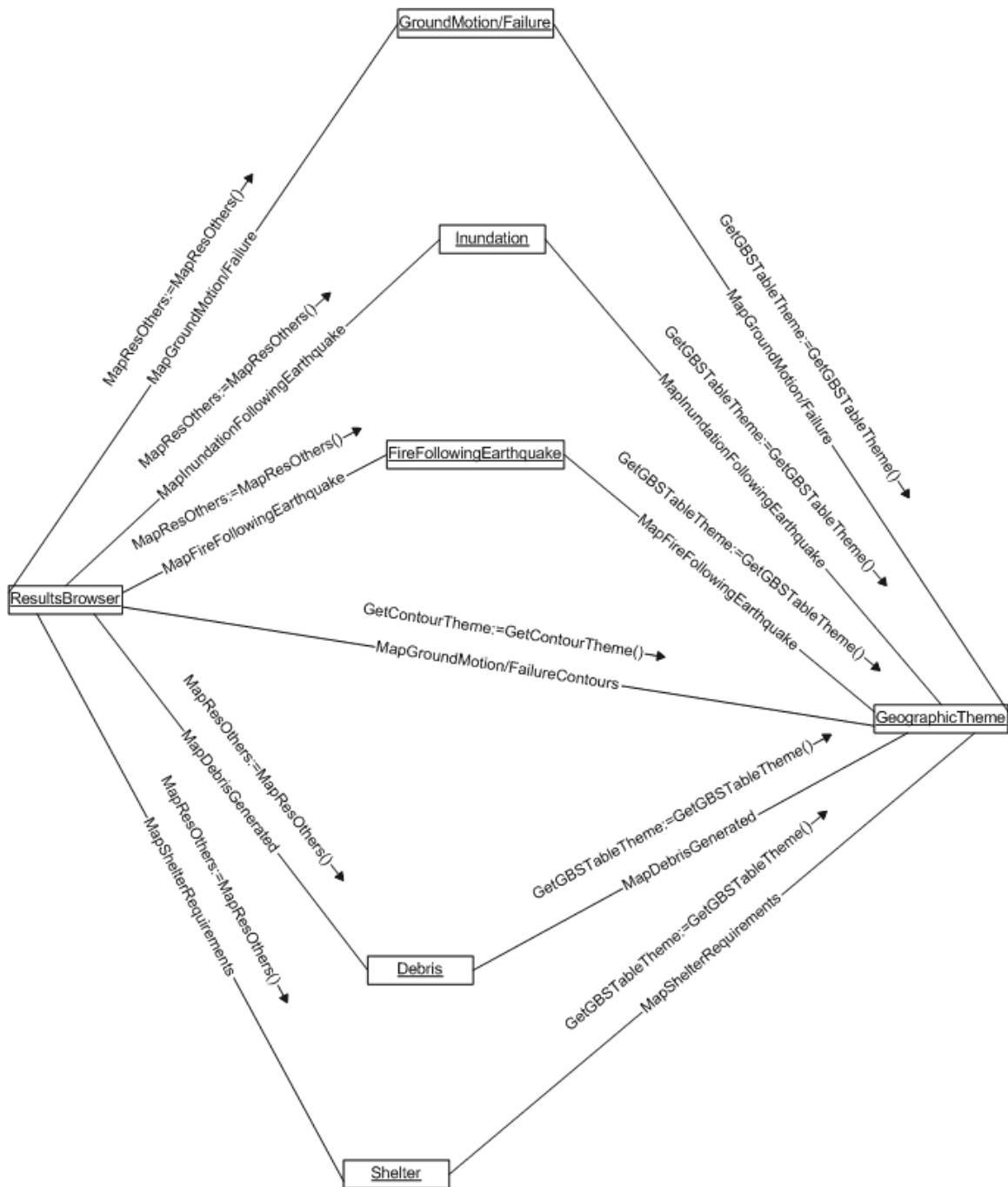


Figure 5-5. Mapping General Building Stock Results Collaboration Diagram (Part 2)

5.1.5. Mapping Inventory Results

The Inventory Results Mapping collaboration diagram shows how the various objects interact with each other for every results mapping operation. The diagram also indicates major methods that are used in the process.

The various objects shown in the diagram are the instances of various components or interfaces discussed above. This is a collaboration between the Data Browser, IMapPolygonFeatures, IMapPOintFeatures, IMapPlineFeatures and IThMapDetails objects. Following table describes the various objects used in the diagram:

Table 5-6. Mapping Inventory Results Objects

Classifier Object	Instance Of	Description
InvtDataBrowser	Data Browser	Represents the dialog box in which the user views the analysis results
EssentialFacility	IMapPointFeatures	Created by InvtDataBrowser to create a range theme using the values selected by the user and various Essential Facilities
HighPotentialLossFacility	IMapPointFeatures	Created by InvtDataBrowser to create a range theme using the values selected by the user and various High Potential Loss Facilities
AEBM/UserDefFacility	IMapPointFeatures	Created by InvtDataBrowser to create a range theme using the values selected by the user and the AEBM or User Defined structures
Bridges	IMapPointFeatures	Created by InvtDataBrowser to create a range theme using the values selected by the user and various types of Bridges
Tunnel	IMapPointFeatures	Created by InvtDataBrowser to create a range theme using the values selected by

Classifier Object	Instance Of	Description
		the user and various types of Tunnels
Highway/RailTrack	IMapPlineFeatures	Created by InvtDataBrowser to create a range theme using the values selected by the user and various types of Highways and Railtracks
Pipeline	IMapPlineFeatures	Created by InvtDataBrowser to create a range theme using the values selected by the user and various types of Pipelines
GeographicTheme	IThMapDetails	Created by the middle tier objects to get geographic data objects



This collaboration diagram shows how the various objects interact with each other during the Scenario Definition. The diagram also indicates major methods that are used in the process.

This is a collaboration between the Data Browser, IScenarioEvent and IMapDetails objects. Following table describes the various objects used in the diagram:

Table 5-7. Scenario Definition Objects

Classifier Object	Instance Of	Description
ScenarioDefinitionWizard	Data Browser	Represents the dialog box in which the user views the scenario parameters
HistoricalEvent	IScenarioEvent	Created by ScenarioDefinitionWizard to map the Historical Epicenters
SourceEvent	IScenarioEvent	Created by ScenarioDefinitionWizard to map Faults
ArbitraryEvent	IScenarioEvent	Created by ScenarioDefinitionWizard to map Faults
ScenarioEvent	IScenarioEvent	Created by ScenarioDefinitionWizard to create the ruptured fault
GeographicData	IMapDetails	Created by the middle tier objects to get geographic data objects

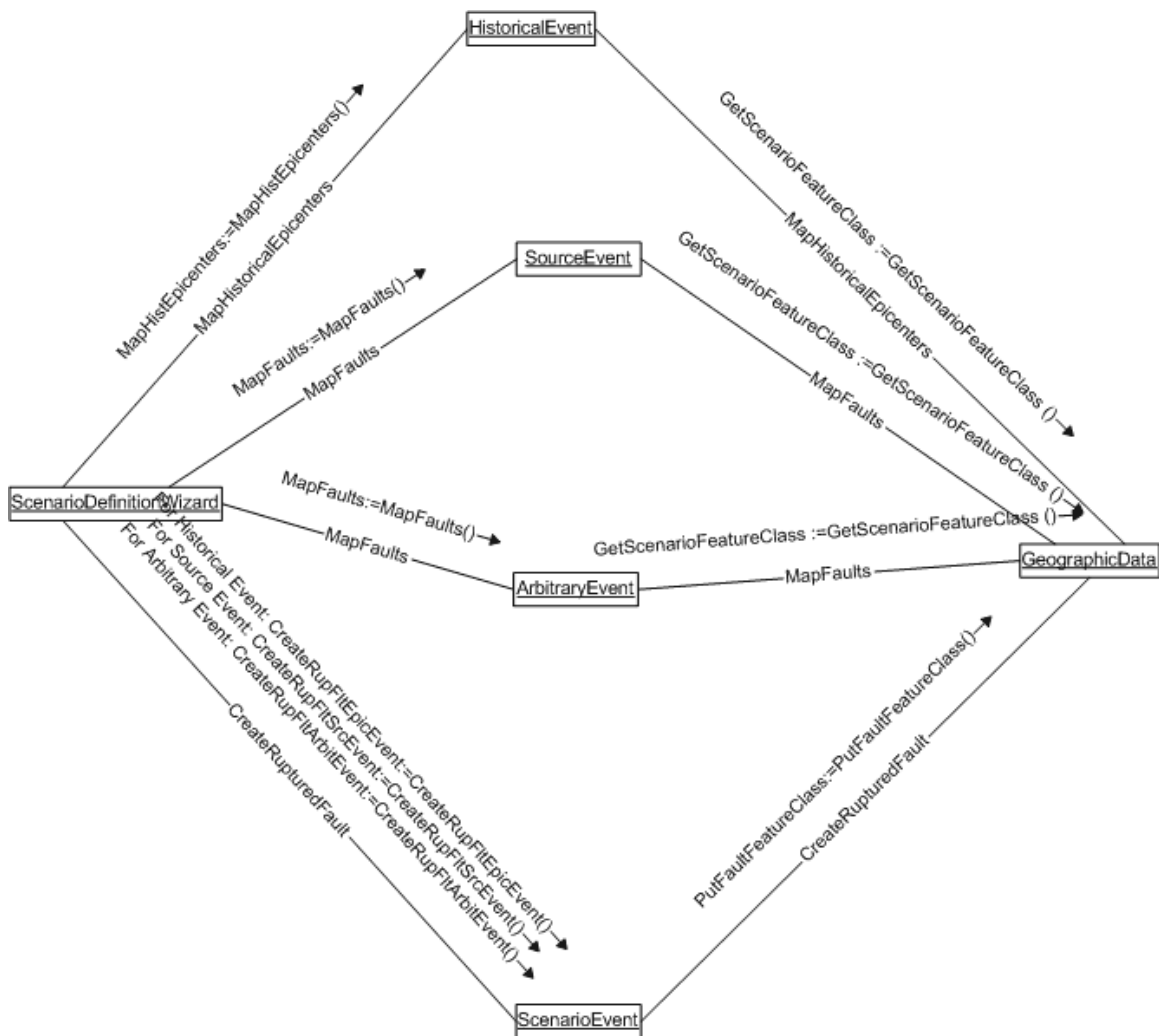


Figure 5-7. Scenario Definition Collaboration Diagram

6. Interface Description

This section consists of a set of interface specifications for each design entity or module of HAZUS and also defines the component infrastructure to support the HAZUS-MH architecture.

6.1. Database Access Layer

The overall architecture of the Data Access Layer in HAZUS-MH EQ is shown in Figure 6-1. The architecture reflects directly the design decisions made at the conceptual level of the database design, specifically reflecting the object-oriented features.

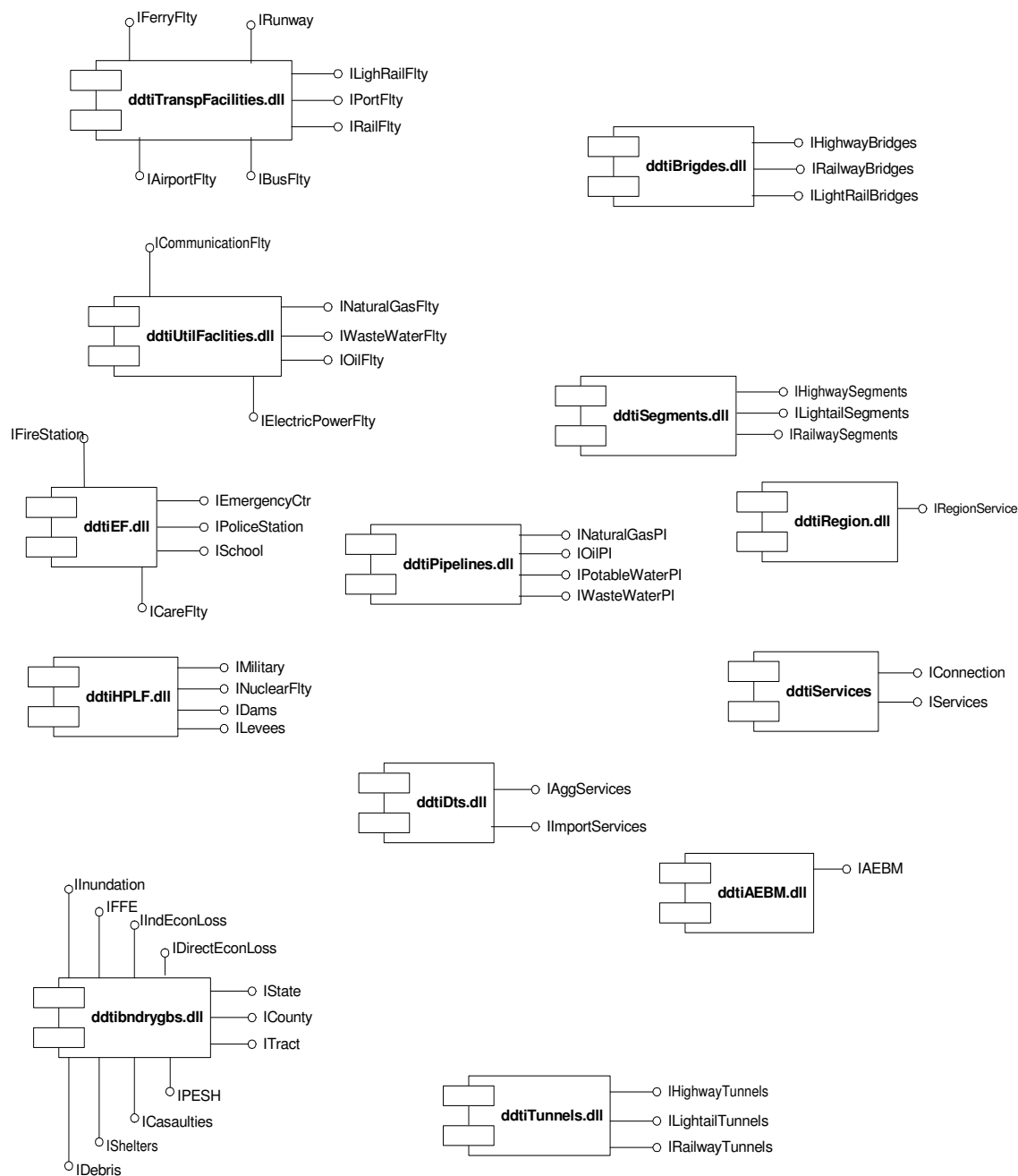


Figure 6-1. Data Access Layer (DAL) Components and Interfaces

Table 6-1 below defines the roles for each of the major DAL components as it was originally planned

Table 6-1. DAL Components Description

Component Name	Purpose
ddtiServices.dll	The core component. Serves as inner object for all other DAL components.
ddtiRegion.dll	Handles old study region interfaces and methods.
ddtiDTS.dll	Handles data transfer of system tables and packaged data from CD using DTS (Data Transformation Services)
ddtiTranspFacilities.dll	Defines all interfaces and methods for the transportation facilities. Both input (inventory) and output (results) are handled.
ddtiUtilFFacilities.dll	Defines all interfaces and methods for the utility facilities. Both input (inventory) and output (results) are handled.
ddtiBridges.dll	Defines all Interfaces and methods for all (transportation) bridges. Both input (inventory) and output (results) are handled.
ddtiSegments.dll	Defines all Interfaces and methods for all (transportation) segments (like highways, railways, etc.). Both input (inventory) and output (results) are handled.
ddtiTunnels.dll	Defines all interfaces and methods for the tunnels related to Inventory and Results
ddtiAEBM	Defines all interfaces and methods for AEBM (Advanced Engineering Building Model) tables. Both input (inventory) and output (results) are handled.
ddtiPipelines.dll	Defines all interfaces and methods for the (utility) pipelines. Both input (inventory) and output (results) are handled.
ddtiEF.dll	Defines all interfaces and methods for the essential facilities. Both input (inventory) and output (results) are handled.
ddtiHPLF.dll	Defines all interfaces and methods for the high potential loss facilities. Both input (inventory) and output (results) are handled.
ddtiBndryGBS.dll	Defines all interfaces and method for all the aggregated data as stored by county, tract, or block. That includes the square footage, building count, and exposure tables.

The following table presents a status update on the different DLLs as of the time of this edition of the SDD.

Table 6-2. Status of DLL Components and Interfaces

Original DLL Name	Existing Name	Interface	Status
ddtiServices.dll	Adtiservices.dll	IConnection	This interface does not exist
		IServices	This interface has additional 20/25 methods to handle the registry settings as well as study region information fetching process. Some of the initial planned methods do not exist nay more
ddtiRegion.dll	Ddtiregion.dll	IRegion	This interface has been written to handle all the study region related data processing. There are additionally 20-25 methods have been implanted to handle aggregation process, import export process for study region and to fetch the region information
ddtiDts.dll	ddtiDts.dll	IDtsServices	This interface handles the execution of DTS packages. Initially only 4 methods were planned as of now there are 18 methods. They execute individual DTS executions for EF, HPLF etc (all the inventory). There are methods which are generic DTS execution methods. Methods have been implemented to execute DTS packages to handle import export

			of the study region
ddtiBoundary.dll			NOT IMPLEMENTED
ddtiTranspFacilities.dll			
ddtiUtilFacilities.dll			NOT IMPLEMENTED
ddtiBridges.dll			NOT IMPLEMENTED
ddtiSegments.dll			NOT IMPLEMENTED
ddtiTunnel.dll			NOT IMPLEMENTED
ddtiAEBM.dll			NOT IMPLEMENTED
ddtiPipelines.dll			NOT IMPLEMENTED
ddtiEF.dll			NOT IMPLEMENTED
ddtiHPLF.dll			NOT IMPLEMENTED
ddtigbs.dll			NOT IMPLEMENTED

The following table shows the DAL components and interfaces that have been added to the module since it was originally designed until the time this SDD was written.

Table 6-3. New DAL Components and Interfaces

Component Name	Purpose
ddtiAnalysisParameters.dll	This component exposing interrelates to handle analysis parameters fetching and setting in the data base this includes the parameters for AEBM, PESH, SCENARIO and Buildings
ddtiDAL.dll	This is the code DAL component providing all the generic data methods handling record sets in the SQL and access data bases including fetching of the record set, saving of the record set , searching of the record set for study region data base as well as the system data base (SYHAZUS)
ddtiDataManager.dll	This component has been written to combine all initial planned following components and expose their interfaces. ddtiTranspFacilities.dll

Component Name	Purpose
	ddtiUtilFFacilities.dll ddtiBridges.dll ddtiSegments.dll ddtiTunnels.dll ddtiAEBM ddtiPipelines.dll ddtiEF.dll ddtiHPLF.dll All the interfaces above provide basic functionality functions like Synchronizing tables between MDB data base and SQL data base of study region for any kind of editing Opening the inventory record set Saving the inventory and result record set Getting the analysis values record set Exporting the record set to table or text file. Additional interface for handling all the occupancy mapping has been implemented names IOccuMap
ddtiSystemParameters.dll	This comment provides interface and methods to handle the system parameters like Thematic Mapping, data maps , Hazard maps for example methods like getLabel, getClassification provide thematic mapping information

6.2. Application Logic Layer

6.2.1. Aggregation Layer

The aggregation map layer will be composed in “adtiAggregation.dll” component. The aggregation map layer (AML) is implemented using COM technology. The aggregation component is shown in Figure 6-2 and the methods for the aggregation interface are described in 6.2.1.1.



Figure 6-2. Aggregation Component

6.2.1.1. Aggregation Components and Methods

The following table shows the aggregation component and method descriptions.

Table 6-4. Components and Methods

Method Name	Description
AggregateBoundary	Method for Aggregation of census tracts, study region boundary and creation of the data maps.
AggregateBridges	Method for aggregation of the bridges.
AggregateTransportFacilities	Method for aggregation of transport facilities.
AggregateUtilityFacilities	Method for aggregation of transport facilities.
AggregateTunnels	Method for aggregation of the tunnels.
AggregateSegments	Method for aggregation of the segments.
AggregatePipeLines	Method for aggregation of the pipe lines
AggregateEssentialFacilities	Method for aggregation of essential facilities.
AggregateHPLF	Method for aggregation of high potential loss facilities.
AggregateAEBM	Method for aggregation of user defined inventory AEBM.

6.2.2. The Analysis Engine

The Analysis Engine Layer will be composed of a set of components as defined in Table 6-4 and will have the interfaces shown in Figure 6-3.

Table 6-5. Analysis Engine Components

Component Name	Purpose
adtiAnalysis.dll	This component is the application level component for running the complete analysis for HAZUS-MH. This in turn calls other analysis specific components.
adtiBridgeTunnel.dll	Handles analysis for the bridges and tunnels includes demand, damage, and direct loss.
adtiSegmentPipe.dll	Handles analysis for the segments and the pipe lines includes demand, damage, and direct loss.
adtiFacility.dll	Handles analysis for essential facilities, HPLF, transport facilities, AEBM and utility facilities includes demand, damage, and direct loss.
adtiAttenuationFunction.dll	Implements the attenuation functions and provides interfaces for the same includes both west and east US.
adtiBuildings.dll	Implements the core modal for demand, damage, direct loss, and FFE, debris, shelter, casualties, indirect economic loss for buildings and provides the interface for the same.
adtiInventory.dll	Implements the core modal for demand, damage, direct loss, system performance, functionality computation for the inventory and provides the interface for the same.

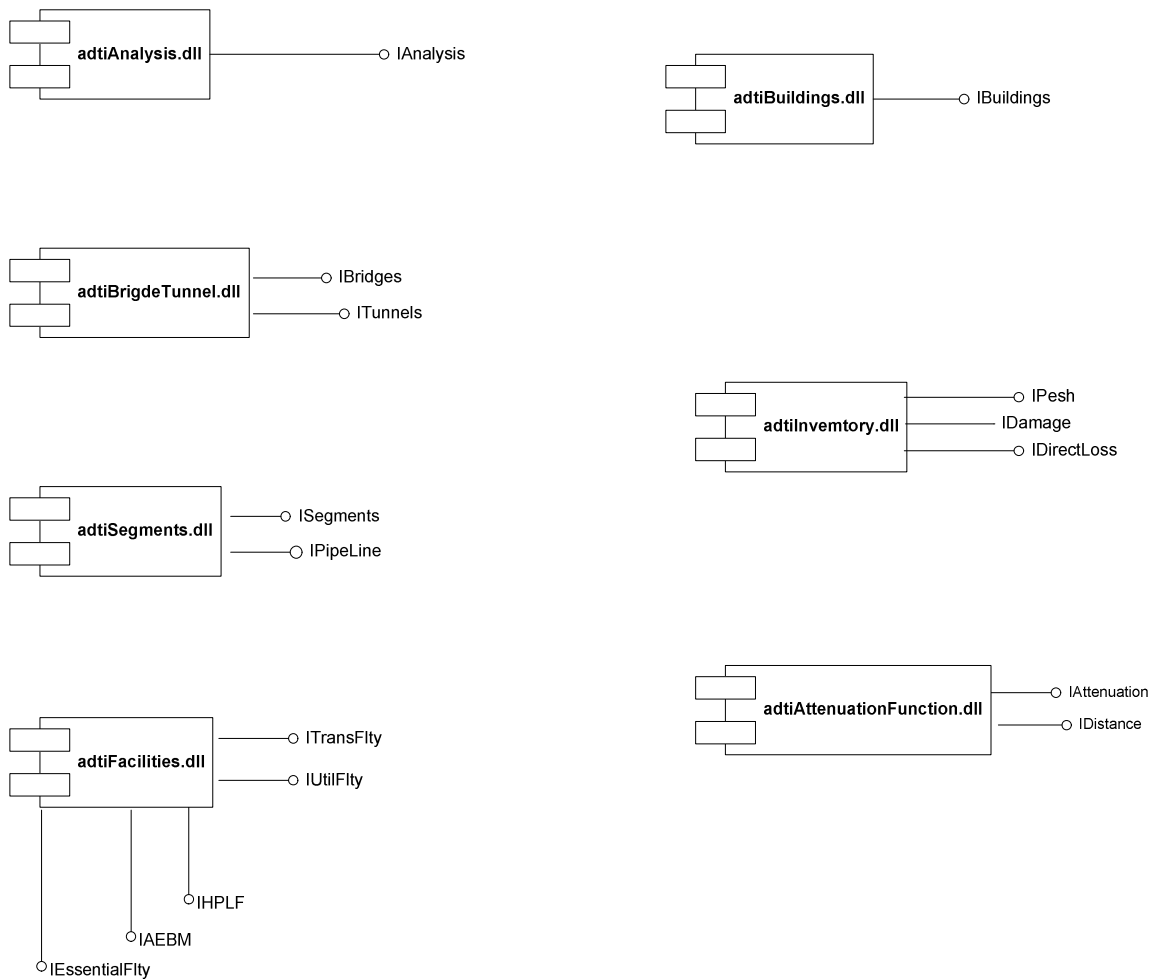


Figure 6-3 Analysis Engine Components and Interfaces

The following sections will provide s description of the interfaces per component.

6.2.2.1. adtiAnalysis.dll Component

Interface	Methods	Methods Purpose
I IAnalysis	UPdateDependencies()	Update the dependencies for analysis module.
	RunAnalysis()	Run the analysis. This method checks for the various analysis selected by the user and then run the specific analysis by inventory.

6.2.2.2. adtiBridgeTunnel.dll Component

Interface	Methods	Methods Purpose
IBridges	RunPesh()	Run the PESH computations for the all the bridges.
	RunDamage()	Run the damage computations for all the bridges.
	RunFunctionality()	Run the functionality computations for all the bridges.
	RunDirectLoss()	Run the direct loss computations for all the bridges.
ITunnels	RunPesh()	Run the PESH computation for all the tunnels..
	RunDamage()	Run the damage computations for all the tunnels.
	RunFunctionality()	Run the functionality computations for all the tunnels..
	RunDirectLoss()	Run the direct loss computations for all the tunnels..

6.2.2.3. adtiSegmentPipe.dll Component

Interface	Methods	Methods Purpose
ISegments	RunPesh()	Run the PESH computation for the highway, rail road, and light rail segments.
	RunDamage()	Run the damage computations for the highway, rail road, and light rail segments.
	RunFunctionality()	Run the functionality computations for the highway, rail road, and light rail segments.
	RunDirectLoss()	Run the direct loss computations for the highway, rail road, and light rail segments.

Interface	Methods	Methods Purpose
IPipeline	RunPesh()	Run the PESH computation for the pipe line like potable water, waste water etc.
	RunDamage()	Run the damage computations for the pipe line like potable water, waste water etc.
	RunFunctionality()	Run the functionality computations for the pipe line like potable water, waste water etc.
	RunDirectLoss()	Run the direct loss computations for the pipe line like potable water, waste water etc.

6.2.2.4. adtiFacility.dll Component

Interface	Methods	Methods Purpose
ITransFlty	RunPesh()	Run PESH, damage, functionality, and direct loss for all the transportation facilities.
	RunDamage()	
	RunFunctionality()	
	RunDirectLoss()	
IUtilFlty	RunPesh()	Run PESH, damage, functionality, and direct loss for all the utility facilities.
	RunDamage()	
	RunFunctionality()	
	RunDirectLoss()	
IEssentialFlty	RunPesh()	Run PESH, damage, functionality for all the essential facilities.
	RunDamage()	
	RunFunctionality()	
IAEBM	RunPesh()	Run the PESH computation for the AEBM.
	RunDamage()	Run the damage computations for the AEBM.
	RunFunctionality()	Run the functionality computations for the AEBM.

