UNISYS

Product Information Announcement

o New Release [✓ ● Revision] o Update o New Mail Code

Title

MCP/AS InfoExec™Semantic Information Manager (SIM) Technical Overview (8600 1674–002)

This announces a retitling and reissue of the *ClearPath HMP NX and A Series InfoExec™ Semantic Information Manager (SIM) Technical Overview*. No new technical changes have been introduced since the HMP 1.0 and SSR 43.2 release in June 1996.

To order a Product Information Library CD-ROM or paper copies of this document

- United States customers, call Unisys Direct at 1-800-448-1424.
- Customers outside the United States, contact your Unisys sales office.
- Unisys personnel, order through the electronic Book Store at http://iwww.bookstore.unisys.com.

Comments about documentation can be sent through e-mail to **doc@unisys.com**.

Announcement only:	Announcement and attachments:		
	AS197	System:	MCP/AS
		Release:	HMP 4.0 and SSR 45.1
		Date:	June 1998
		Part number:	8600 1674–002

MCP/AS

UNISYS

InfoExec™ Semantic Information Manager (SIM)

Technical Overview

HMP 4.0 and SSR 45.1 June 1998

Printed in USA

Priced Item 8600 1674–002

Correspondence regarding this publication should be forwarded to Unisys Corporation either by using the Business Reply Mail form at the back of this document or by addressing remarks to Software Product Information, Unisys Corporation, 25725 Jeronimo Road, Mission Viejo, CA 92691–2792 U.S.A.

Comments about documentation can also be sent through e-mail to **doc@unisys.com**.

Contents

Figures

Figures

Tables

About This Overview

Purpose

The InfoExec family is a collection of software products that give you full data management capabilities, including data definition and storage, security, query, and update. SIM is the underlying conceptual member of this software family. Because the concepts of SIM are significantly different from conventional database management approaches, you are encouraged to read this overview before attempting to use SIM or any of the other members of the InfoExec family.

Scope

This overview presents concepts rather than procedures. Because you use the InfoExec software products to manipulate SIM elements, it is important for you to understand those elements at a conceptual level.

Audience

The audience of this overview consists of those intending to use any of the products in the InfoExec family. The range of intended users encompasses programmers, database designers, and database administrators.

How to Use This Overview

This overview presents the SIM concepts in a serial manner, avoiding cross-references wherever possible. This overview contains examples to help illustrate major concepts. The examples refer to the ORGANIZATION database that is defined in detail in Section 5, "Basic Schema of a SIM Database."

Organization

This overview consists of the following sections. The information can be read serially or randomly, although it is recommended that you read the entire overview the first time through. In addition, an index appears at the end of this overview.

Section 1. What Is SIM?

This section introduces SIM and its relationship to the InfoExec family of products.

Section 2. Basic Concepts

This section describes the basic elements of SIM and of the databases you can create using SIM.

Section 3. Attributes

This section discusses the various characteristics, called attributes, that define elements of your SIM databases.

Section 4. Data Types

This section discusses the powerful data typing mechanisms that provide you with advanced integrity-maintenance capabilities.

Section 5. Basic Schema of a SIM Database

This section defines schema and presents a schema of an example SIM database, ORGANIZATION, to demonstrate the elements of SIM.

Related Product Information

Unless otherwise stated, all documents referred to in this publication are MCP/AS documents. The titles have been shortened for increased usability and ease of reading.

The following documents are included with the software release documentation and provide general reference information:

- The *Glossary* includes definitions of terms used in this document.
- The *Documentation Road Map* is a pictorial representation of the Product Information (PI) library. You follow paths through the road map based on tasks you want to perform. The paths lead to the documents you need for those tasks. The Road Map is available on the PI Library CD-ROM. If you know what you want to do, but don't know where to find the information, start with the Documentation Road Map.
- The *Information Availability List* (IAL) lists all user documents, online help, and HTML files in the library. The list is sorted by title and by part number.

The following documents provide information that is directly related to the primary subject of this publication.