Interface	Methods	Methods Purpose
	RunDirectLoss()	Run the direct loss computations for the AEBM.
	RunCasualties()	Run the casualties for AEBM.
IHPLF	RunPesh()	Run the PESH, damage, functionality, and direct loss for the military installations.
	RunDamage()	
	RunFunctionality()	
	RunDirectLoss()	

6.2.2.5. adtiBuilstdings.dll Component

Interface	Methods	Methods Purpose
IBuildings	CalculateDemandGM()	Calculate ground motion for buildings.
	CalculateDemandGF()	Calculate ground failure for buildings.
	CalculateDamageState()	Calculate damage state for buildings.
	ReturnProbabilities()	Return probabilities for damage state for the EF and military installations.
	CalculateDirectLoss()	Calculate direct loss for buildings.
	CalculateCasualty()	Calculate different casualties.
	CalculateShelter()	Calculate shelter.
	CalculateFFE()	Calculate FFE.
	CalculateDebris()	Calculate Debris.
	CalculateInundations()	Compute the inundations.
	CalculateInDecLoss	Calculate indirect economic loss.

6.2.2.6. adtiInventory.dll Component

Interface	Methods	Methods Purpose
IPesh	CalcDemandGMPoint()	Calculates the demand at ground motion level for the point type inventory.
	CalcDemandGFPoint()	Calculates the demand at ground failure level for the point type inventory

Interface	Methods	Methods Purpose
	CalcDemandGMLine()	Calculates the demand at ground motion level for the line type inventory
	CalcDemandGFLine()	Calculates the demand at ground failure level for the line type inventory
IDamage	CalcDamageStatePoint()	Calculate the damage state for the point type inventory
	CalcDamageStateSegments()	Calculate the damage state for the segments type inventory
	CalcDamageStatePipeline()	Calculate the damage state for the pipeline type inventory
	CalcDamageStatePoint()	Calculate the damage state for the point type inventory
	CalcFunctionalityPoint()	Calculate functionality for the point type inventory.
	CalcFunctionalitySegments()	Calculate functionality for the segment type inventory.
	CalcSystemPerformance	Calculate system performance for the pipe line inventory.
IDirectLoss	CalcDirectLossPoint()	Calculate the Direct Loss for the point type inventory
	CalcDirectLossSegments()	Calculate the Direct Loss for the segments type inventory
	CalcDirectLossPipeline()	Calculate the Direct Loss for the pipeline type inventory

6.2.2.7. adtiAttenuationFunction.dll Component

Interface	Method	Method Purpose
IAttenuation	WP97()	Attenuation function for West Project 97

Interface	Method	Method Purpose
	WP97PN()	Attenuation function for West Project 97PN
	WBJF()	WBJF Attenuation function
	WSAD()	WSAD Attenuation function
	WYNG()	West YNG Attenuation function
	EP97()	East Project 97 Attenuation function.
	EFRNKL()	Franklin Attenuation function
	ETORO()	Toro Attenuation function
	LLNL()	New East Coast Attenuation Function
	HAFMT()	Munson and Thurber Attenuation Function
	HAFP97()	Project 97, Hawaii Attenuation Function
IDistance	CalcPonitToPoint()	This method calculates distance between two points.
	CalcPointToLine()	This method calculates distances between a point and a poly line.

6.3. Presentation Services Layer

6.3.1. Occupancy Mapping Dialog Interface

The Occupancy Mapping Dialog is one of the components in the presentation layer for the Earthquake Model. It will talk to the middle layer, which in turn will be talking to the data layer for all data related operations. At the interface level the organization of the Occupancy Mapping Dialog will be as depicted in the following diagram:

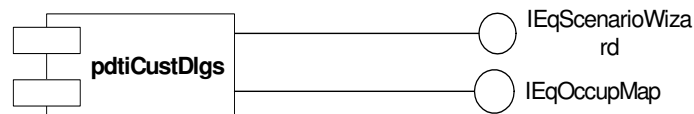


Figure 6-4 Occupancy Mapping Dialog Interface

The pdtiCustDlgs component will implement an interface IeqOccMapDlg that will invoke the main dialog of occupancy mapping.

The Occupancy Mapping Dialog will use the following methods of middle layer component to interact with it. These methods would be created in the IEQOccupMap interface in the adtilventoryMgr component.

Table 6-6. Occupancy Mapping Dialog Components

IEqOccupMap	<p>GetBldgSchemesDetails ([out] _Recordset **schemesDetails)</p> <p>This method will give details of all the Mapping schemes available and the number of tracts assigned to each mapping scheme and will return two recordsets for these. This method will get all the mapping schemes from hzGenBldgSchemes table and the number of tracts assigned per mapping scheme will be taken from hzTract table.</p>
	<p>AddNewGenBldgScheme ([in] BSTR bldgSchemesId, [in] BSTR schemeName, [in] BSTR description, [out] [out, retval] BSTR *newBldgSchemesId)</p> <p>This method will create a new bldgSchemesId via stored procedure dtip_GenerateNewSchemeId and insert a new record for that created bldgSchemesId, input parameter scheme name and description in table hzGenBldgSchemes. It will also replicate all the records in tables' hzGenBldgScheme and eqrInSchemes corresponding to the given bldgSchemesId for the new bldgSchemesId. This method will return the bldgSchemesId for the inserted record.</p>
	<p>GetGenBldgSchemeDetails ([in] BSTR bldgSchemesId, [out] _Recordset **genBldgSchemeDetails)</p> <p>This method creates a recordset for the given bldgSchemesId</p>

	from table hzGenBldgScheme.
	<p>UpdateRecordset ([in] _Recordset *recordset)</p> <p>This method will update the recordset supplied as parameter to the database. This method will call Save method of IDataAccess interface of ddtiDAL component.</p>
	<p>AddNewSpcBldgSchemeAndType ([in] BSTR genBldgSchemeld, [in] BSTR bldgTypeSource, [out] _Recordset **bldgTypeDetails)</p> <p>This method will insert an empty record in eqSpcBldgSchemes. All fields in that table will be populated at this stage except for 'SchemeName' & 'Description'. eqSpcBldgSchemsId will be generated via stored procedure dtip_GenerateNewSchemeld. 'MappingType' will be 'U', and 'BldgTypeSource' will be bldgTypeSource passed as input parameter. Depending on the bldgTypeSource records are replicated in, the corresponding eqXBldgTypeMp table for old eqSpcBldgSchemesId (We will get old eqSpcBldgSchemesId from eqSpcBldgSchemes table for the given genBldgSchemeld, bldgTypeSource through table eqrInSchemes) assigning them the new eqSpcBldgSchemsId generated via stored procedure dtip_GenerateNewSchemeld.</p>
	<p>UpdateSpcBldgSchemeAndType ([in] BSTR spcBldgSchemesId, [in] BSTR name, [in] BSTR description, [in] _Recordset *bldgTypeDetails)</p> <p>This method will update the name and description of a specific scheme for the corresponding spcBldgSchemesId passed as input parameter in table eqSpcBldgSchemes. It will also update the bldgTypeDetails recordset passed as parameter to the database.</p>

	<p>UpdateRlnSchemes ([in] BSTR genBldgSchemeld, [in] BSTR oldSpcBldgSchemesId, [in] BSTR newSpcBldgSchemesId)</p> <p>This method will update the oldSpcBldgSchemesId, with newSpcBldgSchemesId for the corresponding record in eqlnSchemes table for the given genBldgSchemeld and oldSpcBldgSchemesId.</p>
--	--

6.4. Foundation Type Dialog Interface

A foundation type mapping scheme gives percentages of buildings that are on deep foundations for each bldg type, therefore the scheme is 36 rows (since there are 36 building types).

The dialog above makes use of the table **eqFndTypeVals**.

6.5. The Data Browser

Figure 6-5 shows the organization of the data browser at the interface level.

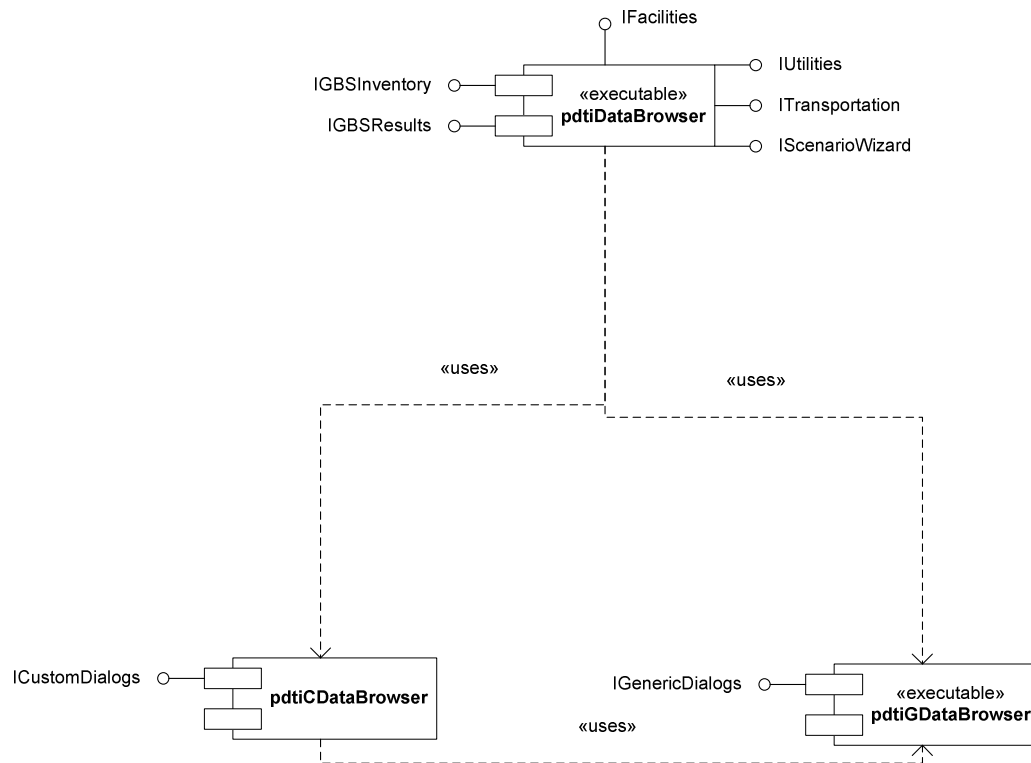


Figure 6-5. Data Browser Interface Diagram

The Data Browser is a set of three components. The first component, pdtiDataBrowser is the controlling application for Data Browser and interacts with the menu. It implements six interfaces. The menu handlers make calls to various methods in the six interfaces provided by the component. The following table describes the interface methods of the Data Browser.

Table 6-7. Data Browser Main Interfaces

S.No	Interface Name	Description
1	IGBSInventory	Provides methods for the display of all the GBS Inventory data like Square Footage, Building Count,

		Dollar Exposure, Mapping schemes, etc.
2	IGBSResults	Provides methods for the display of all the GBS analysis results like Ground Motion and Ground Failure, Damaged Building Count, Economic Loss, Casualties, Shelter, etc.
3	IFacilities	Provides methods for the display of EF, HPLF, User Defined Structures and AEBM inventory data and analysis results.
4	ITransportation	Provides methods for the display of all Transportation systems inventory data and analysis results.
5	IUtilities	Provides methods for the display of all Utility systems inventory data and analysis results.
6	IScenarioWizard	Implements the wizard for the scenario definition and selection.

The second component `pdtiGDataBrowser` provides the basic generic implementation for dialogs as per the Data Browser specifications. This component is used by the `pdtiDataBrowser` and `pdtiCdataBrowser` components for display of all the dialogs that fall in this generic category. This component implements only one interface, ***IGenericDialogs***, which provides a set of attributes and methods to define the look and feel of the dialog.

The third component `pdtiCDataBrowser` provides the implementation for dialogs as per the Data Browser specifications that do not fall in the generic implementation category. This component is used by the `pdtiDataBrowser` component for display of all the dialogs that require different treatment. This component implements only one interface ***IcustomDialogs***.

6.6. The Mapping Engine

The following diagrams describe the relations among these components and the relation among the interfaces within these components

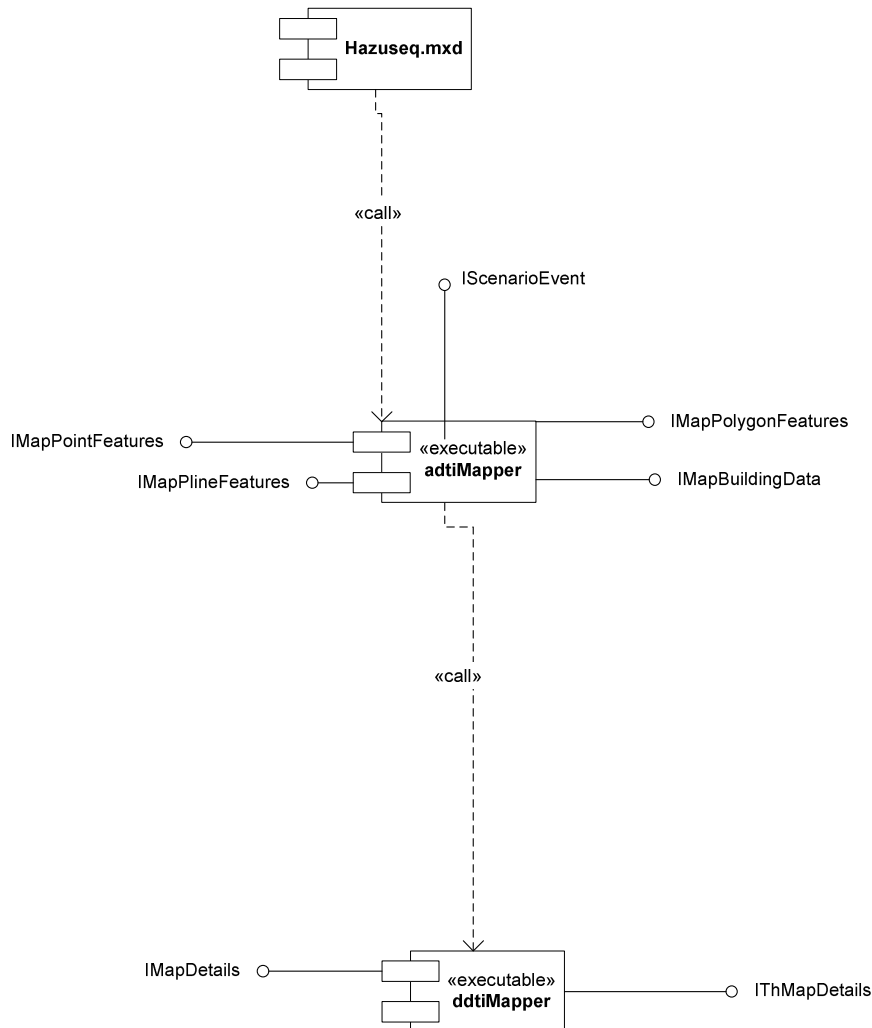


Figure 6-6. Mapping Engine Component Interaction Diagram

The HazusEq.mxd uses the component adtiMapper for various geographic operations. adtiMapper in turn uses the ddtiMapper component for getting hold of the relevant geographic data required to complete the operation requested by HazusEq.mxd. The interaction between these components is through the methods defined in the interfaces implemented by these components.

The Class Static Structure diagram below explains the relationships between the various interfaces defined in the components adtiMapper and ddtiMapper. The general relation is:

An interface in adtiMapper uses an interface in ddtiMapper to access the geographic data from the Geodatabases.

6.6.1. The Mapping Engine Components

The interfaces defined in adtiMapper component are:

1. IMapPointFeatures
2. IMapPlineFeatures
3. ImapPolygonFeatures
4. ImapBuildingData
5. IscenarioEvent
6. ImapSelDlg

These interfaces implement the HAZUS specific logic for grouping the geographic data and thus controlling the display settings, editing, creating and manipulation (like merging, clipping, splitting, etc.) for the geographic data.

The interfaces implemented in ddtiMapper component are:

1. ImapDetails
2. IthMapDetails

All the geographic data-access operations are encapsulated in these two interfaces. The methods in the adtiMapper component interfaces provide the group and display settings information to the methods in the ddtiMapper interfaces provide the group and display settings information to the methods in the ddtiMapper interfaces. The ddtiMapper interface methods locate the proper geographic dataset based on the group information and apply the display setting to it and then pass these fomatted geographic entitites back.

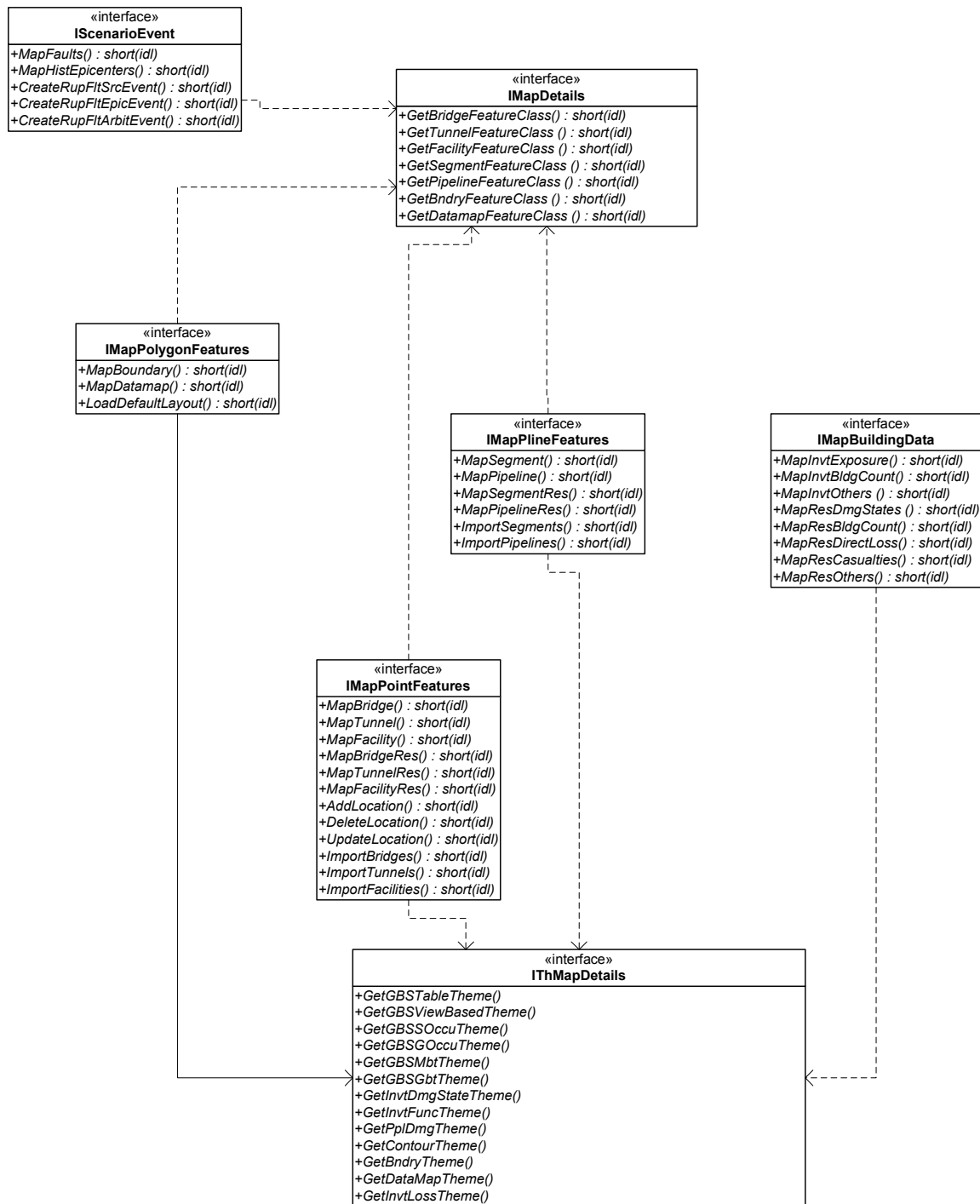


Figure 6-7. The Mapping Engine Components

As described earlier the mapping engine application layer is implemented through the component `adiMapper.dll` and the data access layer is implemented through the

component ddtiMapper.dll. The following are the interfaces exposed by each component and their methods.

6.6.1.1. adtiMapper.dll Component

As mentioned above, this compoent implements six interfaces. The purpose of each interface is described below:

Table 6-8.adtiMapper.dll Component Interface Descriptions

Interface Name	Description
IMapPointFeatures	This interface provides all operations (Display, Editing, Themes, etc.) for point type geographic data. The various methods perform the operations based on the grouping of the data.
IMapPlineFeatures	This interface provides all operations (Display, Import, Themes, etc.) for poly-line type geographic data. The various methods perform the operations based on the grouping of the data.
IMapPolygonFeatures	This interface provides all operations (Display, Editing, Themes, etc.) for polygon type geographic data. The various methods perform the operations based on the grouping of the data. In addition to this it will also provide support for map layouts for printing.
IMapBuildingData	This interface provides operations for displaying General Building data and analysis results themes on the Census Tract boundary map. The various methods perform the operations based on the grouping of the data.
IScenarioEvent	This interface will help in scenario event definition process. It provides methods to display the Historical Epicenter and Fault maps. The most important task of this interface is the creation of the ruptured fault based on the Scenario Event Type.
IMapSelDlg	This interface provides with dialog boxes that allow the

	user to select elements from the map using various geographic tools available on the dialog box.
--	--

The ImapselDlg has been described below for the aggregation level selection.

The User Interface for the Selection Dialogs for Aggregaion will have a view to hold the map layer for the state, county, tract or block. It will have all the basic GIS tools.

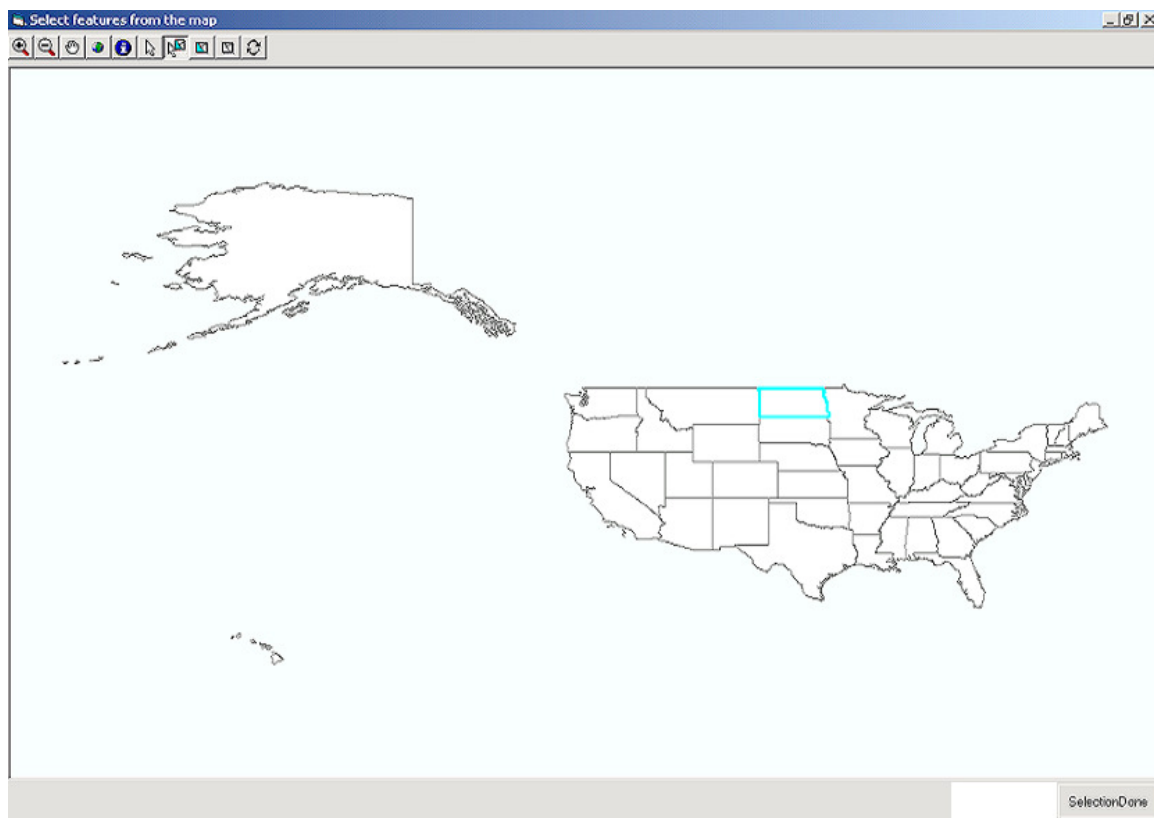


Figure 6-8. The Aggregation Level Selection GUI

Table 6-9 describes the return status.

Table 6-9. Aggregation Selection Return Codes

Button / Dialog	Selection Done Successfully	Selection Cancelled	Error in Selection
Selection Done	1	2	0






Buttons






- **Selection Done:** This button will be enabled only when there is a valid selection. At Click on this button, the study region selection table will be updated and control will go back to the Shell. Wizard for the active selection mode will be displayed with the new selection set.
- **Cancel:** This button is provided to cancel the selection process and discard all the changes made.
- **Tools:** Selection dialog will have the tools depicted in Figure 6-8 to perform the GIS actions for selection. All these tools will be custom tools implemented in the Map Layer component



Figure 6-9. Tools Menu

Table 6-10. Tools Descriptions

Tool	Tool Tip	Description
	Zoom In	This tool can be used for zooming inside the layer to have a larger image of any area on the layer.
	Zoom Out	This tool can be used for zooming outside the layer to have a larger view of the layer and to cover more area on the layer.
	Pan	The pan tool is used to move the layers on the map in any direction up, down, left right to view the area of interest.
	Full Extent	This tool is used to view the complete view of the visible layers. It will cover the maximum extent of the layer.
	Identify	The identify tool is used to get information at any point on the layer. It will give all the attribute information of the layer beneath the point of click.

	Select	This is a regular select.
	Select Features	This tool is used to select more features from the active layers on the view.
	Select All	This tool is used to select all the features of the layer.
	De Select All	This tool is used to un select all the selected features of the User
	Switch Selection	For reversing/switching the selection this tool is used. What it does is, it makes the selected features unselected and, unselected features selected.

The IMapselDlg interface will have the following functions / events to handle the selection.

Table 6-11. IMapselDlg Interfaces Descriptions

Name	Description
AggSelection_MouseMove	Even to trap the Mouse move, if there is a valid tool active then take the appropriate action for ex if the tool is pan, then result into then make the pan action.
AggSelection_MouseUp	Even to trap the Mouse Up, if there is a valid tool active then take the appropriate action for ex if the tool is pan, then result into then stop of pan action.
AggSelection_MouseDown	Even to trap the Mouse Up, if there is a valid tool active then take the appropriate action for ex if the tool is pan, then result into then start the pan action.
AggSelectionDone_Click	At Click on this button, The Study Region Selection table will be updated, all the maps will be closed and cleaned up and control will go back to the

	Shell. Wizard for the Active Selection Mode will be displayed with new selection set. .
AggSelectionToolBar_Click	To handle the click event on the tool bar on the selection dialog, this will change the mouse cursor to the tool clicked and also set the clicked tool to active in the tool bar.
Cross Button	This will be in turn calling the AggSelectionToolBar_Click.
AggSelection_Resize	At Click on this button, all the selections will be cancelled. All the maps will be closed and cleaned up and control will go back to the Shell. Wizard for the Active Selection Mode will be displayed with new selection set.
AggSelecionCancel()	At Click on this button, The Study Region Selection table will be updated and control will go back to the Shell. Wizard for the Active Selection Mode will be displayed with new selection set.
AggSelOpenSelection()	This method will check what is the state of selection and then read form the selection table for any selections previously. This method will in turn call all other methods for the selection process.
AggSelOpenMap()	This method will open the maps and create the feature class for them
AggSelOpenLayer()	This method will be used for opening any layer from the feature class.
AggSelCloseMaps()	This method will close all the maps.
AggSelSelectionSet()	This method will handle the selection process, making the selection and then reflection the selection change on the layer.

The following is a list of the methods implemented by this interface and their short description.

Table 6-12. IMapSelDlg Interface Methods and Their Descriptions

Interface	Methods	Description
IMapPointFeatures	MapBridge(int iBridgeType)	Calls the data-layer method to get the Bridge feature class and then displays the feature class on the map document
	AggBridge(int iBridgeType)	Copies all the bridges that fall in the study region from the default data into the bridge geo-database in the study region
	MapTunnel(int iTunlType)	Calls the data-layer method to get the Tunnel feature class and then displays the feature class on the map document
	AggTunnel(int iTunlType)	Copies all the tunnels that fall in the study region from the default data into the tunnel geo-database in the study region
	MapFacility(int iFclyType)	Calls the data-layer method to get the Facility (EF, HPLF, AEBM, UserDEF, LL) feature class and then displays the feature class on the map document
	AggFacility(int iFclyType)	Copies all the facilities that fall in the study region from the default data into the facility geo-database in the study region

Interface	Methods	Description
	MapBridgeRes(int iBridgeType, int iResType)	Calls the data-layer method to get the Bridge results theme class and then displays the theme on the map document
	MapTunnelRes(int iTunlType, int iResType)	Calls the data-layer method to get the Tunnel results theme class and then displays the theme on the map document
	MapFacilityRes(int iFclyType, int iResType)	Calls the data-layer method to get the Facility results theme class and then displays the theme on the map document
	AddLocation(IFeatureClass pFeatClass, double dLong, double dLat)	Adds the information about the new location in the corresponding attribute table in SQL Server database.
	DeleteLocation(IFeatureClass pFeatClass)	Deletes the record associated with the location in the corresponding attribute table in SQL Server database.
	UpdateLocation(IFeatureClass pFeatClass, double dLong, double dLat)	Updates the information about the new location in the corresponding attribute table in SQL Server database.
	UpdateHazardFactors()	Updates the Soil, Liquefaction, Landslide and WaterDepth values for every location in every inventory including the n locations for every segment/pipeline.

Interface	Methods	Description
	ImportBridges(int iBridgeType, bstr sSource)	Imports the geographic data about the bridges from the geodatabase provided by the user into the current study region's corresponding geodatabase.
	ImportTunnels(int iTunlType, bstr sSource)	Imports the geographic data about the tunnels from the geodatabase provided by the user into the current study region's corresponding geodatabase.
	ImportFacilities(int iFcltyType, bstr sSource)	Imports the geographic data about the facilities from the geodatabase provided by the user into the current study region's corresponding geodatabase.
IMapPlineFeatures	MapSegment(int iSegType)	Calls the data-layer method to get the Segment (Highway/Railway/Lightrail) feature class and then displays the feature class on the map document
	AggSegment(int iSegType)	Copies all the segments that fall in the study region from the default data into the segment geo-database in the study region and clips them at the study region boundary

Interface	Methods	Description
	MapPipeline(int iPplType)	Calls the data-layer method to get the pipeline feature class and then displays the feature class on the map document
	AggPipeline(int iPplType)	Copies all the pipelines that fall in the study region from the default data into the pipeline geo-database in the study region and clips them at the study region boundary and creates the corresponding Hazard Factor Tables (HFT)
	MapSegmentRes(int iSegType, int iResType)	Calls the data-layer method to get the Segment results theme class and then displays the theme on the map document
	MapPipelineRes(int iPplType, int iResType)	Calls the data-layer method to get the Pipeline results theme class and then displays the theme on the map document
	ImportSegments(int iSegType, bstr sSource)	Imports the geographic data about the segments from the geodatabase provided by the user into the current study region's corresponding geodatabase.
	ImportPipelines(int iPplType, bstr sSource)	Imports the geographic data about the pipelines from the geodatabase provided by the user into the current study region's corresponding geodatabase.

Interface	Methods	Description
IMapPolygonFeatures	MapBoundary(int iBndryType)	Calls the data-layer method to get the Boundary (Study Region/ Census tract/ Block) feature class and then displays the feature class on the map document
	AggBoundary(int iBndryType)	Copies all the boundaries that constitute the study region from default data into Boundary geodatabase in the study region.
	CombineBoundaries(int iBndryType, int iPersist)	Combines all the boundaries to get one boundary object and save it or not based on the iPersist parameter
	MapDatamap(int iDataMapType)	Calls the data-layer method to get the Datamap (Soil/ Liquefaction/ Landslide/ WaterDepth) feature class and then displays the feature class on the map document
	CreateDefDatamap(int iDataMapType)	Creates the default datamap based on the default value associated to the datamap and the study region census tract boundaries.
	LoadDefaultLayout()	Loads the default print layout for the maps.
IMapBuildingData	MapInvtExposure(int iExpType)	Calls the data-layer method to get the Exposure (by MBT/ GBT/ SOCCU/ GOCCU) theme class and then displays the theme on the map document

Interface	Methods	Description
	MapInvtBldgCount(int iCountType)	Calls the data-layer method to get the Building Count (by MBT/ GBT/ SOCCU/ GOCCU) theme class and then displays the theme on the map document
	MapInvtOthers (int iTblType)	Calls the data-layer method to get the Square Footage/ Demographics theme class and then displays the theme on the map document
	MapResDmgStates (int iDmgType)	Calls the data-layer method to get the GBS Damage (by MBT/ GBT/ SOCCU/ GOCCU) theme class and then displays the theme on the map document
	MapResBldgCount(int iCountType)	Calls the data-layer method to get the Damaged Building Count (by MBT/ GBT/ SOCCU/ GOCCU) theme class and then displays the theme on the map document
	MapResDirectLoss(int iLossType)	Calls the data-layer method to get the Direct Economic Loss (by MBT/ GBT/ SOCCU/ GOCCU) theme class and then displays the theme on the map document
	MapResCasualties(int iCasType)	Calls the data-layer method to get the Casualty (by MBT/ GBT/ GOCCU) theme class and then displays the theme on the map document

Interface	Methods	Description
	MapResOthers(int iTblType)	Calls the data-layer method to get the Ground Motion/ Ground Failure/ Inundation/ Fire Following/ Debris/ Shelter theme class and then displays the theme on the map document
IScenarioEvent	MapFaults()	Calls the data-layer method to get the Fault feature class and then displays the feature class on the map document
	MapHistEpicenters()	Calls the data-layer method to get the Historical Epicenter feature class and then displays the feature class on the map document
	CreateRupFltSrcEvent(int iSelFltID)	Creates the ruptured fault for a source event scenario.
	CreateRupFltEpicEvent(int iSelEpicID)	Creates the ruptured fault for the historical epicenter event scenario.
	CreateRupFltArbitEvent(double dLong,double dLat)	Creates the ruptured fault for the arbitrary event scenario.
IMapSelDlg	SelStateDlg()	Creates the Dialog for selecting States in Study Region Selection Wizard.
	SelCountyDlg()	Creates the Dialog for selecting the Counties within the selected states in Study Region Selection Wizard.

Interface	Methods	Description
	SelTractDlg()	Creates the Dialog for selecting the Census Tracts within the selected counties in Study Region Selection Wizard.
	SelBlockDlg()	Creates the Dialog for selecting the Blocks within the selected census tracts in Study Region Selection Wizard.
	SelEpicenterDlg(int iEpicenterType)	Creates the Dialog for selecting a epicenter in Scenario Selection Wizard
	SelFaultDlg()	Creates the Dialog for selecting a fault in Scenario Selection Wizard

6.6.1.2. ddtiMapper.dll

This component implements two interfaces as described below

Table 6-13. ddtiMapper.dll Components

S.No	Interface Name	Description
1	IMapDetails	This interface provides methods that access all types of geographic data for various Geo-Databases used by HAZUS-MH. The major task is to identify the underlying Geographic data based on the Data Group information and then create the map objects out of it.
2	IThMapDetails	This interface provides methods that relate the spatial data with non spatial data and then creates themes based on the values in the non-spatial data.

Table 6-14 lists the methods implemented by the interfaces in Table 6-13 and provides a short description of the functionality that they provide.

Table 6-14. ddtiMapper.dll Interface Methods and Their Descriptions

Interface	Methods	Description
IMapDetails	GetBridgeFeatureClass(int iBridgeType, int iPathType, IFeatureClass pBridge)	Based on the bridge type and data path type, accesses the proper geodatabase (from study region or data CD) and creates a feature class using the corresponding bridge table.
	GetTunnelFeatureClass (int iTunlType, int iPathType, IFeatureClass pTunnel)	Based on the tunnel type and data path type, accesses the proper geodatabase (from study region or data CD) and creates a feature class using the corresponding bridge table.
	GetFacilityFeatureClass (int iFclyType, int iPathType, IFeatureClass pFacility)	Based on the facility type and data path type, accesses the proper geodatabase (from study region or data CD) and creates a feature class using the corresponding bridge table.
	GetSegmentFeatureClass (int iFclyType, int iPathType, IFeatureClass pSegment)	Based on the segment type and data path type, accesses the proper geodatabase (from study region or data CD) and creates a feature class using the corresponding bridge table.
	GetPipelineFeatureClass (int iFclyType, int iPathType, IFeatureClass pPipeline)	Based on the pipeline type and data path type, accesses the proper geodatabase (from study region or data CD) and creates a feature class using the corresponding bridge table.

Interface	Methods	Description
	GetBndryFeatureClass (int iFcltyType, int iPathType, IFeatureClass pBoundary)	Based on the boundary type and data path type, accesses the proper geodatabase (from study region or data CD) and creates a feature class using the corresponding bridge table.
	GetScenarioFeatureClass (int iScenarioType, IFeatureClass pBoundary)	Based on the scenario type, accesses the proper geodatabase and creates a feature class using the either the Historical Epicenter or Fault table
	PutFaultFeatureClass(int iScenarioType, IFeatureClass pBoundary)	Take the FeatureClass and insert into the Scenario table
	CopyFeatures(IFeatureClass FromTable, IFeatureClass ToTable)	Copies the features from one feature set to another feature set.
	GetDatamapFeatureClass (int iMapType, IFeatureClass pDatamap)	Based on the datamap type, accesses the proper geodatabase and creates a feature class using the corresponding bridge table.
IThMapDetails	GetGBSTableTheme(int iTblType, IRelationshipClass pRelClass, IClassBreaksRenderer pRenderer)	Creates a relation between the GBS table in SQL server to the Census tract boundary and based on a specific type of attribute creates a range theme.

Interface	Methods	Description
	GetGBSViewBasedTheme(int iTblType, IRelationshipClass pRelClass, IClassBreaksRenderer pRenderer)	Creates a relation between a specific view of a GBS table in SQL server to the Census tract boundary and based on a specific type of attribute creates a range theme.
	GetGBSSOccuTheme(int iTblType, IRelationshipClass pRelClass, IClassBreaksRenderer pRenderer)	Creates a relation between the specific occupancy view of GBS table in SQL server to the Census tract boundary and based on a specific type of attribute creates a range theme.
	GetGBSGOccuTheme(int iTblType, IRelationshipClass pRelClass, IClassBreaksRenderer pRenderer)	Creates a relation between the general occupancy view of the GBS table in SQL server to the Census tract boundary and based on a specific type of attribute creates a range theme.
	GetGBSMbtTheme(int iTblType, IRelationshipClass pRelClass, IClassBreaksRenderer pRenderer)	Creates a relation between the specific building type view of the GBS table in SQL server to the Census tract boundary and based on a specific type of attribute creates a range theme.

Interface	Methods	Description
	GetGBSGbtTheme(int iTblType, IRelationshipClass pRelClass, IClassBreaksRenderer pRenderer)	Creates a relation between the general building type view of the GBS table in SQL server to the Census tract boundary and based on a specific type of attribute creates a range theme.
	GetInvtdmgStateTheme(int iTblType, IRelationshipClass pRelClass, IClassBreaksRenderer pRenderer)	Creates a relation between the Inventory damage view in SQL server to the corresponding inventory feature class and based on a specific type of attribute creates a range theme.
	GetInvtdmgFuncTheme(int iTblType, IRelationshipClass pRelClass, IClassBreaksRenderer pRenderer)	Creates a relation between the Inventory functionality view in SQL server to the corresponding inventory feature class and based on a specific type of attribute creates a range theme.
	GetPplDmgTheme(int iTblType, IRelationshipClass pRelClass, IClassBreaksRenderer pRenderer)	Creates a relation between the pipeline damage view in SQL server to the corresponding inventory feature class and based on a specific type of attribute creates a range theme.

Interface	Methods	Description
	GetInvtLossTheme(int iTblType, IRelationshipClass pRelClass, IClassBreaksRenderer pRenderer)	Creates a relation between the Inventory loss view in SQL server to the corresponding inventory feature class and based on a specific type of attribute creates a range theme.
	GetContourTheme(int iTblType, IRelationshipClass pRelClass, IClassBreaksRenderer pRenderer)	Creates a range theme based on a specific attribute on a grid to show the contours.
	GetBndryTheme(int iTblType, ISimpleRenderer pRenderer)	Defines the symbol and color with which the boundary map is to be displayed
	GetDataMapTheme(int iTblType, IUniqueValueRenderer pRenderer)	Creates a unique value theme based on the datamap attribute.

7. Detailed Design Description

This chapter will present in the detail all the complete design descriptions for the objects described in the previous chapters.

The grouping of the detail design descriptions follows overall the 4-tier architecture implementd in HAZUS-MH, namely: the PSL, the ALL, the WFL, and the DAL;

7.1. Database Access Layer (DAL)

7.1.1. Security Model for the HAZUS-MH Database

The security model for HAZUS-MH is simplified in the latest version of the HAZUS-MH data model such that all access to the databases will via a unique user-account with administrator privilege.

There will be no distinction between users. All users are flattened to a virtual user account called simply hazuspuser. For a minimal security, the standard account is kept separate and is password-protected as shown in the table below:

Table 7-1. HAZUS-MH Database Access Accounts

	User name	Type	Password	Access Rights	Description
1	sa	User account	gohazusplus!!!	sysadmin	SQL-Server built-in account.
2	hazuspuser	User account	gohazusplus!!!	sysadmin	HAZUS-MH user account

In this implementation, both the sa and hazuspuser accounts are similar, but only the latter is used by HAZUS-MH data components. This allows future changes to be implemented without major side effects.

The HAZUS-MH setup program creates both accounts automatically. Since HAZUS-MH is running under W2K, the authentication mode in MSDE/SQL2K need to be set to the mixed mode "SQL Server and Windows".

Creating an MSDE/SQL2K login needs to be done just once at the server level. Use the stored procedure `sp_grantlogin` to add in a domain user, and use `sp_addlogin` to add a non-domain users as follows:

```
sp_grantlogin [@loginame =] 'login'
```

or

```
sp_addlogin [ @loginame = ] 'login'
    [ , [ @passwd = ] 'password' ]
    [ , [ @defdb = ] 'database' ]
    [ , [ @deflanguage = ] 'language' ]
    [ , [ @sid = ] sid ]
    [ , [ @encryptopt = ] 'encryption_option'
```

Example:

```
EXEC sp_grantlogin 'dti\mouradb'
```

Or

```
EXEC sp_addlogin 'hazuspuser', 'gohazusplus!!!'
```

At installation, HAZUS-MH will create the HAZUS-MH user account. In addition, the sa account -which is created automatically by MSDE/SQL2K-, is updated by changing its password to 'gohazusplus!!!' (Originally, it is blank).

Task 1 – Change sa password

```
-- Change 'sa' password
exec sp_password '', 'gohazusplus!!!', 'sa'
go
```

Task 2 – Creating hazuspuser Account

```
-- Create HAZUS-MH user account. Give account administrator role
exec sp_addlogin 'hazuspuser', 'gohazusplus!!!'
go
exec sp_addsrvrolemember 'hazuspuser', 'sysadmin'
go
```

7.1.2. Database Schemas and SQL Scripts

Database schema is constructed from the HAZUS-MH Physical Data Model; refer to Appendix A and Appendix B at the end of this document for the complete database schemas for two HAZUS-MH databases: the system database and the template database.

The HAZUS-MH databases are created at run-time using a set of generated SQL Scripts.

Table 7-2 below gives the full list of the SQL scripts used in HAZUS-MH.

Table 7-2. List of HAZUS-MH SQL Scripts

File Name	Description
tlHazus.sql	HAZUS-MH template database
syHazus.sql	HAZUS-MH System database
tlHazus T&P.sql	HAZUS-MH triggers & stored procedures for the template database
syHazus T&P.sql	HAZUS-MH triggers & stores procedures for the system database
eqErrorHandling.sql	Custom SQL2K/MSDE error messages script. Adds message to SQL system table <code>sysmessages</code>

7.1.3. Data Access Methods

ADO is the technology that will be used to access and manipulate data in HAZUS-MH. As per the requirements, all the data is stored in SQL2K (Microsoft SQL Server 2000) Desktop Engine format (formerly MSDE). Access via OLEDB may be used for cases where performance issues might arise.

Table 7-3 shows each HAZUS-MH modules and DAL functionality used.

Table 7-3. DAL Modules

	Modules				
	Aggregation	Browser	Mapping	Analysis	Reports
Transaction	X			X	

View	X	X			
Stored Procedure	X	X	X	X	X
Triggers		X	X	X	
Error Handling	X	X	X	X	X
Performance, Tuning, and Optimization	X	X	X	X	X

7.1.4. Aggregation Process in DAL

The aggregation process consists of transferring data object from the packaged format on the CDs and system folder to the new study region based on the user's selection of state/counties/tracts and census.

The component ddtiDTS is responsible for aggregating the data, and is based on the use of the Data Transformation Services (DTS) services in MSDE.

The DTS packages are designed and saved as templates, then called and updated from ddtiDTS component connection, source and destination paths are modified, and then executed. Each DAL component has a package template file as listed in Appendix C: HAZUS-MH DTS Packages. Each package consists of several tasks, with each task performing a specific data transfer. Figure 7-1 below shows a diagram with the steps needed to implement the aggregation process in the DAL.

Aggregation Process In DAL

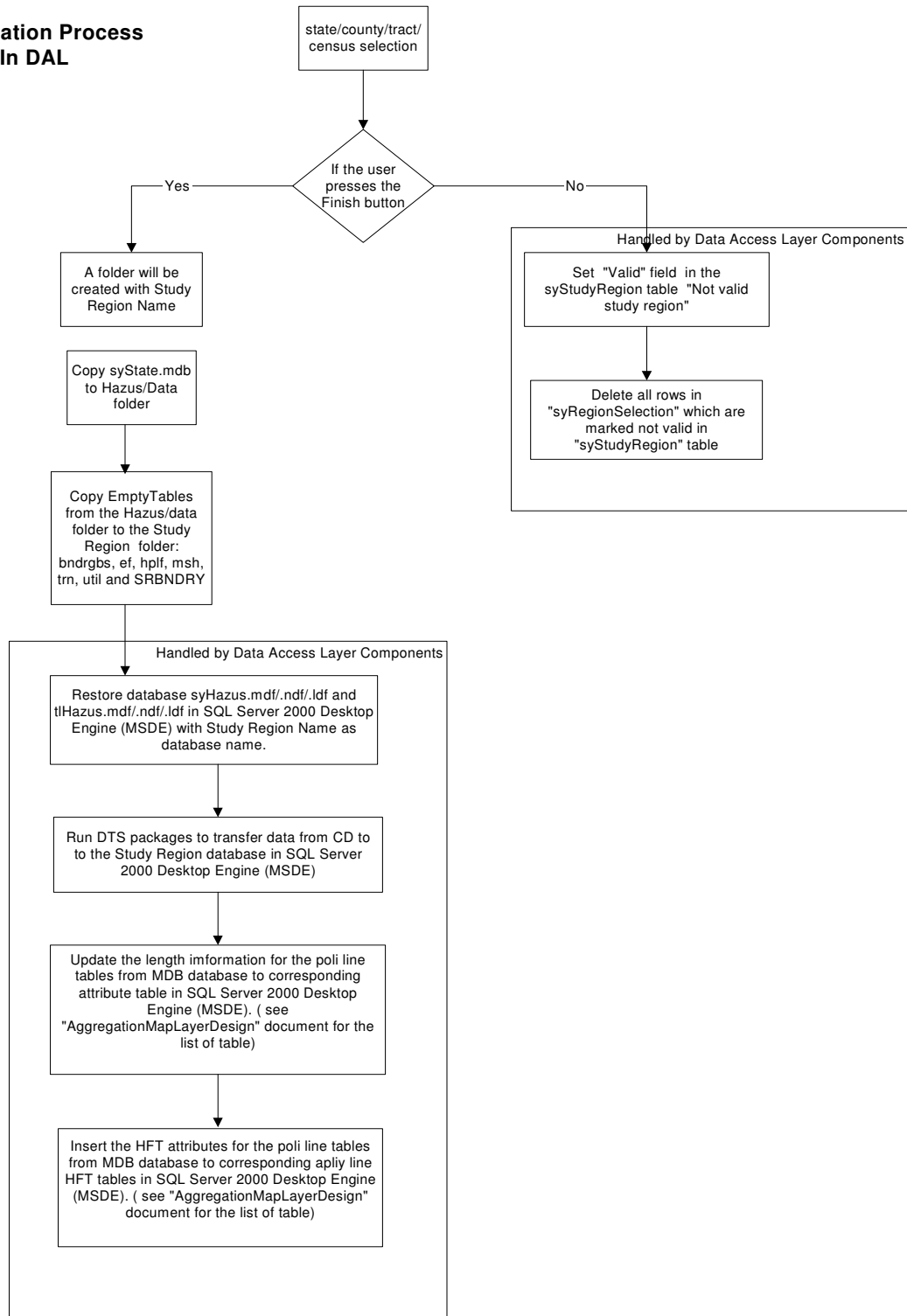


Figure 7-1. Aggregation Process Flow in DAL

7.2. Application Logic Layer (ALL)

7.2.1. The Study Region Aggregation Engine

This section describes the general procedure that should be followed for the implementation of the Aggregation Layer component for HAZUS-MH.

As all other engines, the aggregation engine relies on the n-tier architecture, and therefore make uses of other engines in different layers, mainly the Mapping Engine Component in PSL and the Data Components in DAL.

The aggregation map layer will be composed in “adtiAggregation.dll” component. AML is implemented using COM technology.

Aggregation Process Flow

The aggregation process starts with the launch of the Study Region Aggregation Wizard. The user is provided with the option to select a study region to aggregate either by name or from the map. The study region can be selected at the State level or down to the census block level.

When the user is finished with the selection, the wizard calls the methods from the Aggregation Component to perform the aggregation. The aggregation component performs the aggregation by calling methods from the Mapping Components and the Data Component. Steps for the aggregation process are shown in Figures 7-2 to 7-6.