DMSII Technical Overview

This overview describes the Data Management System II (DMSII) environment, which includes DMSII databases and software. This overview is written for end users, system programmers, database designers, and database administrators who need information on creating, changing, restoring, accessing, and maintaining a DMSII database.

InfoExec Advanced Data Dictionary System (ADDS) Operations Guide

The shortened title for this document is the *InfoExec ADDS Operations Guide*.

This guide describes InfoExec ADDS operations, such as creating and managing database descriptions. This guide is written for those who collect, organize, define, and maintain data and who are familiar with the Data Management System II (DMSII), the Semantic Information Manager (SIM), and the Structured Query Language Database (SQLDB).

InfoExec Capabilities Manual

This manual discusses the capabilities and benefits of the InfoExec data management system. This manual is written for executive and data processing management.

InfoExec DMS.View Operations Guide

This guide explains the InfoExec DMS.View utility and provides operating instructions for it. It is written for Data Management System II (DMSII) users who want to create a Semantic Information Manager (SIM) description and a Structured Query Language Database (SQLDB) description of a DMSII database.

InfoExec Interactive Query Facility (IQF) Operations Guide

The shortened title for this document is the *InfoExec IQF Operations Guide.*

This guide provides an overview of IQF and information about its inquiring, updating, and report painting capabilities. This guide is written primarily for IQF users with little programming knowledge, but also for programmers and database administrators.

InfoExec LINC.View Operations Guide

This guide provides operating instructions for the InfoExec LINC.View utility. It is written for Logic and Information Network Compiler II (LINC II) users who want to create a Semantic Information Manager (SIM) description or a Structured Query Language Database (SQLDB) description of a LINC II database.

InfoExec Semantic Information Manager (SIM) Object Definition Language (ODL) Programming Guide

The shortened title for this document is the *SIM ODL Programming Guide.*

This guide presents information on using ODL to define a SIM database. Each kind of ODL declaration that can be included in a schema is described. Use of the SIM Utility to create and maintain a SIM database is also described. This guide is written primarily for applications and systems developers responsible for defining and administering SIM databases.

InfoExec Semantic Information Manager (SIM) Object Manipulation Language (OML) Programming Guide)

The shortened title for this document is the *SIM OML Programming Guide.*

This guide describes how to interrogate and update SIM databases using SIM OML. Also described are two methods for processing queries: one method embeds calls on the SIM library in an application program, and the other method uses the InfoExec Interactive Query Facility (IQF). This guide is written for application programmers and experienced IQF users.

InfoExec Semantic Information Manager (SIM) Programming Guide

The shortened title for this document is the *InfoExec SIM Programming Guide*.

This guide describes how to use language extensions to access InfoExec SIM databases from application programs written in COBOL74, Pascal, and ALGOL. This guide is written for programmers who know at least one of the host languages thoroughly and who are familiar with SIM.

Personal Workstation^2 InfoExec Semantic Information Manager (SIM) Design Assistant (SDA) Operations Guide

The shortened title for this document is the *SDA Operations Guide*.

This guide describes the Semantic Information Manager (SIM) Design Assistant (SDA), which is used to design, browse, and maintain SIM database schemas on a personal computer (PC). It contains information on operating SDA, including descriptions and explanations of views, forms, and menus. This guide is written for anyone who designs, updates, or views SIM database schemas.

***Workstations InfoExec Workstation Query Facility (WQF) Operations* Guide**

The shortened title for this document is the *InfoExec WQF Operations Guide*.

This guide describes how to use WQF to query the structure or contents of an InfoExec database or to modify the data in the database. This guide is written for casual, infrequent users as well as expert users such as database analysts and administrators.

Section 1
What Is SIM?

The Semantic Information Manager (SIM) is a database management system (DBMS) that simplifies the task of modeling your application environment. SIM is based on the semantic data model, the next step in the evolution of data management tools.