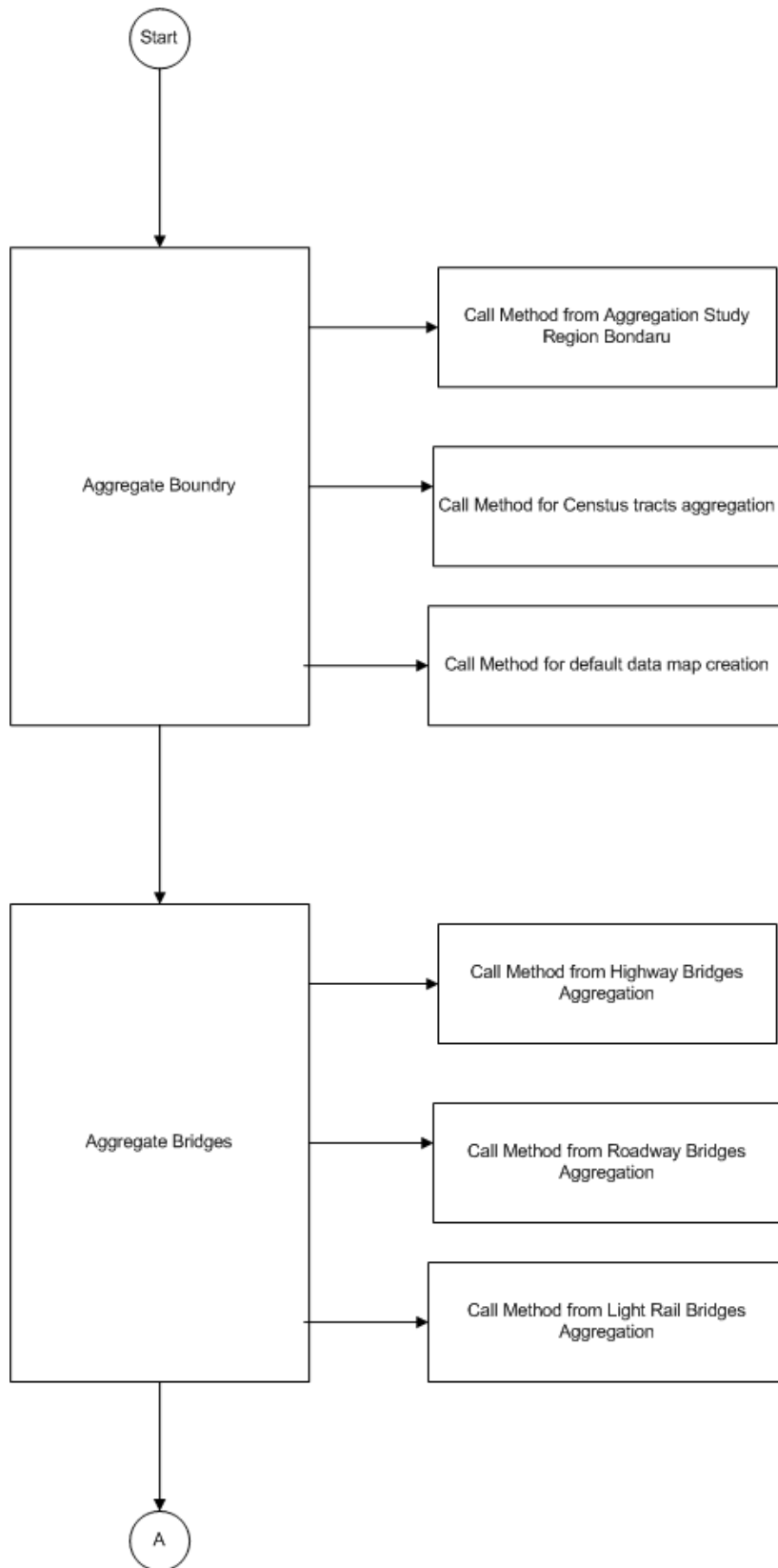


Figure 7-2. Aggregation Flow Process Part 1

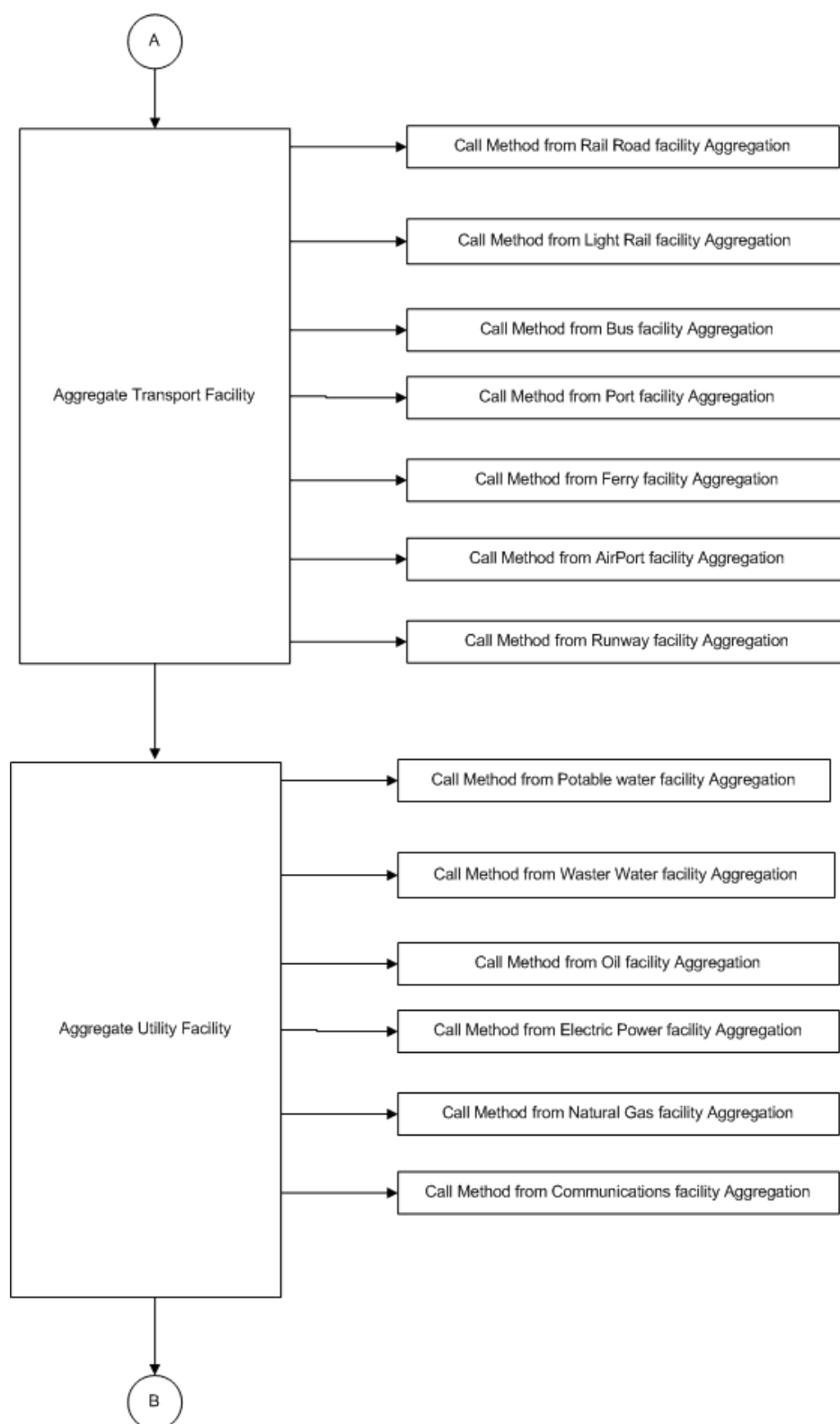


Figure 7-3. Aggregation Flow Process Part 2

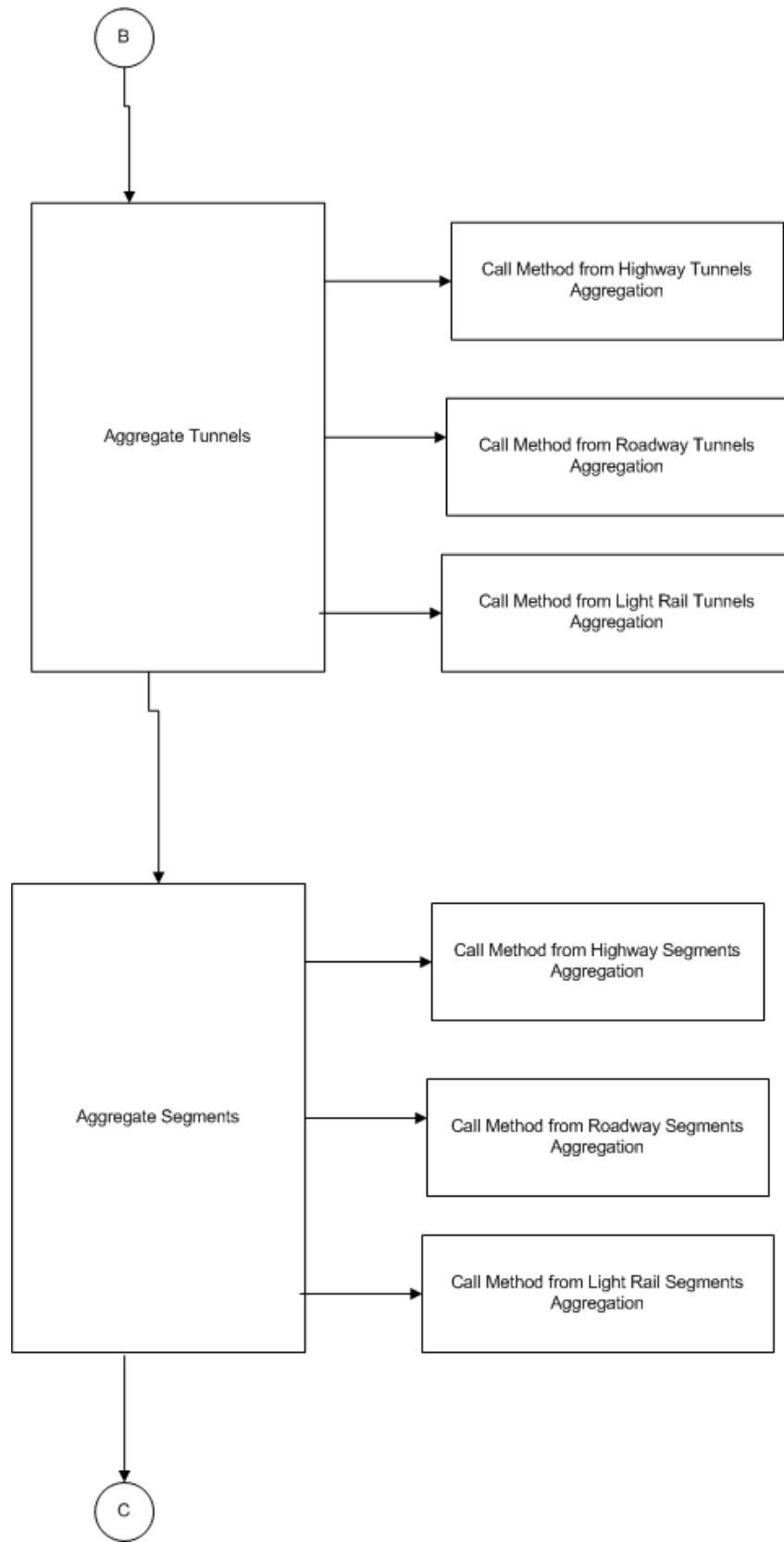


Figure 7-4. Aggregation Flow Process Part 3

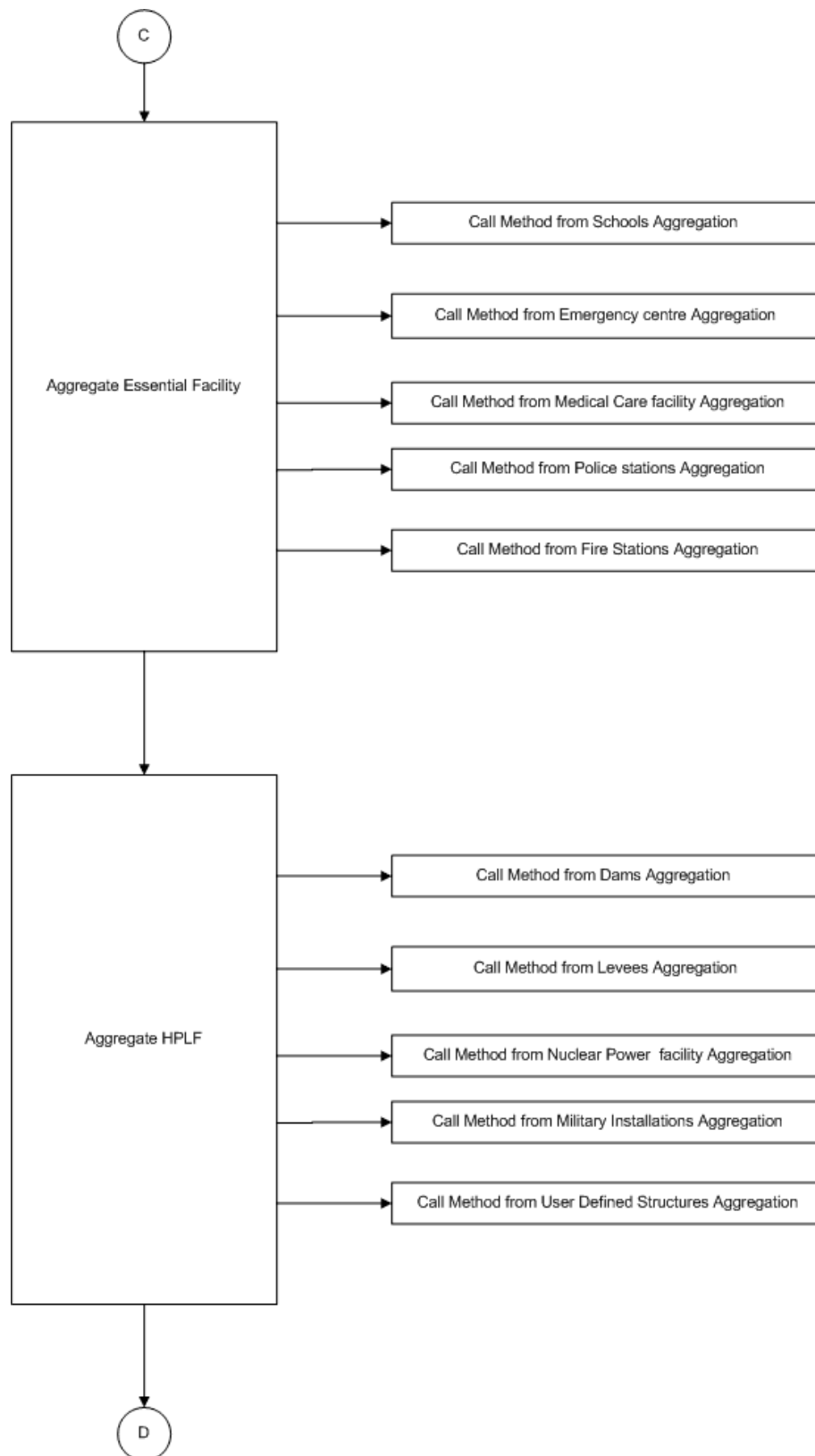


Figure 7-5. Aggregation Flow Process Part 4

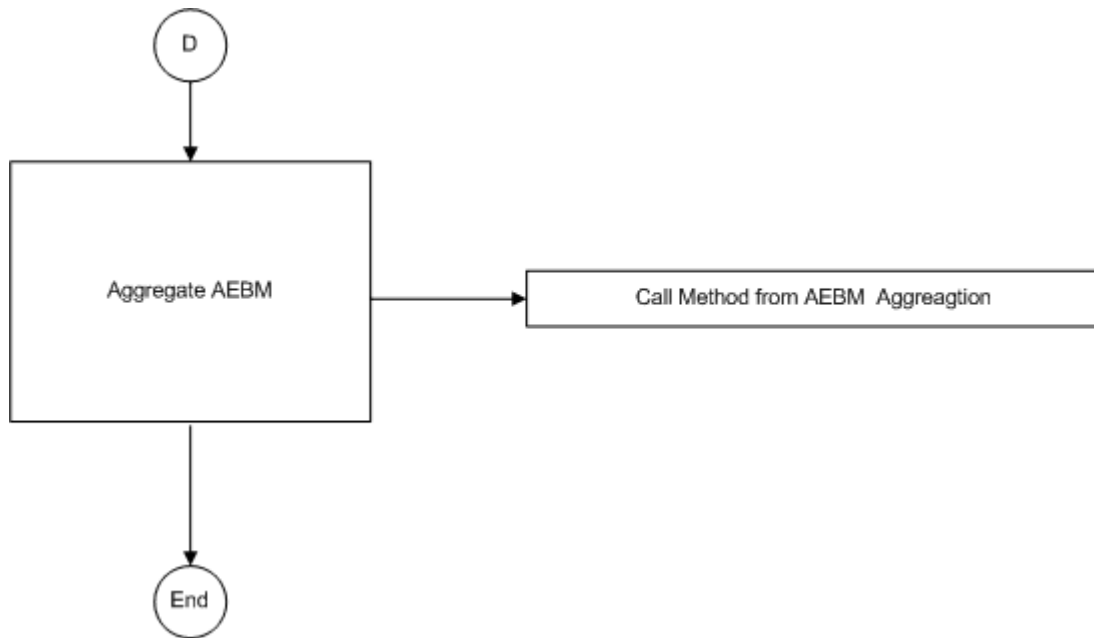


Figure 7-6. Aggregation Flow Process Part 5

7.2.2. The Analysis Engine

This section describes the general procedures that should be followed for the implementation of Analysis Engine Module for HAZUS-MH, which we will refer to as AEM.

AEM will implement the entire analysis algorithm per the official HAZUS-MH Technical Manual. In addition, it will house all the supporting logic to carry out optimally the different analysis algorithms, such as the dependency check component.

The design philosophy used in the AEM architecture is to have it inventory based instead of module based, in other words, the analysis modules are grouped by inventory type (i.e. all the types of calculation which can be performed on bridges for example are grouped together like ground motion, damage, functionality, etc), which matches the HAZUS-MH data model and simplifies greatly the interface.

7.2.2.1. Demand Module

The EQ demand module is a standardized function that is called by all the HAZUS inventory components, for which analysis is expected. This function “EQModule()” returns the following 12 parameters/values:

1. Distance to the ruptured fault plane or epicenter, whichever is applicable when applicable.
2. Peak Ground Acceleration in g's at the specific site or census tract centroid.
3. Spectral Acceleration at 0.3 seconds in g's at the specific site or census tract centroid.
4. Spectral Acceleration at 1.0 second in g's at the specific site or census tract centroid.
5. Peak Ground Velocity in cm/sec at the specific site or census tract centroid.
6. Amount of settlement due to liquefaction in inches at the specific site or census tract centroid.
7. Amount of lateral spreading due to liquefaction in inches at the specific site or census tract centroid.
8. Probability of liquefaction at the specific site or census tract centroid.
9. Amount of ground deformation due to landslide in inches at the specific site or census tract centroid.
10. Probability of landslide at the specific site or census tract centroid.
11. Amount of ground deformation due to surface fault rupture in inches at the specific site or census tract centroid.
12. Probability of surface fault rupture at the specific site or census tract centroid.

Figure 7-7 below illustrates schematically the various arguments/variables used by the demand module to provide the required output information.

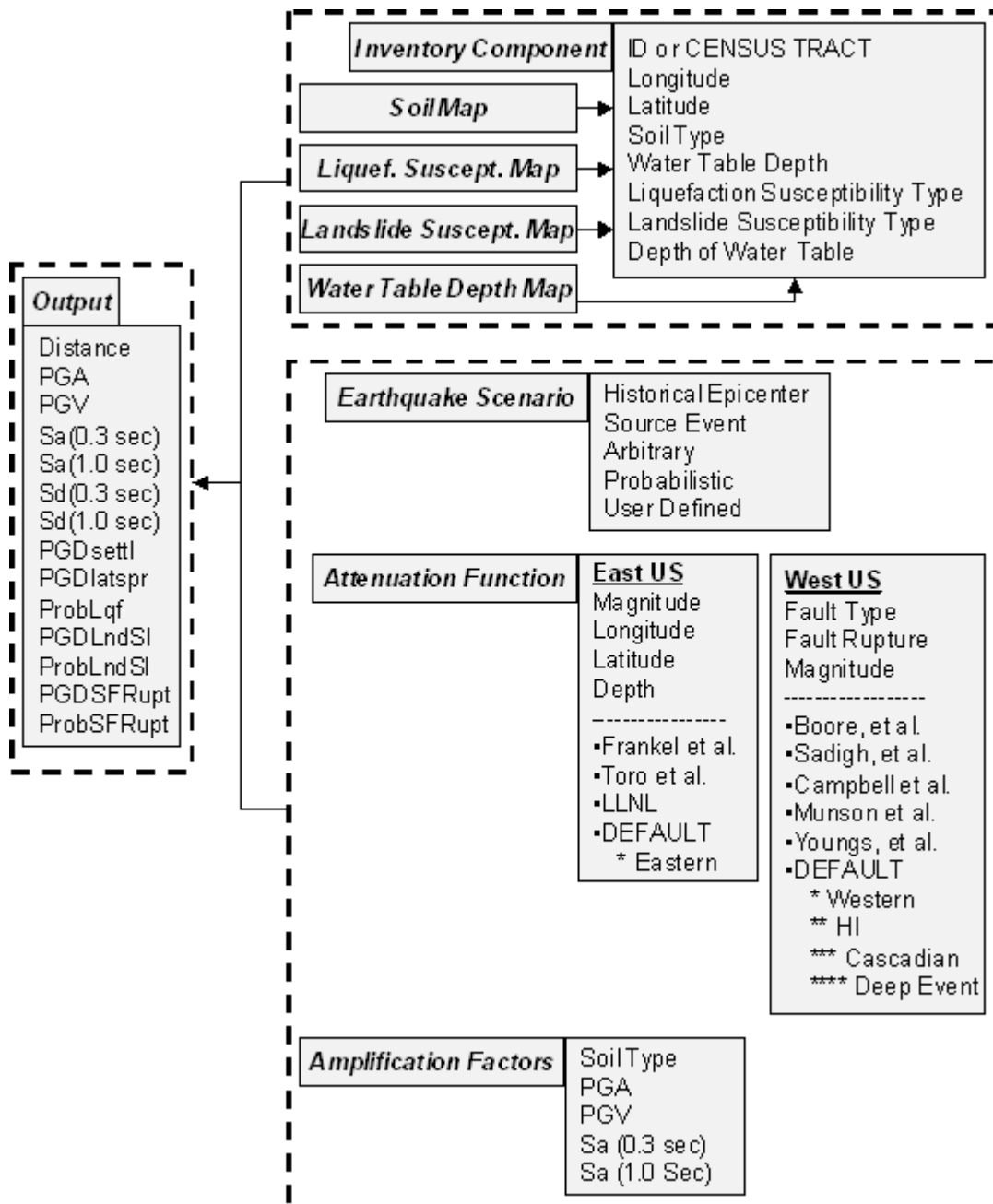


Figure 7-7. Snapshot at the Demand Module

7.2.2.1.1. Overall Approach

In generating the required output information, the EQDemand module of the analysis engine executes the following 10 steps:

Step 1: Obtain information on whether the study region lies in Eastern/Central US or Western US.

Step 2: Obtain the code for the attenuation function to be used. This step includes reading the magnitude specified by the user, the scenario option selected by the user, and accordingly reading the longitude & latitude of the epicenter and/or the geometric configuration of the ruptured fault plane.

Step 3: Read the analysis option parameters selected by the user. This step includes checking whether the hazard data, such as liquefaction susceptibility map and/or landslide susceptibility map, have been specified and will be used in the analysis.

Step 4: For each of the inventory component to be analyzed, whether site specific data or census tract, read the longitude and latitude of the centroid.

Step 5: Except for the User Defined or Probabilistic scenario selected option, compute the minimum distance between the centroid and the epicenter or to the ruptured fault plane. If the User Defined or Probabilistic scenario is the selected option, then skip this step.

Step 6: Except for the User Defined or Probabilistic scenario selected option, compute the PGA, PGV, and Spectral Values based on this minimum distance using the appropriate attenuation function determined in step 2. If the User Defined or Probabilistic scenario option is selected, then do a point and polygon GIS overlay to read directly the PGA, PGV, and Spectral Values. The attenuation relationships for the Western U.S. are implemented such that the demand is computed using the formulas. For Eastern U.S. the values are pre-computed for magnitude and distance and the program performs a lookup.

Step 7: Check for any special cases such as if B&J attenuation, the moment magnitude may be modified.

Step 8: Except for the User Defined or Probabilistic scenario selected option, modify the PGA, PGV, and Spectral Values from Step 6 and 7 for soil amplification. If the User Defined or Probabilistic scenario is the selected option, then skip this step.

Step 9: From step 3, if the liquefaction susceptibility map, water table depth map, and/or landslide susceptibility maps have been provided and to be used in the analysis, compute the ground deformation parameters. This step will use the PGA information computed in step 7.

Step 10: Save and close the computed demand values.

7.2.2.1.2. Detailed Algorithm

In this section, the algorithms for each of the 10 steps described in the previous section are further outlined and detailed.

7.2.2.1.2.1. Step 1: East or West US

Based on the counties included in the study region, the program checks for whether it is a region defined as in the Eastern /Central US or Western US (WUS). WUS regions include locations in, or west of, the Rocky Mountains, Hawaii, and Alaska.

7.2.2.1.2.2. Step 2: Scenario Option Details

In the case where no User Defined or Probabilistic scenario option has been selected, the user will need to specify an attenuation function and a moment magnitude.

The attenuation relationships provided for Western United States (WUS) sites are based on:

- Boore, Joyner & Fumal (1993, 1994a, 1994b) - shallow crustal earthquakes
- Sadigh, Chang, Abrahamson, Chiou, and Power (1993) - shallow crustal earthquakes
- Campbell and Bozorgnia (1994) - shallow crustal earthquakes (PGA only)
- Munson and Thurber (1997) - Hawaiian earthquakes (PGA only)
- Youngs, Chiou, Silva and Humphrey (1997) - deep and subduction zone earthquakes
- Cocktail attenuation function for Western US

For sites in the Central and Eastern United States (CEUS), the attenuation relationships are based on:

- Frankel et al. (1996), Savy (1998) and Toro,
- Abrahamson and Schneider (1997).
- Cocktail attenuation function for Eastern/Central US

Depending on the attenuation relation, read either regression coefficients (Western US).

For the WUS, the attenuation relationships require the user to specify the type and orientation of the fault associated with the selected epicenter. In addition, the algorithm uses the moment magnitude to compute the expected values of surface and subsurface fault rupture length. Fault rupture length is based on the relationship of Wells and Coppersmith (1994) given below:

$$\log_{10}(L) = a + b \cdot M$$

where: L is the rupture length (km)

M is the moment magnitude of the earthquake

a and b are coefficients read from the table below (depending on the options selected by the user)

Table 7-4. Rupture Length Coefficients

Rupture Type	Fault Type	a	b
Surface	Strike Slip	-3.55	0.74
	Reverse	-2.86	0.63
	Normal	-3.55	0.74
	All	-3.22	0.69
Subsurface	Strike Slip	-2.57	0.62
	Reverse	-2.42	0.58
	Strike Slip	-2.57	0.62
	All	-2.44	0.59

Fault rupture is assumed to be of equal length on each side of the epicenter, provided the calculated rupture length is available in both directions along the specified fault segment. If the epicenter location is less than one-half of the rupture length from an end point of the fault segment (e.g., the epicenter is located at or near an end of the fault segment), then fault rupture length is truncated so that rupture does not extend past the end of the fault segment. If the calculated rupture length exceeds the length of the fault segment, then the entire fault segment is assumed to rupture between its end points, unless the fault is connected to other fault segments. In the case where multiple faults segments share common endpoints (i.e. the segments are connected), the methodology provides the user with the ability to create an earthquake rupture across multiple segments.

For Eastern/Central US, the function performs the lookup into the table for the spectral value corresponding to the distance of epicenter from the centroid and the moment magnitude. If it does not find the precise value then it does interpolation.

If the user selects the option for a user-defined or a probabilistic scenario, then the EQDemand() function for Step 2 becomes a point/line and polygon overlay GIS operation.

7.2.2.1.2.3. Step 3: User Options

The user may select one of the following six possible options:

- Use default assumptions
- Use actual soil
- Use actual soil and liquefaction susceptibility and water depth information but no landslide susceptibility
- Use actual soil and liquefaction susceptibility but no water depth information and no landslide susceptibility
- Use actual soil and landslide susceptibility but no liquefaction susceptibility
- Use actual soil, liquefaction susceptibility, water depth information, and landslide susceptibility

The algorithm is optimized depending on the options selected by user as indicated by the following flowchart:

7.2.2.1.2.4. Step 4: Input Information

If the inventory item being analyzed is a census tract, then read the longitude and latitude of the census tract centroid. If the inventory component is facility, then read longitude and latitude of that facility. If the component is a line (road, track or pipe), then sample few points on the line based on the user defined spacing and output the average of the results at the locations of these sampling points.

7.2.2.1.2.5. Step 5: Minimum Distance

If the scenario option is user defined or probabilistic then skip steps 5 and 6. If not, then five cases are possible in computing the minimum distance:

- Minimum distance between two points. The algorithm implemented here is similar to the current algorithm implemented for Eastern/Central US scenarios
- Minimum distance between a point and a line. This algorithm applies for vertical fault rupturing. In this case, the algorithm computes distance of the point from the end point of the line segment. If it lies then it returns the zero as the distance between the point and the line. It checks if the point lies on the line segment. If the point is not on the line segment, then it computes the cosine of angles between the line segment and the lines joining the point to the end points of the line segment. If the angle between the line segment and line joining the point to an end point of the line segment is obtuse angle then the point lies outside towards that end point of line segment and the function returns the distance between that end point and the point as the minimum distance. Otherwise, it computes the minimum distance using the following formula, $\text{MinDistance} = \text{DistFromPt2} * \text{SQRT}[1 - \text{SQRT}(\text{Cos} \theta)]$, here, θ = Angle between the line segment and line joining the point to the second end point of the line segment.
- Minimum distance between a point and polyline. This is a new algorithm that is currently under developed.
- Minimum distance between a point and single plane. The algorithm implemented here is similar to the current algorithm implemented for Western US scenarios
- Minimum distance between a point and multi-planes. This is a new algorithm that is currently under developed.

7.2.2.1.2.6. Step 6: Special Cases

If the attenuation function is BJ and the moment magnitude scenario is greater than 7.7, then modify the magnitude as per the following equation, $\text{Magnitude} = 7.7 + (\text{Magnitude} - 7.7) * 0.5$

From $M = 7.0 - 7.7$, the magnitude term becomes $0.316*(7.0) + 0.216*(M-7.0)$. For $M > 7.7$, a magnitude term is set to a constant value equal to $0.316*(7.0) + 0.216*(7.7-7.0)$.

7.2.2.1.2.7. Step 7: Attenuation or Predefined PESH

For WUS, the weighting rules for default combinations of attenuation functions are summarized below:

- Western US Shallow Crustal Events

Peak Ground Acceleration:

one-third BJF 1994 relationship
one-third Sadigh 1993 relationship
one-third Campbell & Bozorgnia 1994 relationship
(for $r > 60$ km, one-half BJF 1994 and one-half Sadigh 1993; for $M > 7.7$ BJF 1994 is not used)

Spectral Acceleration:

one-half BJF 1994 relationship
one-half Sadigh 1993 relationship
(for $M > 7.7$ only Sadigh 1993 is used)

- Deep Events (e.g., Puget Sound Earthquakes > 50 km in depth):

Youngs 1997 - Intralab relationship

- Cascadia Subduction Zone:

one-half Youngs 1997 - interface relationship
one-half Sadigh 1993 - reverse-slip relationship
(only Youngs 1997 is used for magnitudes greater than $M = 8.0$)

- Hawaiian Events ($M < 7.0$)

Peak Ground Acceleration:

one-fourth BJF 1994 relationship
one-fourth Sadigh 1993 relationship
one-fourth Campbell & Bozorgnia 1994 relationship
one-fourth Munson & Thurber 1997 relationship

Spectral Acceleration:

0.3 Seconds

one-third BJF 1994 relationship
one-third Sadigh 1993 relationship
one-third 2.5^* (Munson & Thurber 1997 relationship)

1.0 Seconds

one-half BJF 1994 relationship
one-half Sadigh 1993 relationship

- Hawaiian Events ($M \geq 7.0$)

Peak Ground Acceleration:

- one-half Sadigh 1993 relationship
- one-half Munson & Thurber 1997 relationship

Spectral Acceleration:

0.3 Seconds

- one-half Sadigh 1993 relationship
- one-half 2.5*(Munson & Thurber 1997 relationship)

1.0 Seconds

- Sadigh 1993 relationship

The weighting rules for default combinations of attenuation functions in Eastern/Central US of are:

Peak Ground Acceleration:

- one-half Frankel 1996 relationship
- one-half Toro 1997 relationship

Spectral Acceleration:

- one-half Frankel 1996 relationship
- one-half Toro 1997 relationship

Unless supplied by the user (i.e., as user-supplied PGV maps), peak ground velocity (inches per second) is inferred from 1-second spectral acceleration, SA1 (units of g), using the Equation below:

$$PGV = \left(\frac{386.4}{2\pi} \cdot S_{A1} \right) / 1.65$$

7.2.2.1.2.8. Step 8: Soil Factors

In the absence of a soil map, the algorithm will amplify the ground motion demand assuming Site Class D soil at all sites. If the soil map is specified and depending on the soil type, the value of amplification factor is read from the amplification factor table and returns it to the calling function: GetAmplFact().