Hierarchical, network, and relational data models allow you to describe the logical structure of a database using trees, collections of nodes and links, or tables. Those models provide general-purpose facilities for accessing and updating data structures. Each of these three kinds of models has been implemented commercially.

SIM, the first commercial implementation of the semantic data model, carries this modeling process a step further. It allows you to include more meaningful information in your database than is possible with conventional data models.

SIM is a step forward in DBMSs. Advances in data management are the result of several factors:

- Users have experienced frustration with physical and logical limitations of traditional data structures, such as files.
- The information needs of diverse users have drawn closer together, so the sharing of data is more desirable than it once was. However, data sharing requires greater central control over data for security and the assurance of integrity.
- End users demand direct and easy access to their data. The proliferation of microcomputers in business has stimulated an awareness of instant access to data, thus access to greater individual power.

Benefits of SIM and Semantic Modeling

SIM is a DBMS designed to overcome the limitations of hierarchical, network, and relational systems.

For example, in spite of the power and complexity of many DBMSs, programmers still must include integrity-checking routines in nearly every program written. These routines can amount to more than half of each program. SIM reduces and might even eliminate the need to provide such integrity-checking routines in application programs by providing system-defined integrity-maintenance mechanisms.

The most popular DBMSs in use up to now have been structured according to hierarchical and relational models. The data structures supported by those models are tied closely to the physical storage constraints of the computer. Because the databases you can construct with hierarchical and relational models are record and table-based, you are limited in representing complex relationships.

Semantic modeling allows you to represent complex data relationships while ensuring that certain relationships are restricted to maintain data integrity. In addition, SIM is set-oriented like the relational model, which means that you can manipulate sets of data rather than one record at a time. This design relieves application programs from the expensive overhead of performing many loops to retrieve and update records. With SIM you can even model your data in a hierarchical or relational fashion, if you want.

The following list summarizes the benefits of SIM:

- Strong real-world data modeling capabilities
  - Direct support for storing objects, including system-provided assignment of surrogates
  - Direct support for a generalization hierarchy that allows attribute inheritance through superclass/subclass relationships
  - Direct support for inter-object relationships that are defined through entity-valued attributes (EVAs)
- A well-defined data definition language, the Object Definition Language (ODL), that allows a schema to be created from a text file and that allows a schema, at the user's option, to be fully integrated with a data dictionary
- A well-defined data manipulation language, the Object Manipulation Language (OML), that needs no JOINs and that gives easy access to extended attributes through EVAs
- An optional workstation-based database design tool, the SIM Design Assistant (SDA)
- Data independence
  - Automatically provided data independence
  - A logical database description that is separated from the physical database description
  - Support of queries that access only what is needed
- Data integrity, including SUBRANGE, REQUIRED, UNIQUE, MAX, and generalized Boolean constraints
- Referential integrity that is automatically enforced through EVAs and the generalization hierarchy and that cannot be circumvented
- Internationalized character sets that support multiple collating sequences
- Application interfaces
  - Application program interface (API), through library calls from most languages
  - Host language interface (HLI), through compiler extensions for COBOL74, ALGOL, and Pascal
- Query and reporting interfaces
  - Interactive Query Facility (IQF), a terminal-based user interface that provides strong reporting features

– Workstation Query Facility (WQF), a workstation-based user interface

Note that SIM provides utilities (DMS.View and LINC.View) for users who wish to create a SIM layer for a DMSII or LINC II database. Refer to the *InfoExec DMS.View Operations Guide* or the *InfoExec LINC.View Operations Guide* for more information.

Data Independence

The InfoExec Environment is three layered, consisting of an external level or user view, a conceptual level, and a physical level. In many of the conventional DBMSs, the three levels are less clearly defined because the conceptual and external levels are constrained by physical storage requirements.

Figure 1–1 shows the relationship among the three levels.