Table 7-5. Soil Amplification Factors

Site Class B Spectral Acceleration	Site Class				
	A	B	C	D	E
Short-Period, S_{AS} (g)	Short-Period Amplification Factor, F_A				
≤ 0.25	0.8	1.0	1.2	1.6	2.5
0.50	0.8	1.0	1.2	1.4	1.7
0.75	0.8	1.0	1.1	1.2	1.2
1.0	0.8	1.0	1.0	1.1	0.9
≥ 1.25	0.8	1.0	1.0	1.0	0.8*
1-Second Period, S_{A1} (g)	1.0-Second Period Amplification Factor, F_v				
≤ 0.1	0.8	1.0	1.7	2.4	3.5
0.2	0.8	1.0	1.6	2.0	3.2
0.3	0.8	1.0	1.5	1.8	2.8
0.4	0.8	1.0	1.4	1.6	2.4
≥ 0.5	0.8	1.0	1.3	1.5	2.0*

The Methodology amplifies rock (Site Class B) PGA by the same factor as that specified in Table above for short-period (0.3-second) spectral acceleration and amplifies rock (Site Class B) PGV by the same factor as that specified for 1.0-second spectral acceleration.

7.2.2.1.2.9. Step 9: PGD Collateral Effects

Permanent ground deformation (PGD) computations due to liquefaction, landslide, and/or surface fault rupture are only enabled if the user chooses to include them as indicated in step 3.

Table 7-6. Functions Used in the PGD Computations.

Name	Description
-------------	--------------------

GetGFValues	It checks the options to see if the PGD is to be computed for liquefaction, landslide or surface fault rupture. Depending on the options it calls 'GetPGDduetoLQF()', 'GetPGDduetoLND()' or 'GetPGDduetoSFR()'.
GetPGDduetoSFR	It computes the PGD and probability of PGD from surface fault rupture.
GetPGDduetoLND	It computes the PGD and probability of PGD from landslides. First, it computes the critical acceleration by calling GetCriticalAcc(). Next, it computes acceleration by induced acceleration.
GetPGDduetoLQF	It computes the PGD due to ground settlement and lateral spreading. First it calls 'ComputeSettlementPGD()' and 'ComputeLateralSpreadingPGD()'. Next it computes the probability by calling 'ComputeLQFProb()'.
ComputeSettlementPGD	Depending on the susceptibility category it assigns the value of PGD.
ComputeLateralSpreadingPGD	It computes the PGD due to lateral spreading using eq. 4-26 in 100% technical manual.
ComputeLQFProb	It computes the probability of liquefaction
ComputeDispFctr	Given a value of Critical Acceleration by Induced acceleration, it determines the associated displacement factor
GetCriticalAcc	Depending on the susceptibility category, it returns the corresponding critical acceleration.

7.2.2.1.2.10. Step 10: Saving the computed demand values

For each facility, all 14 computed values by EQModule() will be stored in the same database containing the other inventory attributes. For the case of census tracts, the computed values will be stored EQTractsDemand Table.

7.2.2.2. Damage State Module

The EQ damage state module computes the expected direct physical damage to inventory components. There are four distinct approaches for computing and expressing physical damage in the earthquake methodology:

- One approach for the general building stock and essential facilities. The methodology of this approach uses the building capacity curve, building fragility curve and spectral demand curve to compute the damage state probabilities (probability of reaching or exceeding a damage state).
- The second approach is for pipelines, where the physical damage is expressed in terms of leaks and breaks.
- The third approach is for bridges where the physical damage is expressed in terms of probabilities of reaching or exceeding a damage state, with the ground motion parameter being the spectral acceleration at 1.0 second.
- The final approach is for all other lifeline components where the physical damage is also expressed in terms of probabilities of reaching or exceeding a damage state, with the ground motion parameter being the peak ground acceleration.

Figure 5-8 below shows an overview of where the EQ damage state module fits in the overall architecture of the analysis engine for the site-specific data,

The remainder of this section describes the algorithms that will be implemented for each of these damage state approaches.

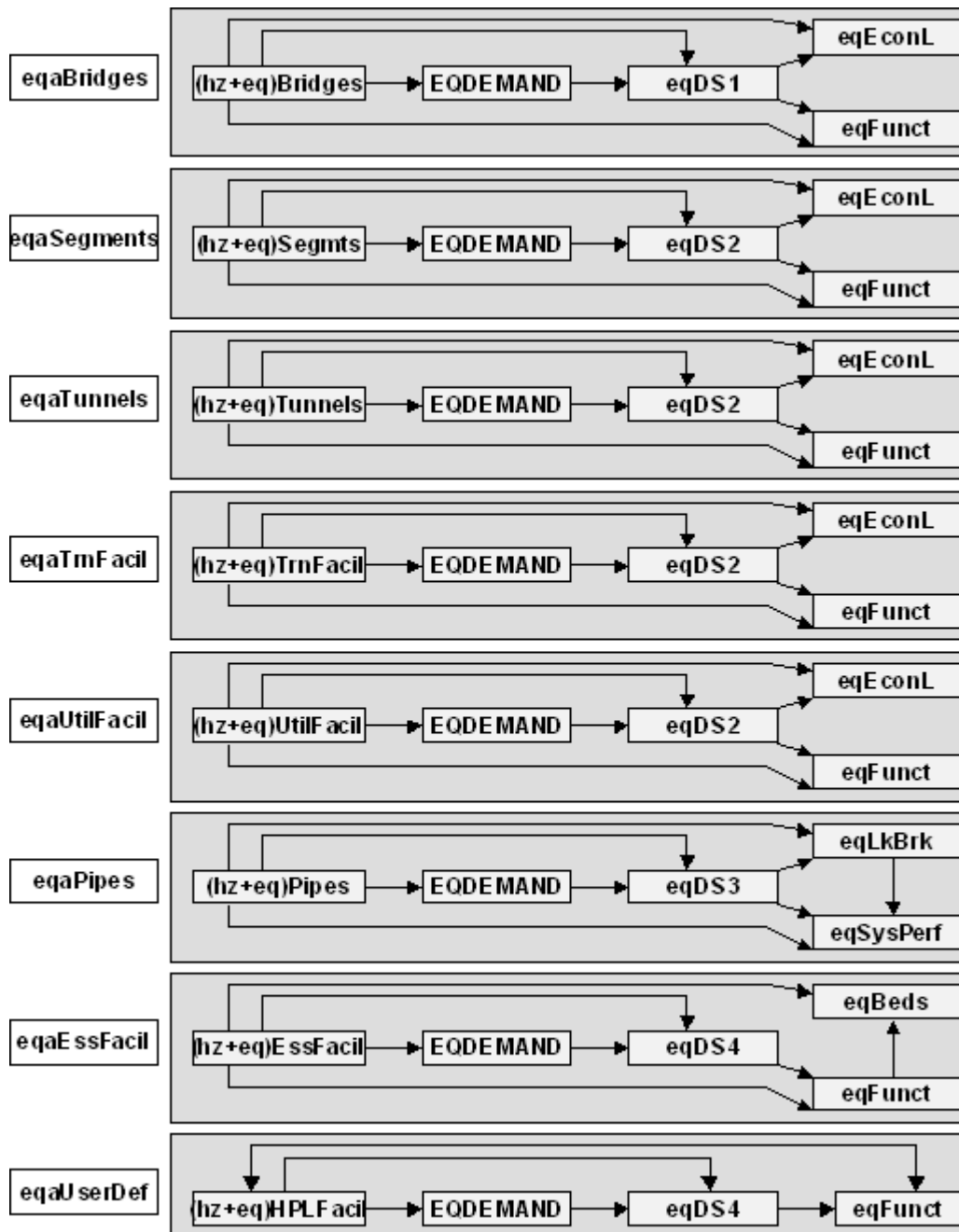


Figure 7-8. Simplified View of the Analysis Engine for the HAZUS Site-Specific Data

7.2.2.3. Functionality Module

TBD.

7.2.2.4. Economic Loss Module

TBD.

7.2.2.5. System Performance Module

TBD.

7.2.2.6. Casualties Module

TBD.

7.2.2.7. Advanced Engineering Building Module

TBD.

7.2.2.8. Annualized Loss Module

TBD.

7.2.2.9. Shelter Module

TBD.

7.2.2.10. Debris Module

TBD.

7.2.2.11. Inundation Module

TBD.

7.2.2.12. Fire Following Module

TBD.

7.2.2.13. Indirect Economic Loss Module

TBD.

7.3. Presentation Services Layer (PSL)

Only the HAZUS-MH specific menus will be described in this section. The standard ArcMap menu options will not be described here unless needed. Refer to ESRI's documentation for details on those menu options.

In support of the PSL, a HAZUS-MH EQ. Prototype has been developed which duplicates the exactly the UI as to be implemented. Refer to it to sense the real look-and-feel of the application.

7.3.1. Study Region Aggregation Wizard

Refer to the Software Design Descriptions of the Shell.

7.3.2. Inventory Menu

TBD

7.3.3. Hazard Menu

TBD

7.3.4. Analysis Menu

TBD

7.3.5. Results Menu

TBD

7.4. The Custom UI Objects

The custom UI objects are a set of objects with some specific encapsulated functionality that are self-contained and architecturally in any or all of our three layers (presentation, logic and data access).

In the Earthquake Model, three such objects have been identified:

- The Database browser which is a UI object that allows browsing of any table in HAZUS-MH. This is a generic and highly flexible component. The high-flexibility allows this interface objects to be used in 90% of the cases.
- The Mapping Engine is a hybrid implementation, since it combines ArcGIS/ArcMap's functionality with HAZUS-MH requirements.
- The Reporting Engine makes use of Crystal Reports engine to generate close to 40+ summary reports.

7.5. The Data Browser

The Data Browser is the module in HAZUS-MH that is responsible for all the table views in the Earthquake Module. The major operations are:

- Provide tabular view for all the tables (both inventory and results)

- Allow the user to edit data on the tabular view for all inventory data
- Allow user to add, delete, and import records for all inventory data other than General Building Stock
- Allow user to export data for all inventory and results
- Allow user to apply a filter to the tabular view
- Allow user to calculate statistics for all inventory and results
- Allow user to view analysis information about the results
- Allow user to view metadata associated with the inventory tables
- Allow user to view the data dictionary

The Data Browser component represents the Presentation layer for the Earthquake module. It will be talking to the middle layer, which in turn will be talking to the data layer for all data related operations. It will provide dialogs with tabular view for inventory data and results information.

The Data Browser is a major component of the presentation layer in the HAZUS-MH Earthquake module. Most of the user interactions with the underlying data are going to be handled through the Data Browser.

Architecturally the Data Browser is a set of components that use ATL (Microsoft Active Template Library), MFC (Microsoft Foundation Classes), ADO (ActiveX Data Objects) and Spread Grid Control as major libraries for the implementation of all the User Interface operations.

Figure 7-9 depicts the macro level organization of the Data Browser and its interactions with the other layers in the architecture.

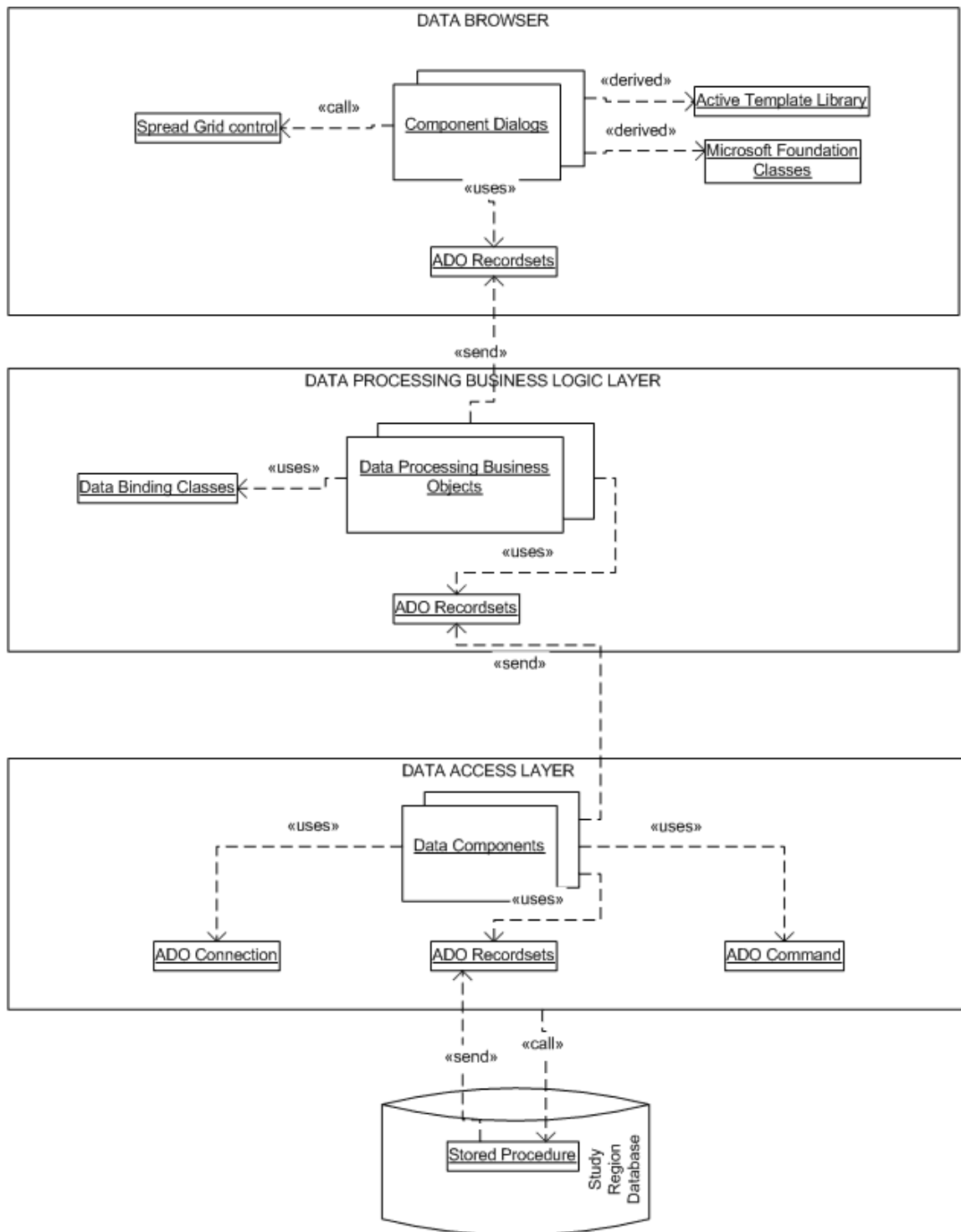


Figure 7-9. Data Browser Architecture

As shown in the diagram above, all the interaction with the Data Browser and other application layers is through the ADO Recordset Objects. These are disconnected recordsets obtained as a result of a Data Component initiating a stored procedure call to the underlying SQL Server based Study Region database.

Overall, the Data Browser will work with disconnected recordset objects having N records; where the number N could be different for different tables being accessed. Every set of N records defines a page of data. The goal is to avoid sending all the records to the client for optimization reasons. At the client side, the user can perform all types of operations on the current page, and the Data Browser will keep updating the disconnected recordset as needed.

The next fetch of the data will be initiated by the Data Browser whenever the user moves, in forward or backward direction, beyond the last or first record in the current page. At that time all the pending updates to the current page will be applied by the Data Component responsible for fetching the next/previous page.

Once the Data Browser receives a recordset, it will populate the Spreadsheet grid with the values in the recordset, applying any validation criteria wherever applicable and also applying the edit-ability options associated to the various fields in the recordset.

7.5.1. Displaying Data in the Spreadsheet Grid on the Dialog

The Data Browser Dialog will work on the concept of paged disconnected recordsets. The paging will be controlled by the data components. The data display process is a collaboration between the Data Browser and the data component associated with the Inventory / result being viewed.

7.5.1.1. Displaying the First Page

The process of displaying the data initially on the dialog proceeds as follows:

1. When any dialog of the data browser is launched, it will make a call to the middle layer component, requesting the data required by the presentation layer.
2. The middle layer component will define the data to be retrieved and pass on the call to the data component.
3. The data component will open the connection to the database, call a stored procedure that returns the first N (N is dynamic and may vary from one view to the other and is part of the data definition passed to the data component) records in the view as a disconnected recordset and another recordset that has all field descriptions and edit-ability options.

4. The data component passes the both the disconnected recordsets to the data browser through the middle layer component which might do some preprocessing on the data before passing it to the Data Browser.
5. The data browser uses the first recordset and populates the spreadsheet grid with the data and the second recordset to set the column names and making them non-editable and displays the dialog to the user. Thus the first page is displayed to the user.

7.5.1.2. Displaying the Previous Page

The user can scroll up and down the page being viewed. The call to fetch the previous page is controlled by the scrolling process. Whenever the user tries to scroll to a record before the first record in the current page, the fetch of the previous page is triggered. If any edits are pending they get committed to the database.

7.5.2. Start Editing

The data browser's right click menu has an option that allows the user to start editing the table being viewed. The step by step process for adding a new record is as follows:

1. First the Data Browser enables the following options on the right click menu: "Add new record", "Delete selected records" and "Import."
2. If the study region was aggregated including the flood hazard, the data that is being viewed is either Square Footage, Building Count, Exposure or Demographics; then the Data Browser switches to the block-wise view of the same data as per the steps outlined for Displaying first page.
3. The Data Browser updates all the columns on the grid as per the edit-ability options specified in the field description recordset.

7.5.3. Add New Record

The data browser's right click menu has an option that allows the user to add a new record to the table being viewed. The step by step process for adding a new record is as follows:

1. The data browser will first check if the last page is being viewed or not. If not then the data browser will make a call to the middle layer component, requesting the

- last page of data required by the presentation layer and returning the recordset representing the current page.
2. The middle layer component may do some post-processing on the returned recordset, define the last page data to be retrieved and pass on the call to the data component with the returned recordset.
 3. The data component will open the connection to the database and will update the returned recordset to the database if there are any pending edits. Then it calls the stored procedure that returns the last N records in the view as a disconnected recordset. The records returned in this recordset are always less than N. IF the number of records in this recordset are equal to N then the data component flags that the last page starts with the new record being added by the user.
 4. The data component passes the disconnected recordset or the flag that the last page starts with the new record being added by the user to the data browser through the middle layer component which might do some preprocessing on the data before passing it to the Data Browser.
 5. If the data browser receives a recordset it uses the recordset and populates the spreadsheet grid, appends a blank row to it and displays the dialog to the user. If the data browser receives the flag that the last page starts with the new record being added by the user then it adds a empty row to the spreadsheet grid and displays it to the user.

7.5.4. Delete Selected Records

The data browser's right-click menu has an option that allows the user to delete selected records from the table being viewed. The step by step process for deleting records is as follows:

1. The user selects the records on the current page and selects the 'Delete' option
2. The Data Browser updates the disconnected recordset by deleting the records from the recordset.
3. This update gets reflected to the database when either the user moves to a different page (as described above), or presses the MAP or CLOSE buttons.
4. When user presses MAP or CLOSE button or the Stop Editing option on the shortcut menu, the data browser will make a call to the middle layer component returning the recordset representing the current page.

5. The middle layer component may do some post-processing on the returned recordset and pass on the call to the data component with the returned recordset.
6. The data component will open the connection to the database and will update the returned recordset to the database if there are any pending edits.

7.5.5. Editing Records

1. Once user has clicked the Start Editing option, he will be able to make the edits to some columns in the grid.
2. The Data Browser keeps updating the disconnected recordset.
3. This update gets reflected to the database when either the user moves to a different page (as described above), presses the MAP or CLOSE buttons, selects 'Stop Editing' option on the context menu.
4. When user presses MAP or CLOSE button or the Stop Editing option on the shortcut menu, the data browser will make a call to the middle layer component returning the recordset representing the current page.
5. The middle layer component may do some post-processing on the returned recordset and pass on the call to the data component with the returned recordset.
6. The data component will open the connection to the database and will update the returned recordset to the database if there are any pending edits.

7.5.6. Import

The data browser's right-click menu has an option that allows the user to import records for Essential Facilities, High Potential Loss Facilities, User Defined Structures, Transportation Systems and Utility Systems from an Access database or a DBF table into the corresponding table. The step by step process for Import is as follows:

1. When user identifies the source for import, the Data Browser calls a data component method that returns a recordset with the data structure of the underlying data in the source.
2. The Data Browser uses that data structure to show user a Field Mapping between the source and the destination. User will have to map the source fields to the destination fields using this.

3. The Data Browser sends the mapping back to the data component that initiates the import from the source to the corresponding table/s in the SQL Server study region database.
4. Once the data import is complete, the Data Component sends the first page back to the Data Browser or sends the details of the error encountered.
5. If the above step is successful, the Data Browser calls the Mapping Engines Import method to initiate the import of the features associated to the attribute data just imported else displays the Error Message to the user.
6. If both the imports are successful, the data browser displays the new recordset on the spreadsheet grid.

7.5.7. Stop Editing

The data browser's right-click menu has an option that allows the user to stop the editing process. The step by step process for stop editing is as follows:

1. When the user clicks the stop editing option, the Data Browser checks for any unsaved edits.
2. If there are any unsaved edits, the Data Browser updates those to the disconnected recordset and then the data browser will make a call to the middle layer component returning the recordset representing the current page.
3. The middle layer component may do some post-processing on the returned recordset and pass on the call to the data component with the returned recordset.
4. The data component will open the connection to the database and will update the returned recordset to the database if there are any pending edits.

7.5.8. Export

The data browser's right click menu has an option that allows the user to Export data to an Access database, Access geo-database, a DBF table or a DBF/SHP format from the corresponding tables in the HAZUS-MH study region database. The step by step process for Export is as follows:

- **If the user wants to export to a Dbase file or Access Database:**
- 2. The Data Browser will call the export method of the data component and pass the recordset for the current page and a flag that indicates if only the recordset is to be exported or the whole table is to be exported.

3. The data component completes the export process and sends a success or failure (with error) back to the Data Browser.
4. Data Browser will display the result of the export operation to the user. The Data Browser will continue to display the same recordset.
 - **If the user wants to export to a geodatabase or SHP/DBF format:**
1. The Data Browser will call the export method of the mapping engine and pass the recordset for the current page and a flag that indicates if only the recordset is to be exported or the whole table is to be exported
2. The Mapping engine will export all the features to the specified geodatabase and returns success or failure to the Data Browser
3. If the earlier operation is successful then the Data Browser will call the exportGeodatabase method of the Data Component and pass the recordset for the current page and a flag that indicates if only the recordset is to be exported or the whole table is to be exported.
4. The Data Component will handle the export process and return success/failure to the Data Browser.
5. Data Browser will display the result of the export operation to the user. The Data Browser will continue to display the same recordset.

7.5.9. Filter

The data browser's right-click menu has an option that allows the user to define and apply filters to the data in the view. The step by step process for Filter is as follows:

1. The user will build the filter using the Filter Wizard.
2. The Data Browser will generate a select statement from the filter and will pass it to the data component along with the recordset that represents the current page
3. The Data Component will make the database connection, will update the returned recordset to the database if there are any pending edits, and close that recordset. Then it calls the stored procedure that executes the select statement passed to it and returns the first N records in the view as a disconnected recordset.
4. The data component passes the disconnected recordset or the error encountered to the data browser through the middle layer component which might do some preprocessing on the data before passing it to the Data Browser.

5. If the data browser receives a recordset it uses the recordset and populates the spreadsheet grid and displays the dialog to the user. If the data browser receives the Error it displays the error to the user and leaves the grid as is, i.e., the grid continues to display the old recordset.

7.5.10. Calculate Statistics

The data browser's right-click menu has an option that allows the user to calculate statistics on numerical data in the view. The step by step process for Calculate Statistics is as follows:

1. The user will select the column on which the statistics are required and will press the Calculate Statistics option on the shortcut menu.
2. The Data Browser will pass the column information to the data component along with the recordset that represents the current page
3. The Data Component will make the database connection; will update the returned recordset to the database if there are any pending edits. Then it calls the stored procedure that calculates the statistics and returns them back to the Data Component.
4. The data component passes the statistics or the error encountered to the data browser through the middle layer component which might do some preprocessing on it before passing it to the Data Browser.
5. When the data browser receives the statistics, it displays them to the user. If the data browser receives the Error it displays the error to the user. In either case the grid remains as is, i.e., the grid continues to display the old recordset.

7.6. The Mapping Engine

The Mapping Engine is the module of the HAZUS-MH application that is responsible for all the Geographic operations that will be required by various Hazard modules. The major operations are:

- Aggregation of all the spatial data for a study region
- Loading the default print layout for maps and provide operations for supporting the layout editing
- Opening the study region boundary maps and the Soil, Liquefaction, Landslide and Water Depth maps

- Mapping of Inventory like Essential Facilities, High Potential Loss facilities, Lifelines, AEBM, etc
- Mapping of General Building Data like Square Footage, Building Count, Dollar Exposure, Demographics, etc
- Editing operations for Inventory like Adding point locations, deleting point locations, moving point locations, importing point and line type inventory
- Mapping of General Buildings and Inventory analysis results based on themes created using specific result values
- Creation of the ruptured fault based on the scenario defined by the user
- Updating of Hazard Factors whenever any or all datamaps change

The mapping engine is distributed at all the three tiers of application design:

- At the Interface layer it is present as the ArcMap document for map view with all the VBA class modules inside it implementing all the menus handlers and related functionality for customizing ArcMap.
- At the application business logic layer as a C++ COM component that interfaces with the Interface layer and implements the logic for handling various types of GIS operations for the HAZUS-MH application
- At the database layer as a C++ COM component that accesses all the geographic data from various Geodatabases, connects them to the corresponding attribute data in SQL Server database and provide the application layer with geographic objects that represent the combined information.

The mapping engine design is based on the three tier architecture. Architecturally the mapping engine is split into three components one at every design tier.

- Hazuseq.mxd is the Arcmap document that formulates the User Interface tier for the HAZUS-MH Earthquake module. This Arcmap document has several VBA class modules in it that implement the entire menu and all the control methods associated to various menus. There is one class module that captures all the editing events. This also serves as the Map View for the HAZUS-MH Earthquake module. Thus this is the controlling component for the Earthquake Hazard program.

- The aditMapper.dll is a COM component that implements the middle tier application logic. The various class modules in Hazuseq.mxd and Aggregation interface call various methods in five interfaces implemented by this component. All the activity on the map and layout window of the Hazuseq.mxd and aggregation of spatial data for a study region are the result of execution of operations within this component.
- The ddtiMapper.dll is the COM component that implements the data tier. Various methods defined in the two interfaces implemented in this component access all the geographic data from various Geodatabases, connects them to the corresponding attribute data in SQL Server database, creates themes based on specific attribute values and provide the application layer will geographic objects that represent the combined information.

Appendix A: HAZUS-MH System Database Schema

Table List

Name	Check Constraint Name	Primary Key
eqEpicenter	CKT_EQEPICENTER	idteqEpicenter
eqFault	CKT_EQFAULT	idteqFault
syControl	CKT_SYCONTROL	idtControl
syRegionScenario	CKT_SYREGIONSCENARIO	idtScenario
syRegionSelection	CKT_SYREGIONSELECTION	idtRegionSelection
syScenario	CKT_SYSCENARIO	idtScenario
syScenarioEq	CKT_SYSCENARIOEQ	idteqScenario
syScenarioFl	CKT_SYSCENARIOFL	idtfIScenario
syScenarioHu	CKT_SYSCENARIOHU	idthuScenario
syState	CKT_SYSTATE	idtStateFips
syStudyRegion	CKT_SYSTUDYREGION	idtStudyRegion

Table eqEpicenter

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqEpicenterID		char(8)	8		TRUE	FALSE
Name		varchar(40)	40		FALSE	FALSE

Table eqFault

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqFaultId		char(8)	8		TRUE	FALSE

Table syControl

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
ControlId		int			TRUE	FALSE
StateFips	2-digit state FIPS code	char(2)	2		FALSE	TRUE
CountyFips		char(5)	5		FALSE	FALSE

Table syRegionScenario

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
RegionID	Study region Id	smallint			TRUE	TRUE
ScenarioId		smallint			TRUE	TRUE

Table syRegionSelection

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
RegionSelectionId		int			TRUE	FALSE
RegionID	Study region Id	smallint			FALSE	TRUE
SelState		char(2)	2		FALSE	FALSE
SelCounty		char(3)	3		FALSE	FALSE
SelTract		char(6)	6		FALSE	FALSE
SelBlock		char(5)	5		FALSE	FALSE

Table syScenario

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
ScenarioId	EQ, FL, or HU	smallint			TRUE	FALSE
Description		varchar(100)	100		FALSE	FALSE
Type		char(2)	2		FALSE	FALSE

Table syScenarioEq

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqScenarioId	EQ, FL, or HU	smallint			TRUE	FALSE
ScenarioId		smallint			FALSE	TRUE
Type		char(2)	2		FALSE	FALSE
EventId	Sub-surface rupture length (kms)	char(8)	8		FALSE	FALSE
Magnitude		decimal(4,2)	4	2	FALSE	FALSE
SSRuptLength		real			FALSE	FALSE
SRuptLength	Surface rupture length (kms)	real			FALSE	FALSE
RuptOrientation		smallint			FALSE	FALSE
EpicenterLat		decimal(11,6)	11	6	FALSE	FALSE
EpicenterLongit		decimal(11,6)	11	6	FALSE	FALSE
EpicenterDepth		real			FALSE	FALSE
EpicenterWidth		real			FALSE	FALSE
EqScenarioType		tinyint			FALSE	FALSE
FaultType		tinyint			FALSE	FALSE
EventType		tinyint			FALSE	FALSE
ReturnPeriod		smallint			FALSE	FALSE
PgaMap		varchar(40)	40		FALSE	FALSE
PgvMap		varchar(40)	40		FALSE	FALSE
Spec03Map		varchar(40)	40		FALSE	FALSE
Spec10Map		varchar(40)	40		FALSE	FALSE
EqDuration		tinyint			FALSE	FALSE
DipAngle		smallint			FALSE	FALSE
ID_	Map object id (matches id in .dbf file)	char(8)	8		FALSE	FALSE

Table syScenarioFl

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
flScenarioId		smallint			TRUE	FALSE
ScenarioId		smallint			FALSE	TRUE
Reserved		smallint			FALSE	FALSE

Table syScenarioHu

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
huScenarioId		smallint			TRUE	FALSE
ScenarioId		smallint			FALSE	TRUE
Reserved		smallint			FALSE	FALSE

Table syState

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
StateFips	2-digit state FIPS code	char(2)	2		TRUE	FALSE
State	2-letter state name	char(2)	2		FALSE	FALSE
StateName	Object name	varchar(40)	40		FALSE	FALSE
Region	Region= 1 for West, = 2 for East	tinyint			FALSE	FALSE
BasemntRBlock	% of bldgs with basement in riverine census blocks	tinyint			FALSE	FALSE
BasemntCBlock	% of bldgs with basement in coastal census blocks	tinyint			FALSE	FALSE
Pct1StryRes1	% of 1-story structures for RES1 occupancy	tinyint			FALSE	FALSE
Pct2StryRes1	% of 2-story structures for RES1 occupancy	tinyint			FALSE	FALSE
Pct3StryRes1	% of 3-story structures for RES1 occupancy	tinyint			FALSE	FALSE
PctSplitLvRes1	% of split-level structures for RES1 occupancy	tinyint			FALSE	FALSE
Pct1to2StryRes3	% of 1 to 2-story structures for RES3 occupancy	tinyint			FALSE	FALSE
Pct3to4StryRes3	% of 3 to 4-story structures for RES3 occupancy	tinyint			FALSE	FALSE
Pct5StryplusRes3	% of 5+ story structures for RES3 occupancy	tinyint			FALSE	FALSE
PctLowRiseOther	% of low-rise structures for all other occupancies (non RES1 or RES3)	tinyint			FALSE	FALSE
PctMidRiseOther	% of mid-rise structures for all other occupancies (non RES1 or RES3)	tinyint			FALSE	FALSE
PctHighRiseOther	% of high-rise structures for all other occupancies (non RES1 or RES3)	tinyint			FALSE	FALSE
Pct1CarGarage	% of bldgs with 1-car garage	tinyint			FALSE	FALSE
Pct2CarGarage	% of bldgs with 2-car garage	tinyint			FALSE	FALSE
Pct3CarGarage	% of bldgs with 3-car garage	tinyint			FALSE	FALSE
PctCoverPort	% of bldgs with cover-ports	tinyint			FALSE	FALSE
PctNoGarage	% of bldgs with no garages	tinyint			FALSE	FALSE
IncomeRatio	Income ratio	real			FALSE	FALSE

Table syStudyRegion

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
RegionID	Study region Id	smallint			TRUE	FALSE
DatabaseName	Study region database name	varchar(50)	50		FALSE	FALSE
Description		varchar(100)	100		FALSE	FALSE
Created	Data/time region was created	datetime			FALSE	FALSE
LastAccess	Date/time region was last accessed	datetime			FALSE	FALSE
HasEqHazard	Flag= true of region has earthquake data	bit			FALSE	FALSE
HasFIHazard	Flag= true of region has flood data	bit			FALSE	FALSE
HasHuHazard	Flag= true of region has hurricane data	bit			FALSE	FALSE
Valid	Flag= true if region has been successfully aggregated and therefor is valid. = false otherwise.	bit			FALSE	FALSE
ID_	Map object id (matches id in .dbf file)	char(8)	8		FALSE	FALSE

Appendix B: The HAZUS-MH Template Database Schema

Table List

Name	Check Constraint Name	Primary Key
clBridges	CKT_CLBRIDGES	idtcBridgeClass
clEF	CKT_CLEF	idtcEf
clHplf	CKT_CLHPLF	idtcHplf
clInventory	CKT_CLINVENTORY	idtcInventory
clPipelines	CKT_CLPIPELINES	idtcPipelines
clSegments	CKT_CLSEGMENTS	idtcSegments
clTranspFacilities	CKT_CLTRANSPFACILITIES	idtcTranspFacilities
clTunnels	CKT_CLTUNNELS	idtcTunnels
clUtilFacilities	CKT_CLUTILFACILITIES	idtcUtilFacilities
eqAirportFlty	CKT_EQAIRPORTFLTY	idtKey
eqBldgCounBldgTypeB	CKT_EQBDLGCCOUNBLDGTYPB	idtKey
eqBldgCountBldgTypeT	CKT_EQBLDGCOUNTBLDGTYPET	idtKey
eqBridgeRestFuncts	CKT_EQBRIDGERESTFUNCTS	idtKey
eqBridgesDmgFuncts	CKT_EQBRIDGESDMGFUNCTS	idtKey
eqBridgesEconFuncts	CKT_EQBRIDGESECONFUNCTS	idtKey
eqBusFlty	CKT_EQBUSFLTY	idtKey
eqCapacityCurve	CKT_EQCAPACITYCURVE	idtKey
eqCareFlty	CKT_EQCAREFLTY	idtKey
eqCasAnalParms	CKT_EQCASANALPARMS	idtKey
eqCBldgTypeMp	CKT_EQCBLDGTYPEMP	idteqCBldgTypeMp
eqclBldgType	CKT_EQCLBLDGTYPB	idteqclBldgType
eqCommunicationFlty	CKT_EQCOMMUNICATIONFLTY	idtKey
eqDebrisAnalParms	CKT_EQDEBRISANALPARMS	idtKey
eqEfScheme	CKT_EQEFScheme	idteqEfScheme
eqEfSchemes	CKT_EQEFSchemes	idteqEfSchemes
eqElectricPowerFlty	CKT_EQELECTRICPOWERFLTY	idtKey
eqEmergencyCtr	CKT_EQEMERGENCYCTR	idtKey
eqExposureBldgTypeB	CKT_EQEXPOSUREBLDGTYPB	idtKey
eqExposureBldgTypeT	CKT_EQEXPOSUREBLDGTYPET	idtKey
eqFerryFlty	CKT_EQFERRYFLTY	idtKey
eqFireStation	CKT_EQFIRESTATION	idtKey
eqFragilityCurve	CKT_EQFRAGILITYCURVE	idtKey
eqGenBldgSchemes	CKT_EQGENBLDGSchemes	idteqBldgSchemes
eqGenSpecSchemes	CKT_EQGENSPCSchemes	idteqSpcBlgdSchemes
eqHazard	CKT_EQHAZMAT	idtKey
eqHBldgTypeMp	CKT_EQHBLDGTYPEMP	idteqHBldgTypeMp
eqHighwayBridge	CKT_EQHIGHWAYBRIDGE	idtKey
eqHighwaySegment	CKT_EQHIGHWAYSEGMENT	idtKey
eqHighwyTunnel	CKT_EQHIGHWYTUNNEL	idtKey
eqLighRailFlty	CKT_EQLIGHTRAILFLTY	idtKey
eqLighttailSegment	CKT_EQLIGHTTAILSEGMENT	idtKey
eqLightRailBridge	CKT_EQLIGHTRAILBRIDGE	idtKey
eqLightRailTunnel	CKT_EQLIGHTRAILTUNNEL	idtKey
eqMBldgTypeMp	CKT_EQMBLDGTYPEMP	idteqMBldgTypeMp
eqMilitary	CKT_EQMILITARY	idtKey
eqNaturalGasFlty	CKT_EQNATURALGASFLTY	idtKey
eqNaturalGasPl	CKT_EQNATURALGASPL	idtKey
eqNuclearFlty	CKT_EQNUCLEARFLTY	idtKey
eqOilFlty	CKT_EQOILFLTY	idtKey
eqOilPl	CKT_EQOILPL	idtKey
eqPoliceStation	CKT_EQPOLICESTATION	idtKey
eqPortFlty	CKT_EQPORTFLTY	idtKey
eqPotableWaterPl	CKT_EQPOTABLEWATERPL	idtKey
eqRailFlty	CKT_EQRAILFLTY	idtKey
eqRailwayBridge	CKT_EQRAILWAYBRIDGE	idtKey
eqRailwaySegment	CKT_EQRAILWAYSEGMENT	idtKey

Name	Check Constraint Name	Primary Key
eqRailwayTunnel	CKT_EQRAILWAYTUNNEL	idtKey
eqRunway	CKT_EQRUNWAY	idteqYfa
eqSBldgTypeMp	CKT_EQSBLDGTYPEMP	idteqSBldgTypeMp
eqSchool	CKT_EQSCHOOL	idtKey
eqSegAnalParms	CKT_EQSEGANALPARMS	idtKey
eqSpcBlgdSchemes	CKT_EQSPCBLGDSCHEMES	idteqSpcBlgdSchemes
eqSqFootageBldgTypeT	CKT_EQSQFOOTAGEBLDGTYPEPET	idtKey
eqTractAttribs	CKT_EQTRACTATTRIBS	idtKey
eqTunnelsDmgFuncnts	CKT_EQTUNNELSDMGFUNCTS	idteqTunnelsDmgFuncnts
eqTunnelsEconFuncnts	CKT_EQTUNNELSECONFUNCTS	idteqTunnelsEconFuncnts
eqTunnelsRestFuncnts	CKT_EQTUNNELSRESTFUNCTS	idteqTunnelsRestFuncnts
eqWasteWaterFlty	CKT_EQWASTEWATERFLTY	idtKey
eqWasteWaterPl	CKT_EQWASTEWATERPL	idtKey
eqWBldgTypeMp	CKT_EQWBLDGTYPEMP	idteqWBldgTypeMp
hzAirportFlty	CKT_HZAIRPORTFLTY	idthzAfa
hzBldgCountOccupB	CKT_HZBLDGCOUNTOCCUPB	idtKey
hzBldgCountOccupT	CKT_HZBLDGCOUNTOCCUPT	idtKey
hzBusFlty	CKT_HZBUSFLTY	idthzBfa
hzCareFlty	CKT_HZCAREFLTY	idthzCareFlty
hzCensusBlock	CKT_HZCENSUSBLOCK	idthzCensusBlock
hzCommunicationFlty	CKT_HZCOMMUNICATIONFLTY	idthzCfa
hzCounty	CKT_HZCOUNTY	idthzCounty
hzDams	CKT_HZDAMS	idthzDams
hzDemographicsB	CKT_HZDEMOGRAPHICSB	idtKey
hzDemographicsT	CKT_HZDEMOGRAPHICST	idtKey
hzElectricPowerFlty	CKT_HZELECTRICPOWERFLTY	idthzEfa
hzEmergencyCtr	CKT_HZEMERGENCYCTR	idthzEmergencyCtr
hzExposureOccupB	CKT_HZEXPOSUREOCCUPB	idtKey
hzExposureOccupT	CKT_HZEXPOSUREOCCUPT	idtKey
hzFerryFlty	CKT_HZFERRYFLTY	idthzFfa
hzFireStation	CKT_HZFIRESTATION	idthzFireStation
hzGenBldgScheme	CKT_HZGENBLDGSCHEME	idthzGenBldgScheme
hzHazmat	CKT_HZHAZMAT	idthzHazMat
hzHighwayBridge	CKT_HZHIGHWAYBRIDGE	idthzHbr
hzHighwaySegment	CKT_HZHIGHWAYSEGMENT	idthzHsg
hzHighwayTunnel	CKT_HZHIGHWAYTUNNEL	idthzhtu
hzLevees	CKT_HZLEVEES	idthzLevees
hzLightRailBridge	CKT_HZLIGHTRAILBRIDGE	idhztLbr
hzLightRailFlty	CKT_HZLIGHTRAILFLTY	idthzLfa
hzLightRailSegment	CKT_HZLIGHTRAILSEGMENT	idthzLsg
hzLightRailTunnel	CKT_HZLIGHTRAILTUNNEL	idthzltu
hzMilitary	CKT_HZMILITARY	idthzMilitary
hzNaturalGasFlty	CKT_HZNATURALGASFLTY	idthzNfa
hzNaturalGasPl	CKT_HZNATURALGASPL	idthznpl
hzNuclearFlty	CKT_HZNUCLEARFLTY	idthzNuclearFlty
hzOiFlty	CKT_HZOIFLTY	idthzOfa
hzOilPl	CKT_HZOILPL	idthzopl
hzPoliceStation	CKT_HZPOLICESTATION	idtPoliceStation
hzPortFlty	CKT_HZPORTFLTY	idthzTfa
hzPotableWaterPl	CKT_HZPOTABLEWATERPL	idthzppl
hzRailFlty	CKT_HZRAILFLTY	idthzRfa
hzRailwayBridge	CKT_HZRAILWAYBRIDGE	idthzRbr
hzRailwaySegment	CKT_HZRAILWAYSEGMENT	idthzRsg
hzRailwayTunnel	CKT_HZRAILWAYTUNNEL	idthzrtu
hzRunway	CKT_HZRUNWAY	idthzYfa
hzSchool	CKT_HZSCHOOL	idthzSchool
hzSqFootageOccupB	CKT_HZSQFOOTAGEOCCUPB	idtKey
hzSqFootageOccupT	CKT_HZSQFOOTAGEOCCUPT	idtKey
hzTract	CKT_HZTRACT	idthzTract
hzWasteWaterFlty	CKT_HZWASTEWATERFLTY	idthzWfa
hzWasteWaterPl	CKT_HZWASTEWATERPL	idthzwpl

Table clBridges

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
BridgeClass		char(5)	5		TRUE	FALSE
Category		varchar(10)	10		FALSE	FALSE

Table clBridges

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
BridgeClass		char(5)	5		TRUE	FALSE
Category		varchar(10)	10		FALSE	FALSE

Table clEF

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
EfClass		char(5)	5		TRUE	FALSE
GDescription	General description	varchar(25)	25		FALSE	FALSE
SDescription	Specific description	varchar(40)	40		FALSE	FALSE
DisplayOrder	Numeric sequence to control the display order of the diff. bldgy type classes	tinyint			FALSE	FALSE

Table clHPLF

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
HplfClass		char(5)	5		TRUE	FALSE
GDescription	General description	varchar(25)	25		FALSE	FALSE
SDescription	Specific description	varchar(40)	40		FALSE	FALSE
DisplayOrder	Numeric sequence to control the display order of the diff. bldgy type classes	tinyint			FALSE	FALSE

Table clInventory

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
CategoryId		smallint			TRUE	FALSE

Table clPipelines

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
PipelinesClass		char(5)	5		TRUE	FALSE
Category		varchar(10)	10		FALSE	FALSE

Table clSegments

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
SegmentClass		char(5)	5		TRUE	FALSE
Category		varchar(10)	10		FALSE	FALSE

Table clTranspFacilities

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
TranspFclyClass		char(5)	5		TRUE	FALSE
CategoryId		smallint			FALSE	TRUE
Category		varchar(10)	10		FALSE	FALSE

Table clUtilFacilities

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
UtilFclyClass		char(5)	5		TRUE	FALSE
CategoryId		smallint			FALSE	TRUE
Description	Description	varchar(40)	40		FALSE	FALSE

Table eqAirportFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
AirportId		char(8)	8		TRUE	TRUE
DefSoil	Default soil type	char(1)	1		FALSE	FALSE
DefWaterDepth	Default water depth (ft)	smallint			FALSE	FALSE
WaterDepthVal	Water depth value at the site	smallint			FALSE	FALSE
ValsUptodate	Are SoilTypeVal, LiquefVal, LndSlide & WaaterDepthVal filed up-to-date?	bit			FALSE	FALSE
MinDistance		real			FALSE	FALSE
Pga100	Probabilistic PGA value at 100-year return period	real			FALSE	FALSE
Pgv100	Probabilistic PGV value at 100-year return period	real			FALSE	FALSE
PgdSxxx		real			FALSE	FALSE
PgdLatSpread		real			FALSE	FALSE
PgdLndSlide		real			FALSE	FALSE
PgdSurfFaultRupt		real			FALSE	FALSE
ProbGrndDef		decimal(5,2)	5	2	FALSE	FALSE
ProbLiquef		decimal(5,2)	5	2	FALSE	FALSE
ProbLndSlide		decimal(5,2)	5	2	FALSE	FALSE
ProbSurfFaultRupt		decimal(5,2)	5	2	FALSE	FALSE
ScenarioDmnd	ID of scenario for demand fields (pga, pgv, pgdxx,...)	tinyint			FALSE	FALSE
DsNone	Probability for damage state NONE	decimal(3,2)	3	2	FALSE	FALSE
DsSlight	Probability for damage state SLIGHT	decimal(3,2)	3	2	FALSE	FALSE
DsModerate	Probability for damage state MODERATE	decimal(3,2)	3	2	FALSE	FALSE
DsExtensive	Probability for damage state EXTENSIVE	decimal(3,2)	3	2	FALSE	FALSE
DsComplete	Probability for damage state COMPLETE	decimal(3,2)	3	2	FALSE	FALSE

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
DsExSlight	Probability for damage state exceeding SLIGHT Probability for damage state exceeding MODERATE Probability for damage state exceeding EXTENSIVE ID of scenario for damage (ds) fields	decimal(3,2)	3	2	FALSE	FALSE
DsExModerate		decimal(3,2)	3	2	FALSE	FALSE
DsExExtensive		decimal(3,2)	3	2	FALSE	FALSE
ScenarioDmg		tinyint			FALSE	FALSE
FunctDay0	ID of scenario for functionality fields	decimal(4,1)	4	1	FALSE	FALSE
FunctDay1		decimal(4,1)	4	1	FALSE	FALSE
FunctDay3		decimal(4,1)	4	1	FALSE	FALSE
FunctDay7		decimal(4,1)	4	1	FALSE	FALSE
FunctDay30	ID of scenario for functionality fields	decimal(4,1)	4	1	FALSE	FALSE
FunctDay90		decimal(4,1)	4	1	FALSE	FALSE
ScenarioFunct		tinyint			FALSE	FALSE
EconLoss		money			FALSE	FALSE
ScenarioLoss	Minimum distance to epicenter	tinyint			FALSE	FALSE

Table eqBldgCounBldgTypeB

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
CensusBlock	Full census block number	char(16)	16		TRUE	TRUE
Woodl		smallint			FALSE	FALSE
Steell		smallint			FALSE	FALSE
Concretel		smallint			FALSE	FALSE
Masonryl		smallint			FALSE	FALSE
ManufHousingl		smallint			FALSE	FALSE
W1l		smallint			FALSE	FALSE
W2l		smallint			FALSE	FALSE
S1LI		smallint			FALSE	FALSE
S1MI		smallint			FALSE	FALSE
S1HI		smallint			FALSE	FALSE
S2LI		smallint			FALSE	FALSE
S2MI		smallint			FALSE	FALSE
S2HI		smallint			FALSE	FALSE
S3l		smallint			FALSE	FALSE
S4LI		smallint			FALSE	FALSE
S4MI		smallint			FALSE	FALSE
S4HI		smallint			FALSE	FALSE
S5LI		smallint			FALSE	FALSE
S5MI		smallint			FALSE	FALSE
S5HI		smallint			FALSE	FALSE
C1LI		smallint			FALSE	FALSE
C1MI		smallint			FALSE	FALSE
C1HI		smallint			FALSE	FALSE
C2LI		smallint			FALSE	FALSE
C2MI		smallint			FALSE	FALSE
C2HI		smallint			FALSE	FALSE
C3LI		smallint			FALSE	FALSE
C3MI		smallint			FALSE	FALSE
C3HI		smallint			FALSE	FALSE

PC1I		smallint			FALSE	FALSE
PC2LI		smallint			FALSE	FALSE
PC2MI		smallint			FALSE	FALSE
PC2HI		smallint			FALSE	FALSE
RM1LI		smallint			FALSE	FALSE
RM1MI		smallint			FALSE	FALSE
RM2LI		smallint			FALSE	FALSE
RM2MI		smallint			FALSE	FALSE
RM2HI		smallint			FALSE	FALSE
URMLI		smallint			FALSE	FALSE
URMMI		smallint			FALSE	FALSE
MHI		smallint			FALSE	FALSE

Table eqBldgConutBldgTypeT

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
Tract		char(11)	11		TRUE	TRUE
WoodI		smallint			FALSE	FALSE
SteelI		smallint			FALSE	FALSE
Concretel		smallint			FALSE	FALSE
MasonryI		smallint			FALSE	FALSE
ManufHousingI		smallint			FALSE	FALSE
W1I		smallint			FALSE	FALSE
W2I		smallint			FALSE	FALSE
S1LI		smallint			FALSE	FALSE
S1MI		smallint			FALSE	FALSE
S1HI		smallint			FALSE	FALSE
S2LI		smallint			FALSE	FALSE
S2MI		smallint			FALSE	FALSE
S2HI		smallint			FALSE	FALSE
S3I		smallint			FALSE	FALSE
S4LI		smallint			FALSE	FALSE
S4MI		smallint			FALSE	FALSE
S4HI		smallint			FALSE	FALSE
S5LI		smallint			FALSE	FALSE
S5MI		smallint			FALSE	FALSE
S5HI		smallint			FALSE	FALSE
C1LI		smallint			FALSE	FALSE
C1MI		smallint			FALSE	FALSE
C1HI		smallint			FALSE	FALSE
C2LI		smallint			FALSE	FALSE
C2MI		smallint			FALSE	FALSE
C2HI		smallint			FALSE	FALSE
C3LI		smallint			FALSE	FALSE
C3MI		smallint			FALSE	FALSE
C3HI		smallint			FALSE	FALSE
PC1I		smallint			FALSE	FALSE
PC2LI		smallint			FALSE	FALSE
PC2MI		smallint			FALSE	FALSE
PC2HI		smallint			FALSE	FALSE
RM1LI		smallint			FALSE	FALSE
RM1MI		smallint			FALSE	FALSE
RM2LI		smallint			FALSE	FALSE
RM2MI		smallint			FALSE	FALSE
RM2HI		smallint			FALSE	FALSE
URMLI		smallint			FALSE	FALSE
URMMI		smallint			FALSE	FALSE
MHI		smallint			FALSE	FALSE

Table eqBridgeRestFuncnts

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
BridgeClass		char(5)	5		TRUE	TRUE

Table eqBridgesDmgFuncnts

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
BridgeClass		char(5)	5		TRUE	TRUE

Table eqBridgesEconFuncnts

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
BridgeClass		char(5)	5		TRUE	TRUE

Table eqBusFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
BusFltyId		char(8)	8		TRUE	TRUE

Table eqCBldgTypeMp

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqCBldgTypeMpld	Essential facilities mapping scheme id	int			TRUE	FALSE
eqSpcBlgdSchemesId		int			FALSE	TRUE
BldgQuality	C, I, S (Code, Inferior, Superior)	char(1)	1		FALSE	FALSE
DesignLevel		char(1)	1		FALSE	FALSE
C1LP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
C1MP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
C1HP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
C2LP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
C2MP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
C2HP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
C3LP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
C3MP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
C3HP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
PC1P	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
PC2LP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
PC2MP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
PC2HP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE

Table eqCapacityCurve

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqBldgType	Specific bldg class	char(4)	4		TRUE	TRUE

Table eqCasAnalParms

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqBldgType	Specific bldg class	char(4)	4		TRUE	TRUE

Table eqCommunicationFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
CommunicationFltyId		char(8)	8		TRUE	TRUE

Table eqDebrisAnalParms

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqBldgType	Specific bldg class	char(4)	4		TRUE	TRUE

Table eqEfScheme

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqEfSchemeld	Essential facilities mapping scheme id	int			TRUE	FALSE
eqEfSchemesId		smallint			FALSE	TRUE
Occupancy	Specific occupancy class	char(5)	5		FALSE	FALSE
WPct	Pct of wood distribution	tinyint			FALSE	FALSE
CPct	Pct of concrete distribution	tinyint			FALSE	FALSE
SPct	Pct of steel distribution	tinyint			FALSE	FALSE
MPct	Pct of masonry distribution	tinyint			FALSE	FALSE
MhPct	Pct of mobile homes distribution	tinyint			FALSE	FALSE

Table eqEfSchemes

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqEfSchemesId	Description	smallint	20		TRUE	FALSE
SchemeName		varchar(20)	40		FALSE	FALSE
Description	Type to which scheme applies: M for military, P for police, F for fire stations, and E for eoc's	varchar(40)			FALSE	FALSE
Created		datetime			FALSE	FALSE
Updated		datetime			FALSE	FALSE
EfSchemeType		char(1)	1		FALSE	FALSE
					FALSE	FALSE

Table eq ElectricPowerFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
ElectricPowerFltyId		char(8)	8		TRUE	TRUE

Table eqEmergencyCtr

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
EocId		char(8)	8		TRUE	TRUE
eqEfSchemesId		smallint			FALSE	TRUE
DesignLevel		char(1)	1		FALSE	FALSE
Bias		char(1)	1		FALSE	FALSE
BldgType		char(4)	4		FALSE	FALSE