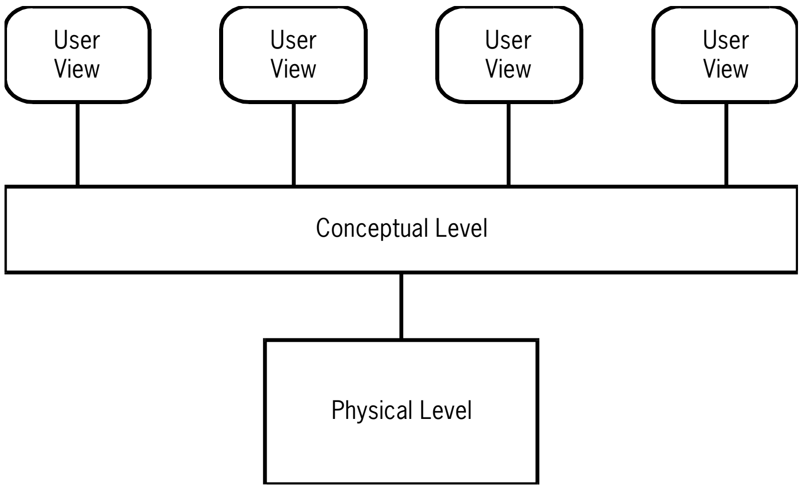

Figure 1–1. Three-Tiered Schema

SIM provides a number of concepts that are not tied to physical storage considerations. These conceptual elements are abstract enough to allow you to use them to model different views of applications. With SIM, you develop data descriptions and relationships to suit your application environment and users' needs. Other components of the InfoExec family of products then map this conceptual model to the appropriate physical elements. You are not limited by those physical elements and requirements.

By separating the conceptual from the physical, SIM provides data independence. Data independence implies that changes to your databases do not needlessly impair the corresponding applications.

Data Relationships

Semantics is the study of meanings. SIM, as a semantic data model, captures more of the meaning of data with database objects that emulate real-world objects. The data model gives additional meaning to data by supporting and enforcing relationships between objects. SIM provides the means to represent relationships of one-to-one, one-to-many, and many-to-many.

SIM provides the means to predefine relationships between many types of data. By predefining relationships, you are relieved from defining those relationships in program declarations or in query statements to the database.

Integrity Maintenance

How does SIM ensure integrity maintenance and thereby reduce the overhead of integrity validation routines in application programs?

SIM provides strong data typing capabilities by supporting a number of predefined types. SIM supports user-defined data types as well, which gives you even greater flexibility and power.

SIM allows you to define ranges and values, and performs range and value checking at run time.

SIM provides referential integrity. Referential integrity means that if a database object is updated, SIM automatically updates the relationships between that object and other objects in the database. Referential integrity overcomes the limitations of conventional DBMSs, which do not perform this type of integrity maintenance.

SIM provides the *verify* feature, which allows you to define conditions that must be satisfied by the database. Each verify that you declare specifies a condition. Any update transaction that violates this condition is aborted.

Database Definition

Three methods are available for defining all the elements of a database, including attributes, relationships, allowable ranges, values, and more. These methods are the following:

- Direct use of the Object Definition Language (ODL)
- Screen-based interface of the Advanced Data Dictionary System (ADDS)
- Workstation-based interface of the SIM Design Assistant (SDA)

ODL and ADDS are available to all InfoExec SIM users. SDA is available as an option; this product is used to design, browse, and maintain SIM database schemas on a personal computer (PC).

For more information on ODL, refer to the *SIM ODL Programming Guide*. For more information on ADDS, refer to the *InfoExec ADDS Operations Guide*. For more information on SDA, refer to the *SDA Operations Guide*.

Database Creation, Maintenance, and Storage

Whether you define a database using ODL, ADDS, or SDA, you can use the utilities provided by SIM to create and maintain the database. The database schema is protected by the security mechanisms provided by SIM and your master control program (MCP). You can optionally store the schema in ADDS, in which case the schema is also protected by the security mechanisms provided by ADDS.