Table eqExposureBldgTypeB

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
CensusBlock	Full census block number	char(16)	16		TRUE	TRUE
WoodI		smallint			FALSE	FALSE
SteelI		smallint			FALSE	FALSE
Concretel		smallint			FALSE	FALSE
MasonryI		smallint			FALSE	FALSE
ManufHousingI		smallint			FALSE	FALSE
W1I		smallint			FALSE	FALSE
W2I		smallint			FALSE	FALSE
S1LI		smallint			FALSE	FALSE
S1MI		smallint			FALSE	FALSE
S1HI		smallint			FALSE	FALSE
S2LI		smallint			FALSE	FALSE
S2MI		smallint			FALSE	FALSE
S2HI		smallint			FALSE	FALSE
S3I		smallint			FALSE	FALSE
S4LI		smallint			FALSE	FALSE
S4MI		smallint			FALSE	FALSE
S4HI		smallint			FALSE	FALSE
S5LI		smallint			FALSE	FALSE
S5MI		smallint			FALSE	FALSE
S5HI		smallint			FALSE	FALSE
C1LI		smallint			FALSE	FALSE
C1MI		smallint			FALSE	FALSE
C1HI		smallint			FALSE	FALSE
C2LI		smallint			FALSE	FALSE

C2MI		smallint			FALSE	FALSE
C2HI		smallint			FALSE	FALSE
C3LI		smallint			FALSE	FALSE
C3MI		smallint			FALSE	FALSE
C3HI		smallint			FALSE	FALSE
PC1I		smallint			FALSE	FALSE
PC2LI		smallint			FALSE	FALSE
PC2MI		smallint			FALSE	FALSE
PC2HI		smallint			FALSE	FALSE
RM1LI		smallint			FALSE	FALSE
RM1MI		smallint			FALSE	FALSE
RM2LI		smallint			FALSE	FALSE
RM2MI		smallint			FALSE	FALSE
RM2HI		smallint			FALSE	FALSE
URMLI		smallint			FALSE	FALSE
URMMI		smallint			FALSE	FALSE
MHI		smallint			FALSE	FALSE

Table eqExposureBldgTypeT

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
Tract		char(11)	11		TRUE	TRUE
WoodI		smallint			FALSE	FALSE
SteelI		smallint			FALSE	FALSE
Concretel		smallint			FALSE	FALSE
MasonryI		smallint			FALSE	FALSE
ManufHousingI		smallint			FALSE	FALSE
W1I		smallint			FALSE	FALSE
W2I		smallint			FALSE	FALSE
S1LI		smallint			FALSE	FALSE
S1MI		smallint			FALSE	FALSE
S1HI		smallint			FALSE	FALSE
S2LI		smallint			FALSE	FALSE
S2MI		smallint			FALSE	FALSE
S2HI		smallint			FALSE	FALSE
S3I		smallint			FALSE	FALSE
S4LI		smallint			FALSE	FALSE
S4MI		smallint			FALSE	FALSE
S4HI		smallint			FALSE	FALSE
S5LI		smallint			FALSE	FALSE
S5MI		smallint			FALSE	FALSE
S5HI		smallint			FALSE	FALSE
C1LI		smallint			FALSE	FALSE
C1MI		smallint			FALSE	FALSE
C1HI		smallint			FALSE	FALSE
C2LI		smallint			FALSE	FALSE
C2MI		smallint			FALSE	FALSE
C2HI		smallint			FALSE	FALSE
C3LI		smallint			FALSE	FALSE
C3MI		smallint			FALSE	FALSE
C3HI		smallint			FALSE	FALSE
PC1I		smallint			FALSE	FALSE
PC2LI		smallint			FALSE	FALSE
PC2MI		smallint			FALSE	FALSE
PC2HI		smallint			FALSE	FALSE
RM1LI		smallint			FALSE	FALSE
RM1MI		smallint			FALSE	FALSE
RM2LI		smallint			FALSE	FALSE
RM2MI		smallint			FALSE	FALSE
RM2HI		smallint			FALSE	FALSE
URMLI		smallint			FALSE	FALSE

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
URMMI		smallint			FALSE	FALSE
MHI		smallint			FALSE	FALSE

Table eqFerryFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
FerryFltyId		char(8)	8		TRUE	TRUE

Table eqFireStation

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqEfSchemesId		smallint			TRUE	TRUE
FireStationId		char(8)	8		FALSE	TRUE
DesignLevel		char(1)	1		FALSE	FALSE
Bias		char(1)	1		FALSE	FALSE
BldgType		char(4)	4		FALSE	FALSE

Table eqFragilityCurve

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqBldgType	Specific bldg class	char(4)	4		TRUE	TRUE

Table eqGenBldgSchemes

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqBldgSchemesId		int			TRUE	FALSE
SchemeName		varchar(20)	20		FALSE	FALSE
Description	Description	varchar(40)	40		FALSE	FALSE
Created		datetime			FALSE	FALSE
Updated		datetime			FALSE	FALSE

Table eqHBldgTypeMp

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqHBldgTypeMpId	Essential facilities mapping scheme id	int			TRUE	FALSE
eqSpcBldgSchemesId		int			FALSE	TRUE
BldgQuality	C, I, S (Code, Inferior, Superior)	char(1)	1		FALSE	FALSE
DesignLevel		char(1)	1		FALSE	FALSE
MHP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE

Table eqHazmat

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
HazmatID		char(8)	8		TRUE	TRUE
Bias		char(1)	1		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
DesignLevel		char(1)	1		FALSE	FALSE
BldgType		char(4)	4		FALSE	FALSE

Table eqHighwayBridge

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
HighwayBridgeId		char(8)	8		TRUE	TRUE

Table eqHighwaySegment

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
HighwaySegId		char(8)	8		TRUE	TRUE

Table eqHighwayTunnel

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
HighwayTunnelId		char(8)	8		TRUE	TRUE

Table eqLightRailBridge

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
LightRailBridgeId		char(8)	8		TRUE	TRUE

Table eqLight RailTunnel

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
LightRailTunnelId		char(8)	8		TRUE	TRUE

Table eqLightRailSegment

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
LightRailSegId		char(8)	8		TRUE	TRUE

Table eqMBldgTypeMp

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqMBldgTypeMpld	Essential facilities mapping scheme id	int			TRUE	FALSE
eqSpcBldgSchemesId		int			FALSE	TRUE
BldgQuality	C, I, S (Code, Inferior, Superior)	char(1)	1		FALSE	FALSE
DesignLevel		char(1)	1		FALSE	FALSE
RM1LP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
RM1MP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
RM2LP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
RM2MP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
RM2HP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
URMLP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE
URMMP	% distribution for given bldg type/occupa	tinyint			FALSE	FALSE

Table eqMilitary

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqEfSchemesId		smallint			TRUE	TRUE
MilitaryFtyId		char(8)	8		FALSE	TRUE
DesignLevel		char(1)	1		FALSE	FALSE
Bias		char(1)	1		FALSE	FALSE
BldgType		char(4)	4		FALSE	FALSE

Table eqNaturalGasFty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
NaturalGasFtyId		char(8)	8		TRUE	TRUE

Table eqNaturalGasPl

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
NaturalGasPlId		char(8)	8		TRUE	TRUE

Table eqNuclearFty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
NuclearFtyId		char(8)	8		TRUE	TRUE
DesignLevel		char(1)	1		FALSE	FALSE
Bias		char(1)	1		FALSE	FALSE

Table eqOilFty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
OilFtyId		char(8)	8		TRUE	TRUE

Table eqOilPl

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
OilPlId		char(8)	8		TRUE	TRUE

Table eqPoliceStation

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqEfSchemesId		smallint			TRUE	TRUE
PoliceStationId		char(8)	8		FALSE	TRUE
DesignLevel		char(1)	1		FALSE	FALSE
Bias		char(1)	1		FALSE	FALSE
BldgType		char(4)	4		FALSE	FALSE

Table eqPortFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
PortFltyId		char(8)	8		TRUE	TRUE

Table eqPotableWaterPl

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
PotableWaterPlId		char(8)	8		TRUE	TRUE
GClass		char(1)	1		FALSE	FALSE
Status		decimal(1,0)	1		FALSE	FALSE
UpNode		varchar(5)	5		FALSE	FALSE
DownNode		varchar(5)	5		FALSE	FALSE
Roughness		decimal(5,2)	5	2	FALSE	FALSE
MinLoss		decimal(3,2)	3	2	FALSE	FALSE

Table eqRailFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
RailFltyId		char(8)	8		TRUE	TRUE
eqRailFltyId		int			FALSE	FALSE

Table eqRailwayBridge

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
RailwayBridgeId		char(8)	8		TRUE	TRUE

Table eqRailwaySegment

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
RailwaySegId		char(8)	8		TRUE	TRUE

Table eqRailwayTunnel

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
RailwayTunnelId		char(8)	8		TRUE	TRUE

Table eqRunway

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqRunwayId		int			TRUE	FALSE
RunwayId		smallint			FALSE	TRUE

Table eqSBldgTypeMp

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqSBldgTypeMpId		int			TRUE	FALSE
eqSpcBldgSchemesId		int			FALSE	TRUE
DesignLevel		char(1)	1		FALSE	FALSE
BldgQuality	C, I, S (Code, Inferior, Superior)	char(1)	1		FALSE	FALSE
S1LP	% distribution for given bldg	tinyint			FALSE	FALSE

S1MP	type/occupancy % distribution for given bldg	tinyint			FALSE	FALSE
S1HP	type/occupancy % distribution for given bldg	tinyint			FALSE	FALSE
S2LP	type/occupa % distribution for given bldg	tinyint			FALSE	FALSE
S2MP	type/occupa % distribution for given bldg	tinyint			FALSE	FALSE
S2HP	type/occupa % distribution for given bldg	tinyint			FALSE	FALSE
S3P	type/occupa % distribution for given bldg	tinyint			FALSE	FALSE
S4LP	type/occupa % distribution for given bldg	tinyint			FALSE	FALSE
S4MP	type/occupa % distribution for given bldg	tinyint			FALSE	FALSE
S4HP	type/occupa % distribution for given bldg	tinyint			FALSE	FALSE
S5LP	type/occupa % distribution for given bldg	tinyint			FALSE	FALSE
S5MP	type/occupa % distribution for given bldg	tinyint			FALSE	FALSE
S5HP	type/occupa % distribution for given bldg	tinyint			FALSE	FALSE

Table eqSchool

Name	Comment	Data Type	Length	Preci sion	Primary	Foreign Key
SchoolId		char(8)	8		TRUE	TRUE
eqEfSchemesId		smallint			FALSE	TRUE
ShelterCapacity		smallint			FALSE	FALSE
District		varchar(30)	30		FALSE	FALSE
NumStudents		smallint			FALSE	FALSE
Area	Area (sq.ft) for given occupancy	real			FALSE	FALSE
Kitchen		bit			FALSE	FALSE
BldgType		char(4)	4		FALSE	FALSE

Table eqSegAnalParms

Name	Comment	Data Type	Length	Preci sion	Primary	Foreign Key
SegmentClass		char(5)	5		TRUE	TRUE

Table eqSpcBlgdSchemes

Name	Comment	Data Type	Length	Preci sion	Primary	Foreign Key
eqSpcBlgdSchemesId		int			TRUE	FALSE
eqEfSchemeld	Essential facilities mapping scheme id	int			FALSE	TRUE
SchemeName		varchar(20)	20		FALSE	FALSE
Description	Description	varchar(40)	40		FALSE	FALSE
Created		datetime			FALSE	FALSE
Updated		datetime			FALSE	FALSE
TypeMapping	Specifies what general bldg type this mapping refers to: W, C, S, M, MH	char(1)	1		FALSE	FALSE

Table eqSqFootageBldgTypeT

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
Tract		char(11)	11		TRUE	TRUE
WoodF		real			FALSE	FALSE
SteelF		real			FALSE	FALSE
ConcreteF		real			FALSE	FALSE
MasonryF		real			FALSE	FALSE
ManufHousingF		real			FALSE	FALSE
W1F		real			FALSE	FALSE
W2F		real			FALSE	FALSE
S1LF		real			FALSE	FALSE
S1MF		real			FALSE	FALSE
S1HF		real			FALSE	FALSE
S2LF		real			FALSE	FALSE
S2MF		real			FALSE	FALSE
S2HF		real			FALSE	FALSE
S3F		real			FALSE	FALSE
S4LF		real			FALSE	FALSE
S4MF		real			FALSE	FALSE
S4HF		real			FALSE	FALSE
S5LF		real			FALSE	FALSE
S5MF		real			FALSE	FALSE
S5HF		real			FALSE	FALSE
C1LF		real			FALSE	FALSE
C1MF		real			FALSE	FALSE
C1HF		real			FALSE	FALSE
C2LF		real			FALSE	FALSE
C2MF		real			FALSE	FALSE
C2HF		real			FALSE	FALSE
C3LF		real			FALSE	FALSE
C3MF		real			FALSE	FALSE
C3HF		real			FALSE	FALSE
PC1F		real			FALSE	FALSE
PC2LF		real			FALSE	FALSE
PC2MF		real			FALSE	FALSE
PC2HF		real			FALSE	FALSE
RM1LF		real			FALSE	FALSE
RM1MF		real			FALSE	FALSE
RM2LF		real			FALSE	FALSE
RM2MF		real			FALSE	FALSE
RM2HF		real			FALSE	FALSE
URMLF		real			FALSE	FALSE
URMMF		real			FALSE	FALSE
MHF		real			FALSE	FALSE

Table eqTractAttribs

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
Tract		char(11)	11		TRUE	TRUE
DefSoil	Default soil type	char(1)	1		FALSE	FALSE
DefLiquef	Default liquefaction susceptibility category	char(1)	1		FALSE	FALSE
DefLandslide	Default landslide susceptibility category	char(2)	2		FALSE	FALSE
DefWaterDepth	Default water depth (ft)	smallint			FALSE	FALSE
Pga100	Probabilistic PGA value at 100-year return period	real			FALSE	FALSE
Pga250	Probabilistic PGA value at 250-	real			FALSE	FALSE

Pga500	year return period Probabilistic PGA value at 500-	real			FALSE	FALSE
Pga750	year return period Probabilistic PGA value at	real			FALSE	FALSE
Pga1000	750year return period Probabilistic PGA value at 1000-	real			FALSE	FALSE
Pga1500	year return period Probabilistic PGA value at 1500-	real			FALSE	FALSE
Pga2000	year return period Probabilistic PGA value at 2000-	real			FALSE	FALSE
Pga2500	year return period Probabilistic PGA value at 2500-	real			FALSE	FALSE
Pgv100	year return period Probabilistic PGV value at 100-	real			FALSE	FALSE
Pgv250	year return period Probabilistic PGV value at 250-	real			FALSE	FALSE
Pgv500	year return period Probabilistic PGV value at 500-	real			FALSE	FALSE
Pgv750	year return period Probabilistic PGV value at 750-	real			FALSE	FALSE
Pgv1000	year return period Probabilistic PGV value at 1000-	real			FALSE	FALSE
Pgv1500	year return period Probabilistic PGV value at 1500-	real			FALSE	FALSE
Pgv2000	year return period Probabilistic PGV value at 2000-	real			FALSE	FALSE
Pgv2500	year return period Probabilistic PGV value at 250-	real			FALSE	FALSE
Sa03100	year return period Probabilistic Sa at 0.3 secs for	real			FALSE	FALSE
Sa03250	100-year return period Probabilistic Sa at 0.3 secs for	real			FALSE	FALSE
Sa03500	250-year return period Probabilistic Sa at 0.3 secs for	real			FALSE	FALSE
Sa03750	500-year return period Probabilistic Sa at 0.3 secs for	real			FALSE	FALSE
Sa031000	750-year return period Probabilistic Sa at 0.3 secs for	real			FALSE	FALSE
Sa031500	1000-year return period Probabilistic Sa at 0.3 secs for	real			FALSE	FALSE
Sa032000	1500-year return period Probabilistic Sa at 0.3 secs for	real			FALSE	FALSE
Sa032500	2000-year return period Probabilistic Sa at 0.3 secs for	real			FALSE	FALSE
Sa10100	2500-year return period Probabilistic Sa at 1.0 secs for	real			FALSE	FALSE
Sa10250	100-year return period Probabilistic Sa at 1.0 secs for	real			FALSE	FALSE
Sa10500	250-year return period Probabilistic Sa at 1.0 secs for	real			FALSE	FALSE
Sa10750	500-year return period Probabilistic Sa at 1.0 secs for	real			FALSE	FALSE
Sa101000	750-year return period Probabilistic Sa at 1.0 secs for	real			FALSE	FALSE
Sa101500	1000-year return period Probabilistic Sa at 1.0 secs for	real			FALSE	FALSE
Sa102000	1500-year return period Probabilistic Sa at 1.0 secs for	real			FALSE	FALSE
Sa102500	2000-year return period Probabilistic Sa at 1.0 secs for	real			FALSE	FALSE
	2500-year return period					

Table eqTunnelsDmgFuncnts

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqTunnelsDmgFuncntsId		smallint			TRUE	FALSE
TunnelClass		char(5)	5		FALSE	TRUE

Table eqTunnelsEconFuncnts

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqTunnelsEconFuncntsId		smallint			TRUE	FALSE
TunnelClass		char(5)	5		FALSE	TRUE

Table eqTunnelsRestFuncnts

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqTunnelsRestFuncntsId		smallint			TRUE	FALSE
TunnelClass		char(5)	5		FALSE	TRUE

Table eqWBdlgTypeMp

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqWBdlgTypeMpId		int			TRUE	FALSE
eqSpcBldgSchemesId		int			FALSE	TRUE
BldgQuality	C, I, S (Code, Inferior, Superior)	char(1)	1		FALSE	FALSE
DesignLevel		char(1)	1		FALSE	FALSE
W1P	% distribution for given bldg type/occupancy	tinyint			FALSE	FALSE
W2P	% distribution for given bldg type/occupancy	tinyint			FALSE	FALSE

Table eqWasteWaterFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
WasteWaterFltyId		char(8)	8		TRUE	TRUE

Table eqWasteWaterPl

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
WasteWaterPlId		char(8)	8		TRUE	TRUE
DmgScenario	Scenario ID for damage results	smallint			FALSE	FALSE
FunctScenario	Scenario ID for functionality results	smallint			FALSE	FALSE
EconScenario	Scenario ID for economic loss results	smallint			FALSE	FALSE

Table eqclBldgType

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
eqBldgType	Specific bldg class	char(4)	4		TRUE	FALSE
GBldgType	Correspondg general bldg class	char(2)	2		FALSE	FALSE
Description	Description	varchar(40)	40		FALSE	FALSE
Example	Example	varchar(40)	40		FALSE	FALSE
DisplayOrder	Numeric sequence to control the display order of the diff. bldgy	tinyint			FALSE	FALSE

	type classes					
--	--------------	--	--	--	--	--

Table hzAirportFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
AirportId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
TranspFclyClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
Use		varchar(10)	10		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
Anchor		bit			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
BldgType		char(4)	4		FALSE	FALSE
Cargo		int			FALSE	FALSE
NumFlights		smallint			FALSE	FALSE
NumPassengers		smallint			FALSE	FALSE
BackupPower		bit			FALSE	FALSE

Table hzBldgCountOccupB

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
CensusBlock	Full census block number	char(16)	16		TRUE	TRUE
RES1		smallint			FALSE	FALSE
COM1		smallint			FALSE	FALSE
IND1		smallint			FALSE	FALSE
AGRI		smallint			FALSE	FALSE
RELI		smallint			FALSE	FALSE
GOV1		smallint			FALSE	FALSE
EDUI		smallint			FALSE	FALSE
RES1I		smallint			FALSE	FALSE
RES2I		smallint			FALSE	FALSE
RES3AI		smallint			FALSE	FALSE
RES3BI		smallint			FALSE	FALSE
RES3CI		smallint			FALSE	FALSE
RES3DI		smallint			FALSE	FALSE
RES3EI		smallint			FALSE	FALSE
RES3FI		smallint			FALSE	FALSE
RES4I		smallint			FALSE	FALSE
RES5I		smallint			FALSE	FALSE
RES6I		smallint			FALSE	FALSE
COM1I		smallint			FALSE	FALSE
COM2I		smallint			FALSE	FALSE
COM3I		smallint			FALSE	FALSE
COM4I		smallint			FALSE	FALSE
COM5I		smallint			FALSE	FALSE
COM6I		smallint			FALSE	FALSE
COM7I		smallint			FALSE	FALSE
COM8I		smallint			FALSE	FALSE

COM9I		smallint			FALSE	FALSE
COM10I		smallint			FALSE	FALSE
IND1I		smallint			FALSE	FALSE
IND2I		smallint			FALSE	FALSE
IND3I		smallint			FALSE	FALSE
IND4I		smallint			FALSE	FALSE
IND5I		smallint			FALSE	FALSE
IND6I		smallint			FALSE	FALSE
AGR1I		smallint			FALSE	FALSE
REL1I		smallint			FALSE	FALSE
GOV1I		smallint			FALSE	FALSE
GOV2I		smallint			FALSE	FALSE
EDU1I		smallint			FALSE	FALSE
EDU2I		smallint			FALSE	FALSE

Table hzBldgCountOccupT

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
Tract		char(11)	11		TRUE	TRUE
RESI		smallint			FALSE	FALSE
COMI		smallint			FALSE	FALSE
INDI		smallint			FALSE	FALSE
AGRI		smallint			FALSE	FALSE
RELI		smallint			FALSE	FALSE
GOVI		smallint			FALSE	FALSE
EDUI		smallint			FALSE	FALSE
RES1I		smallint			FALSE	FALSE
RES2I		smallint			FALSE	FALSE
RES3AI		smallint			FALSE	FALSE
RES3BI		smallint			FALSE	FALSE
RES3CI		smallint			FALSE	FALSE
RES3DI		smallint			FALSE	FALSE
RES3EI		smallint			FALSE	FALSE
RES3FI		smallint			FALSE	FALSE
RES4I		smallint			FALSE	FALSE
RES5I		smallint			FALSE	FALSE
RES6I		smallint			FALSE	FALSE
COM1I		smallint			FALSE	FALSE
COM2I		smallint			FALSE	FALSE
COM3I		smallint			FALSE	FALSE
COM4I		smallint			FALSE	FALSE
COM5I		smallint			FALSE	FALSE
COM6I		smallint			FALSE	FALSE
COM7I		smallint			FALSE	FALSE
COM8I		smallint			FALSE	FALSE
COM9I		smallint			FALSE	FALSE
COM10I		smallint			FALSE	FALSE
IND1I		smallint			FALSE	FALSE
IND2I		smallint			FALSE	FALSE
IND3I		smallint			FALSE	FALSE
IND4I		smallint			FALSE	FALSE
IND5I		smallint			FALSE	FALSE
IND6I		smallint			FALSE	FALSE
AGR1I		smallint			FALSE	FALSE
REL1I		smallint			FALSE	FALSE
GOV1I		smallint			FALSE	FALSE
GOV2I		smallint			FALSE	FALSE
EDU1I		smallint			FALSE	FALSE
EDU2I		smallint			FALSE	FALSE

Table hzBusFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
BusFltyId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
TranspFcltyClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
Use		varchar(10)	10		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
Anchor		bit			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
BldgType		char(4)	4		FALSE	FALSE
BackupPower		bit			FALSE	FALSE
Traffic		int			FALSE	FALSE

Table hzCareFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
CareFltyId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
EfClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Elevation		numeric(6,1)	6	1	FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
BackupPower		bit			FALSE	FALSE
NumBeds		int			FALSE	FALSE
Ahald		char(7)	7		FALSE	FALSE
Use		varchar(10)	10		FALSE	FALSE

Table hzCensusBlock

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
CensusBlock	Full census block number	char(16)	16		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
CensusBlock5	6-digit census block number	varchar(5)	5		FALSE	FALSE
BlockType	R= Riverine, C= Coastal, N= NA	char(1)	1		FALSE	FALSE
BlockArea	Census block area (sq. km)	real			FALSE	FALSE
CenLat	Centroid latitude	decimal(11,6)	11	6	FALSE	FALSE
CenLongit	Centroid longitude	decimal(11,6)	11	6	FALSE	FALSE
PctWithBasemnt	% of structures with basement	tinyint			FALSE	FALSE

Pct1StoryRes1	% of Res1 1-story bldgs	tinyint			FALSE	FALSE
Pct2StoryRes1	% of Res1 2-story bldgs	tinyint			FALSE	FALSE
Pct3StoryRes1	% of Res1 3-story bldgs	tinyint			FALSE	FALSE
PctSplitLvIRes1	% of Res1 split-level bldgs	tinyint			FALSE	FALSE
Pct1to2StryRes3	% of Res3 1- to 2-story bldgs	tinyint			FALSE	FALSE
Pct3to4StryRes3	% of Res3 3- to 4-story bldgs	tinyint			FALSE	FALSE
Pct5StryplusRes3	% of Res3 5+ stories bldgs	tinyint			FALSE	FALSE
PctLowRiseOther	% of other (non-res1 or non-res3) low-rise structures	tinyint			FALSE	FALSE
PctMidRiseOther	% of other (non-res1 or non-res3) mid-rise structures	tinyint			FALSE	FALSE
PctHighRiseOther	% of other (non-res1 or non-res3) high-rise structures	tinyint			FALSE	FALSE
Pct1CarGarage	% of bldgs with 1-car garages	tinyint			FALSE	FALSE
Pct2CarGarage	% of bldgs with 2-car garages	tinyint			FALSE	FALSE
Pct3CarGarage	% of bldgs with 3-car garages	tinyint			FALSE	FALSE
PctCoverPort	% of bldgs with covred ports	tinyint			FALSE	FALSE
PctNoGarage	% of bldgs with no garages	tinyint			FALSE	FALSE
IncomeRatio	Income ratio	real			FALSE	FALSE

Table hzCommunicationFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
CommunicationFltyId		char(8)	8		TRUE	FALSE
UtilFclyClass		char(5)	5		FALSE	TRUE
Tract		char(11)	11		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
Use		varchar(10)	10		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
Anchor		bit			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
BackupPower		bit			FALSE	FALSE

Table hzCounty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
CountyFips	Full county FIPS code	char(5)	5		TRUE	FALSE
CountyFips3	3-digit county FIPS Code	char(3)	3		FALSE	FALSE
CountyName		varchar(40)	40		FALSE	FALSE
State	2-char state county	char(2)	2		FALSE	FALSE
StateFips		char(2)	2		FALSE	FALSE
NumAggrTracts	# of tracts aggregated within the county	int			FALSE	FALSE

Table hzDams

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
DamId		char(8)	8		TRUE	FALSE

CountyFips	Full county FIPS code	char(5)	5		FALSE	TRUE
------------	-----------------------	---------	---	--	-------	------

Table hzDemographicsB

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
CensusBlock	Full census block number	char(16)	16		TRUE	TRUE
Population	Total population	int			FALSE	FALSE
Households	Number of households	int			FALSE	FALSE
Quarters	Number group quarters	int			FALSE	FALSE
Less16	Population age less than 16 years	int			FALSE	FALSE
Pop16to65	Population age 16 to 65 years	int			FALSE	FALSE
Over65	Population age over 65 years	int			FALSE	FALSE
White	White population	int			FALSE	FALSE
Black	Black population	int			FALSE	FALSE
Native	Native American population	int			FALSE	FALSE
Asian	Asian population	int			FALSE	FALSE
Hispanic	Hispanic population	int			FALSE	FALSE
IncLess10	Number households w/income < \$10K	int			FALSE	FALSE
Inc10to15	Number households w/income \$10-15K	int			FALSE	FALSE
Inc15to25	Number households w/income \$15-25K	int			FALSE	FALSE
Inc25to35	Number households w/income \$25-35K	int			FALSE	FALSE
IncOver35	Number households w/income > \$35K	int			FALSE	FALSE
ResidDay	Population resident at day	int			FALSE	FALSE
ResidNight	Population resident at night	int			FALSE	FALSE
WorkingCom	Population working commercial sector	int			FALSE	FALSE
WorkingInd	Population working industrial sector	int			FALSE	FALSE
Commuting5PM	Population commuting at 5 PM	int			FALSE	FALSE
OwnerSingleUnits	Owner occupied single units	int			FALSE	FALSE
OwnerMultUnits	Owner occupied single units	int			FALSE	FALSE
OwnerMultStruct	Owner occupied multiple structures	int			FALSE	FALSE
OwnerMHs	Owner occupied mobile homes	int			FALSE	FALSE
RenterSingleUnits	Renter occupied single units	int			FALSE	FALSE
RenterMultUnits	Renter occupied multiple units	int			FALSE	FALSE
RenterMultStructs	Renter occupied multiple structures	int			FALSE	FALSE
RenterMHs	Renter occupied mobile homes	int			FALSE	FALSE
VacantSingleUnits	Vacant single units	int			FALSE	FALSE
VacantMultUnits	Vacant multiple units	int			FALSE	FALSE
VacantMultStructs	Vacant multiple structures	int			FALSE	FALSE
VacantMHs	Vacant mobile homes	int			FALSE	FALSE
BuiltBefore40	Units built before 1940	int			FALSE	FALSE
BuiltAfter40	Units built after 1940	int			FALSE	FALSE
AvgRent	Average monthly rent (\$)	int			FALSE	FALSE
AvgValue	Average value property (\$)	int			FALSE	FALSE

Table hzDemographicsT

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
Tract		char(11)	11		TRUE	TRUE
Population	Total population	int			FALSE	FALSE
Households	Number of households	int			FALSE	FALSE

Quarters	Number group quarters	int			FALSE	FALSE
Less16	Population age less than 16 years	int			FALSE	FALSE
Pop16to65	Population age 16 to 65 years	int			FALSE	FALSE
Over65	Population age over 65 years	int			FALSE	FALSE
White	White population	int			FALSE	FALSE
Black	Black population	int			FALSE	FALSE
Native	Native Ameican population	int			FALSE	FALSE
Asian	Asian population	int			FALSE	FALSE
Hispanic	Hispanic population	int			FALSE	FALSE
IncLess10	Number households w/incomde < \$10K	int			FALSE	FALSE
Inc10to15	Number households w/incomde \$10-15K	int			FALSE	FALSE
Inc15to25	Number households w/incomde \$15-25K	int			FALSE	FALSE
Inc25to35	Number households w/incomde \$25-35K	int			FALSE	FALSE
IncOver35	Number households w/incomde > \$35K	int			FALSE	FALSE
ResidDay	Population resident at day	int			FALSE	FALSE
ResidNight	Population resident at night	int			FALSE	FALSE
WorkingCom	Population working commercial sector	int			FALSE	FALSE
WorkingInd	Population working industrial sector	int			FALSE	FALSE
Commuting5PM	Population commuting at 5 PM	int			FALSE	FALSE
OwnerSingleUnits	Owner occupeid single units	int			FALSE	FALSE
OwnerMultUnits	Owner occupeid single units	int			FALSE	FALSE
OwnerMultStruct	Owner occupeid multiple structures	int			FALSE	FALSE
OnwerMHs	Owner occupeid mobile homes	int			FALSE	FALSE
RenterSingleUnits	Renter occupeid single units	int			FALSE	FALSE
RenterMultUnits	Renter occupeid multiple units	int			FALSE	FALSE
RenterMultStructs	Renter occupeid multiple structures	int			FALSE	FALSE
RenterMHs	Renter occupeid mobile homes	int			FALSE	FALSE
VacantSingleUnits	Vacant single units	int			FALSE	FALSE
VacantMultUnits	Vacant multiple units	int			FALSE	FALSE
VacantMultStructs	Vacant multiple structures	int			FALSE	FALSE
VacantMHs	Vacant mobile homes	int			FALSE	FALSE
BuiltBefore40	Units built before 1940	int			FALSE	FALSE
BuiltAfter40	Units built after 1940	int			FALSE	FALSE
AvgRent	Average monthly rent (\$)	int			FALSE	FALSE
AvgValue	Average value property (\$)	int			FALSE	FALSE

Table hzElectricPowerFlty

Name	Comment	Data Type	Length	Preci sion	Primary	Foreign Key
ElectricPowerFltyId		char(8)	8		TRUE	FALSE
UtilFciltyClass		char(5)	5		FALSE	TRUE
Tract		char(11)	11		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
Use		varchar(10)	10		FALSE	FALSE

FoundationType		char(1)	1		FALSE	FALSE
Anchor		bit			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
Capacity		int			FALSE	FALSE

Table hzEmergencyCtr

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
EocId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
EfClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Elevation		numeric(6,1)	6	1	FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
NumStories		tinyint			FALSE	FALSE
BackupPower		bit			FALSE	FALSE
Area	Area (sq.ft) for given occupancy	real			FALSE	FALSE
Capacity		int			FALSE	FALSE
Kitchen		bit			FALSE	FALSE

Table hzExposureOccupB

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
CensusBlock	Full census block number	char(16)	16		TRUE	TRUE
RES1		smallint			FALSE	FALSE
COM1		smallint			FALSE	FALSE
IND1		smallint			FALSE	FALSE
AGRI		smallint			FALSE	FALSE
RELI		smallint			FALSE	FALSE
GOVI		smallint			FALSE	FALSE
EDUI		smallint			FALSE	FALSE
RES1I		smallint			FALSE	FALSE
RES2I		smallint			FALSE	FALSE
RES3AI		smallint			FALSE	FALSE
RES3BI		smallint			FALSE	FALSE
RES3CI		smallint			FALSE	FALSE
RES3DI		smallint			FALSE	FALSE
RES3EI		smallint			FALSE	FALSE
RES3FI		smallint			FALSE	FALSE
RES4I		smallint			FALSE	FALSE
RES5I		smallint			FALSE	FALSE
RES6I		smallint			FALSE	FALSE
COM1I		smallint			FALSE	FALSE
COM2I		smallint			FALSE	FALSE
COM3I		smallint			FALSE	FALSE
COM4I		smallint			FALSE	FALSE
COM5I		smallint			FALSE	FALSE
COM6I		smallint			FALSE	FALSE

COM7I		smallint			FALSE	FALSE
COM8I		smallint			FALSE	FALSE
COM9I		smallint			FALSE	FALSE
COM10I		smallint			FALSE	FALSE
IND1I		smallint			FALSE	FALSE
IND2I		smallint			FALSE	FALSE
IND3I		smallint			FALSE	FALSE
IND4I		smallint			FALSE	FALSE
IND5I		smallint			FALSE	FALSE
IND6I		smallint			FALSE	FALSE
AGR1I		smallint			FALSE	FALSE
REL1I		smallint			FALSE	FALSE
GOV1I		smallint			FALSE	FALSE
GOV2I		smallint			FALSE	FALSE
EDU1I		smallint			FALSE	FALSE
EDU2I		smallint			FALSE	FALSE

Table hzExposureOccupT

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
Tract		char(11)	11		TRUE	TRUE
RESI		smallint			FALSE	FALSE
COMI		smallint			FALSE	FALSE
INDI		smallint			FALSE	FALSE
AGRI		smallint			FALSE	FALSE
RELI		smallint			FALSE	FALSE
GOVI		smallint			FALSE	FALSE
EDUI		smallint			FALSE	FALSE
RES1I		smallint			FALSE	FALSE
RES2I		smallint			FALSE	FALSE
RES3AI		smallint			FALSE	FALSE
RES3BI		smallint			FALSE	FALSE
RES3CI		smallint			FALSE	FALSE
RES3DI		smallint			FALSE	FALSE
RES3EI		smallint			FALSE	FALSE
RES3FI		smallint			FALSE	FALSE
RES4I		smallint			FALSE	FALSE
RES5I		smallint			FALSE	FALSE
RES6I		smallint			FALSE	FALSE
COM1I		smallint			FALSE	FALSE
COM2I		smallint			FALSE	FALSE
COM3I		smallint			FALSE	FALSE
COM4I		smallint			FALSE	FALSE
COM5I		smallint			FALSE	FALSE
COM6I		smallint			FALSE	FALSE
COM7I		smallint			FALSE	FALSE
COM8I		smallint			FALSE	FALSE
COM9I		smallint			FALSE	FALSE
COM10I		smallint			FALSE	FALSE
IND1I		smallint			FALSE	FALSE
IND2I		smallint			FALSE	FALSE
IND3I		smallint			FALSE	FALSE
IND4I		smallint			FALSE	FALSE
IND5I		smallint			FALSE	FALSE
IND6I		smallint			FALSE	FALSE
AGR1I		smallint			FALSE	FALSE
REL1I		smallint			FALSE	FALSE
GOV1I		smallint			FALSE	FALSE
GOV2I		smallint			FALSE	FALSE
EDU1I		smallint			FALSE	FALSE
EDU2I		smallint			FALSE	FALSE

Table hzFerryFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
FerryFltyId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
TranspFcltyClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
Use		varchar(10)	10		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
Anchor		bit			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
BldgType		char(4)	4		FALSE	FALSE
BackupPower		bit			FALSE	FALSE
Traffic		int			FALSE	FALSE

Table hzFireStation

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
FireStationId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
EfClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Elevation		numeric(6,1)	6	1	FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
NumStories		tinyint			FALSE	FALSE
BackupPower		bit			FALSE	FALSE
Area	Area (sq.ft) for given occupancy	real			FALSE	FALSE
Capacity		int			FALSE	FALSE
Kitchen		bit			FALSE	FALSE
NumTrucks		smallint			FALSE	FALSE

Table hzGenBldgScheme

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
GenBldgSchemeId	Id	int			TRUE	FALSE
eqBldgSchemeId		int			FALSE	TRUE
Occupancy	Specific occupancy class	char(5)	5		FALSE	FALSE
WPct	Pct of wood distribution	tinyint			FALSE	FALSE
CPct	Pct of concrete distribution	tinyint			FALSE	FALSE
SPct	Pct of steel distribution	tinyint			FALSE	FALSE

MPct	Pct of masonry distribution	tinyint			FALSE	FALSE
MhPct	Pct of mobile homes distribution	tinyint			FALSE	FALSE

Table hzHazmat

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
HazmatID	SIC Code	char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Cas		char(10)	10		FALSE	FALSE
ChemicalName		varchar(20)	20		FALSE	FALSE
ChemicalQuant		int			FALSE	FALSE
SIC		varchar(10)	10		FALSE	FALSE
EfClass		char(5)	5		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
EPAID		varchar(20)	20		FALSE	FALSE
PerAmount		real			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE

Table hzHighwayBridge

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
HighwayBridgeId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
BridgeClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
BridgeType		varchar(8)	8		FALSE	FALSE
Width		decimal(5,1)	5	1	FALSE	FALSE
NumSpans		tinyint			FALSE	FALSE
PipeLength		int			FALSE	FALSE
MaxSpanLength		decimal(6,0)	6		FALSE	FALSE
SkewAngle		decimal(2,0)	2		FALSE	FALSE
SeatLength		decimal(5,1)	5	1	FALSE	FALSE
SeatWidth		decimal(5,1)	5	1	FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
YearRemodeled		smallint			FALSE	FALSE
PierType		varchar(10)	10		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
ScourIndex		varchar(1)	1		FALSE	FALSE
Traffic		int			FALSE	FALSE
TrafficIndex		varchar(2)	2		FALSE	FALSE
Condition		varchar(3)	3		FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE

Table hzHighwaySegment

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
HighwaySegId		char(8)	8		TRUE	FALSE

CountyFips	Full county FIPS code	char(5)	5		FALSE	TRUE
SegmentClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
PipeLength		int			FALSE	FALSE
Traffic		int			FALSE	FALSE
Cost		money			FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
NumLanes		tinyint			FALSE	FALSE
Pavement		varchar(10)	10		FALSE	FALSE
Width		decimal(5,1)	5	1	FALSE	FALSE
Capacity		int			FALSE	FALSE

Table hzHighwayTunnel

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
HighwayTunnelId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
TunnelClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Type		char(5)	5		FALSE	FALSE
Width		decimal(5,1)	5	1	FALSE	FALSE
Length		real			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Traffic		int			FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE

Table hzLevees

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
LeveeId		char(8)	8		TRUE	FALSE
CountyFips	Full county FIPS code	char(5)	5		FALSE	TRUE

Table hzLightRailBridge

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
LightRailBridgeId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
BridgeClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
BridgeType		varchar(8)	8		FALSE	FALSE
Width		decimal(5,1)	5	1	FALSE	FALSE
NumSpans		tinyint			FALSE	FALSE
PipeLength		int			FALSE	FALSE
MaxSpanLength		decimal(6,0)	6		FALSE	FALSE
SkewAngle		decimal(2,0)	2		FALSE	FALSE
SeatLength		decimal(5,1)	5	1	FALSE	FALSE
SeatWidth		decimal(5,1)	5	1	FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
YearRemodeled		smallint			FALSE	FALSE
PierType		varchar(10)	10		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
ScourIndex		varchar(1)	1		FALSE	FALSE
Traffic		int			FALSE	FALSE

TrafficIndex		varchar(2)	2		FALSE	FALSE
Condition		varchar(3)	3		FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE

Table hzLightRailFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
LightRailFlty		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
TranspFclyClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
Use		varchar(10)	10		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
Anchor		bit			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
BldgType		char(4)	4		FALSE	FALSE
BackupPower		bit			FALSE	FALSE
Traffic		int			FALSE	FALSE

Table hzLightRailSegment

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
LightRailSegId		char(8)	8		TRUE	FALSE
CountyFips	Full county FIPS code	char(5)	5		FALSE	TRUE
SegmentClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
PipeLength		int			FALSE	FALSE
Traffic		int			FALSE	FALSE
Cost		money			FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
NumTracks		tinyint			FALSE	FALSE

Table hzLightRailTunnel

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
LightRailTunnelId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
TunnelClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Type		char(5)	5		FALSE	FALSE
Width		decimal(5,1)	5	1	FALSE	FALSE
Length		real			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Traffic		int			FALSE	FALSE

Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE

Table hzMilitary

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
MilitaryFtyld		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
HplfClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Elevation		numeric(6,1)	6	1	FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
ShelterCapacity		smallint			FALSE	FALSE
Use		varchar(10)	10		FALSE	FALSE
BldgCost		money			FALSE	FALSE
ContentCost		money			FALSE	FALSE

Table hzNaturalGasFty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
NaturalGasFtyld		char(8)	8		TRUE	FALSE
UtilFcltyClass		char(5)	5		FALSE	TRUE
Tract		char(11)	11		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
Use		varchar(10)	10		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
Anchor		bit			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
BackupPower		bit			FALSE	FALSE
Capacity		int			FALSE	FALSE

Table hzNaturalGasPI

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
NaturalGasPIld		char(8)	8		TRUE	FALSE
CountyFips	Full county FIPS code	char(5)	5		FALSE	TRUE
PipelinesClass		char(5)	5		FALSE	TRUE

Name		varchar(40)	40		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Material		varchar(10)	10		FALSE	FALSE
Diameter		numeric(5,1)	5	1	FALSE	FALSE
PipeLength		int			FALSE	FALSE
Joint		varchar(10)	10		FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
SourceId	Stores id of segment from which current record was split of (each pipeline is split into pieces at tract boundaries)	int			FALSE	FALSE

Table hzNuclearFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
NuclearFltyId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
HplfClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Elevation		numeric(6,1)	6	1	FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Capacity		int			FALSE	FALSE
BldgType		char(4)	4		FALSE	FALSE

Table hzOiFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
OilFltyId		char(8)	8		TRUE	FALSE
UtilFclyClass		char(5)	5		FALSE	TRUE
Tract		char(11)	11		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
Use		varchar(10)	10		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
Anchor		bit			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
BackupPower		bit			FALSE	FALSE
Capacity		int			FALSE	FALSE

Table hzOilPl

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
OilPlId	Full county FIPS code	char(8)	8	1	TRUE	FALSE
CountyFips		char(5)	5		FALSE	TRUE
PipelinesClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Material		varchar(10)	10		FALSE	FALSE
Diameter		numeric(5,1)	5		FALSE	FALSE
PipeLength		int			FALSE	FALSE
Joint		varchar(10)	10		FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
SourceId		int			FALSE	FALSE
	Stores id of segment from which current record was split of (each pipeline is split into pieces at tract boundaries)					

Table hzPoliceStation

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
PoliceStationId	Area (sq.ft) for given occupancy	char(8)	8	1	TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
EfClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Elevation		numeric(6,1)	6		FALSE	FALSE
Latitude		decimal(11,6)	11		FALSE	FALSE
Longitude		decimal(11,6)	11		FALSE	FALSE
NumStories		tinyint			FALSE	FALSE
BackupPower		bit			FALSE	FALSE
Area		real			FALSE	FALSE
Capacity		int			FALSE	FALSE
Kitchen		bit			FALSE	FALSE

Table hzPortFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
PortFltyId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
TranspFclyClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
Use		varchar(10)	10		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE

Anchor		bit			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
BldgType		char(4)	4		FALSE	FALSE
Capacity		int			FALSE	FALSE
NumBerths	Number of berths	smallint			FALSE	FALSE
NumCranes	Number of cranes	smallint			FALSE	FALSE

Table hzPotableWaterPI

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
PotableWaterPIId		char(8)	8		TRUE	FALSE
CountyFips	Full county FIPS code	char(5)	5		FALSE	TRUE
PipelinesClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Material		varchar(10)	10		FALSE	FALSE
Diameter		numeric(5,1)	5	1	FALSE	FALSE
PipeLength		int			FALSE	FALSE
Joint		varchar(10)	10		FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Comment	Stores id of segment from which current record was split of (each pipeline is split into pieces at tract boundaries)	varchar(40)	40		FALSE	FALSE
SourceId		int			FALSE	FALSE

Table hzRailFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
RailFltyId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
TranspFclyClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
Use		varchar(10)	10		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
Anchor		bit			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
BldgType		char(4)	4		FALSE	FALSE
BackupPower		bit			FALSE	FALSE
Traffic		int			FALSE	FALSE

Table hzRailwayBridge

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
RailwayBridgeId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
BridgeClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
BridgeType		varchar(8)	8		FALSE	FALSE
Width		decimal(5,1)	5	1	FALSE	FALSE
NumSpans		tinyint			FALSE	FALSE
PipeLength		int			FALSE	FALSE
MaxSpanLength		decimal(6,0)	6		FALSE	FALSE
SkewAngle		decimal(2,0)	2		FALSE	FALSE
SeatLength		decimal(5,1)	5	1	FALSE	FALSE
SeatWidth		decimal(5,1)	5	1	FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
YearRemodeled		smallint			FALSE	FALSE
PierType		varchar(10)	10		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
ScourIndex		varchar(1)	1		FALSE	FALSE
Traffic		int			FALSE	FALSE
TrafficIndex		varchar(2)	2		FALSE	FALSE
Condition		varchar(3)	3		FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE

Table hzRailwaySegment

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
RailwaySegId		char(8)	8		TRUE	FALSE
CountyFips	Full county FIPS code	char(5)	5		FALSE	TRUE
SegmentClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
PipeLength		int			FALSE	FALSE
Traffic		int			FALSE	FALSE
Cost		money			FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
NumTracks		tinyint			FALSE	FALSE

Table hzRailwayTunnel

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
RailwayTunnelId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
TunnelClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Type		char(5)	5		FALSE	FALSE
Width		decimal(5,1)	5	1	FALSE	FALSE
Length		real			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Traffic		int			FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE

Table hzRunway

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
RunwayId		smallint			TRUE	FALSE
AirportId		char(8)	8		FALSE	TRUE
TranspFclyClass		char(5)	5		FALSE	TRUE
AlternateName		varchar(40)	40		FALSE	FALSE
RunwayLength		decimal(8,2)	8	2	FALSE	FALSE
Capacity		int			FALSE	FALSE
Pavement		varchar(10)	10		FALSE	FALSE

Table hzSchool

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
SchoolId		char(8)	8		TRUE	FALSE
Tract		char(11)	11		FALSE	TRUE
EfClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Elevation		numeric(6,1)	6	1	FALSE	FALSE
Latitude		decimal(11,6)	11	6	FALSE	FALSE
Longitude		decimal(11,6)	11	6	FALSE	FALSE

Table hzSqFootageOccupB

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
CensusBlock	Full census block number	char(16)	16		TRUE	TRUE
RESF	Square footage for given occupy	real			FALSE	FALSE
COMF		real			FALSE	FALSE
INDF		real			FALSE	FALSE
AGRF		real			FALSE	FALSE
RELF		real			FALSE	FALSE
GOVF		real			FALSE	FALSE
EDUF		real			FALSE	FALSE
RES1F	Square footage for given occupy	real			FALSE	FALSE
RES2F	Square footage for given occupy	real			FALSE	FALSE
RES3AF	Square footage for given occupy	real			FALSE	FALSE
RES3BF	Square footage for given occupy	real			FALSE	FALSE
RES3CF	Square footage for given occupy	real			FALSE	FALSE
RES3DF	Square footage for given occupy	real			FALSE	FALSE
RES3EF	Square footage for given occupy	real			FALSE	FALSE
RES3FF	Square footage for given occupy	real			FALSE	FALSE
RES4F	Square footage for given occupy	real			FALSE	FALSE

RES6F	Square footage for given occupany	real			FALSE	FALSE
RES5F	Square footage for given occupany	real			FALSE	FALSE
COM1F	Square footage for given occupany	real			FALSE	FALSE
COM2F	Square footage for given occupany	real			FALSE	FALSE
COM3F	Square footage for given occupany	real			FALSE	FALSE
COM4F	Square footage for given occupany	real			FALSE	FALSE
COM5F	Square footage for given occupany	real			FALSE	FALSE
COM6F	Square footage for given occupany	real			FALSE	FALSE
COM7F	Square footage for given occupany	real			FALSE	FALSE
COM8F	Square footage for given occupany	real			FALSE	FALSE
COM9F	Square footage for given occupany	real			FALSE	FALSE
COM10F	Square footage for given occupany	real			FALSE	FALSE
IND1F	Square footage for given occupany	real			FALSE	FALSE
IND2F	Square footage for given occupany	real			FALSE	FALSE
IND3F	Square footage for given occupany	real			FALSE	FALSE
IND4F	Square footage for given occupany	real			FALSE	FALSE
IND5F	Square footage for given occupany	real			FALSE	FALSE
IND6F	Square footage for given occupany	real			FALSE	FALSE
AGR1F	Square footage for given occupany	real			FALSE	FALSE
REL1F	Square footage for given occupany	real			FALSE	FALSE
GOV1F	Square footage for given occupany	real			FALSE	FALSE
GOV2F	Square footage for given occupany	real			FALSE	FALSE
EDU1F	Square footage for given occupany	real			FALSE	FALSE
EDU2F	Square footage for given occupany	real			FALSE	FALSE

Table hzSqFootageOccupT

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
Tract RESF	Square footage for given occupany	char(11) real	11		TRUE FALSE	TRUE FALSE
COMF		real			FALSE	FALSE
INDF		real			FALSE	FALSE
AGRF		real			FALSE	FALSE
RELF		real			FALSE	FALSE
GOVF		real			FALSE	FALSE
EDUF		real			FALSE	FALSE

RES1F	Square footage for given occupany	real		FALSE	FALSE
RES2F	Square footage for given occupany	real		FALSE	FALSE
RES3AF	Square footage for given occupany	real		FALSE	FALSE
RES3BF	Square footage for given occupany	real		FALSE	FALSE
RES3CF	Square footage for given occupany	real		FALSE	FALSE
RES3DF	Square footage for given occupany	real		FALSE	FALSE
RES3EF	Square footage for given occupany	real		FALSE	FALSE
RES3FF	Square footage for given occupany	real		FALSE	FALSE
RES4F	Square footage for given occupany	real		FALSE	FALSE
RES6F	Square footage for given occupany	real		FALSE	FALSE
RES5F	Square footage for given occupany	real		FALSE	FALSE
COM1F	Square footage for given occupany	real		FALSE	FALSE
COM2F	Square footage for given occupany	real		FALSE	FALSE
COM3F	Square footage for given occupany	real		FALSE	FALSE
COM4F	Square footage for given occupany	real		FALSE	FALSE
COM5F	Square footage for given occupany	real		FALSE	FALSE
COM6F	Square footage for given occupany	real		FALSE	FALSE
COM7F	Square footage for given occupany	real		FALSE	FALSE
COM8F	Square footage for given occupany	real		FALSE	FALSE
COM9F	Square footage for given occupany	real		FALSE	FALSE
COM10F	Square footage for given occupany	real		FALSE	FALSE
IND1F	Square footage for given occupany	real		FALSE	FALSE
IND2F	Square footage for given occupany	real		FALSE	FALSE
IND3F	Square footage for given occupany	real		FALSE	FALSE
IND4F	Square footage for given occupany	real		FALSE	FALSE
IND5F	Square footage for given occupany	real		FALSE	FALSE
IND6F	Square footage for given occupany	real		FALSE	FALSE
AGR1F	Square footage for given occupany	real		FALSE	FALSE
REL1F	Square footage for given occupany	real		FALSE	FALSE
GOV1F	Square footage for given occupany	real		FALSE	FALSE
GOV2F	Square footage for given occupany	real		FALSE	FALSE

EDU1F	Square footage for given occupy	real			FALSE	FALSE
EDU2F	Square footage for given occupy	real			FALSE	FALSE

Table hzTract

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
Tract	Full county FIPS code	char(11)	11		TRUE	FALSE
CountyFips		char(5)	5		FALSE	TRUE
eqBldgSchemesId		int			FALSE	TRUE
Tract6		char(6)	6		FALSE	FALSE
TractArea		real			FALSE	FALSE
NumAggrBocks	Cenus tract area (sq. km) # blocks aggregated for tract. If flood hazard isn't included, this would be zero.	int		6	FALSE	FALSE
CenLat		decimal(11,6)	11		FALSE	FALSE
CenLongit		decimal(11,6)	11		FALSE	FALSE

Table hzWasteWaterFlty

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
WasteWaterFltyId		char(8)	8		TRUE	FALSE
UtilFoltyClass		char(5)	5		FALSE	TRUE
Tract		char(11)	11		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Address		varchar(40)	40		FALSE	FALSE
City		varchar(40)	40		FALSE	FALSE
Statea		char(2)	2		FALSE	FALSE
Zipcode		varchar(10)	10		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Contact		varchar(40)	40		FALSE	FALSE
Phone		varchar(14)	14		FALSE	FALSE
Use		varchar(10)	10		FALSE	FALSE
FoundationType		char(1)	1		FALSE	FALSE
Anchor		bit			FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE
Latitude		decimal(11,6)	11		FALSE	FALSE
Longitude		decimal(11,6)	11		FALSE	FALSE
Comment		varchar(40)	40		FALSE	FALSE
BackupPower		bit			FALSE	FALSE
Capacity		int			FALSE	FALSE

Table hzWasteWaterPI

Name	Comment	Data Type	Length	Precision	Primary	Foreign Key
WasteWaterPIId	Full county FIPS code	char(8)	8	1	TRUE	FALSE
CountyFips		char(5)	5		FALSE	TRUE
PipelinesClass		char(5)	5		FALSE	TRUE
Name		varchar(40)	40		FALSE	FALSE
Owner		varchar(25)	25		FALSE	FALSE
Material		varchar(10)	10		FALSE	FALSE
Diameter		numeric(5,1)	5		FALSE	FALSE
PipeLength		int			FALSE	FALSE
Joint		varchar(10)	10		FALSE	FALSE
YearBuilt		smallint			FALSE	FALSE
Cost		money			FALSE	FALSE

Comment SourceId	Stores id of segment from which current record was split of (each pipeline is split into pieces at tract boundaries)	varchar(40) int	40		FALSE FALSE	FALSE FALSE
---------------------	---	--------------------	----	--	----------------	----------------