Query Language

SIM supports immediate, improvised queries through the InfoExec Interactive Query Facility (IQF) and the Workstation Query Facility (WQF). These facilities provide interfaces for using the SIM Object Manipulation Language (OML). IQF screens and WQF windows help you develop queries by prompting you with the potential objects and characteristics you need to include in your formal query statement. The query statements you build are arranged according to simple, English-like syntax. For more information on these facilities, refer to the *InfoExec IQF Operations Guide* and the *InfoExec WQF Operations Guide*.

Applications can interface to SIM through a set of library calls. For more information on this kind of interface, refer to the *SIM OML Programming Guide*.

In addition, queries can be included in application programs by using selected language interfaces for ALGOL, COBOL74, and Pascal. For more information on these interfaces, refer to the following:

- *InfoExec Semantic Information Manager (SIM) Programming Guide*
- *ALGOL Programming Reference Manual, Volume 2: Product Interfaces*, the section on the SIM interface
- *COBOL ANSI-74 Programming Reference Manual, Volume 2: Product Interfaces*, the section on the SIM interface
- *Pascal Programming Reference Manual, Volume 2: Product Interfaces*, the section on the SIM interface

Summary

In summary,

- SIM provides facilities for representing data in more realistic structures and relationships.
- SIM provides data independence.
- SIM ensures data integrity with predefined relationships, powerful data typing mechanisms, range and value checking, referential integrity, and the *verify* feature.
- SIM provides security against unauthorized or unintentional access, and provides control over DBMS resources when it is used with ADDS.
- SIM supports multiple views of data through an advanced query facility and language extensions for application programs. Users can develop selective views through on-the-spot, improvised queries or through more formal programmatic means.

In addition, you can maintain SIM databases with utilities available through the Operations Control Manager (OCM).

Section 2
Basic Concepts

SIM terminology is used throughout the InfoExec family of products. The terms are found on various screens and menus. An understanding of the terms and concepts described in the following pages can help you design SIM databases.

Entity

The entity is the most important database object in SIM. Entities represent real objects, such as people, places, or things. For example, in a database describing the organization of a company, some entities might be the employee named John Smith, the spouse of John Smith, the Accounting department, and the Annual Report Preparation project.

Attribute

An attribute is a characteristic of an entity. For example, an employee can have attributes of NAME, ADDRESS, and EMPLOYEE-ID.

SIM offers the capability of defining various attributes to serve different needs. Every attribute is typed. An attribute type restricts the range of values that the attribute can have. Using attribute types is one method by which SIM prevents the storage of invalid data and thereby maintains the integrity of your database. Integrity criteria are further explained in Section 4, “Data Types.”

SIM supports several system-defined attribute types as well as those that you declare yourself. The system-defined attribute types are described in more detail in Section 3, “Attributes.”

Class

A class is a major structural component of a SIM database. It represents a collection of similar entities. For example, all the people who work for a company are members of the EMPLOYEE class.

Each class is given a name that must be unique in the database. In other words, you can only have one class called EMPLOYEE to represent company employees. The entities contained in a class share the same attributes.

The entities of a class do not have an inherent or predictable order. Queries can be used to return data ordered to your specifications. For more information about queries, refer to the *InfoExec IQF Operations Guide*, the *InfoExec WQF Operations Guide*, or the *SIM OML Programming Guide*.

Subclass-Superclass

A subclass is a class that is defined as a subset of another class. The class that serves as the PARENT class is known as the superclass. Just as a class defines a type of entity, a subclass defines a subtype of that entity. As you add layers of detail to your objects, they become more limited in what they represent. EMPLOYEE can refer to anybody who works for somebody else. ENGINEER is a specific kind of employee. Thus all engineers are contained in the subclass. A class can possess many subclasses.

In defining your database, you can build a hierarchy of classes and subclasses, known as a generalization hierarchy. You begin with at least one class and, from that, develop subclasses at levels of greater detail, as illustrated in Figure 2–1. During the analysis of your organization, you might find it easier to work from the bottom up, defining the various subclasses and then generalizing them into higher level groups. A class that has no superclasses is known as a base class.

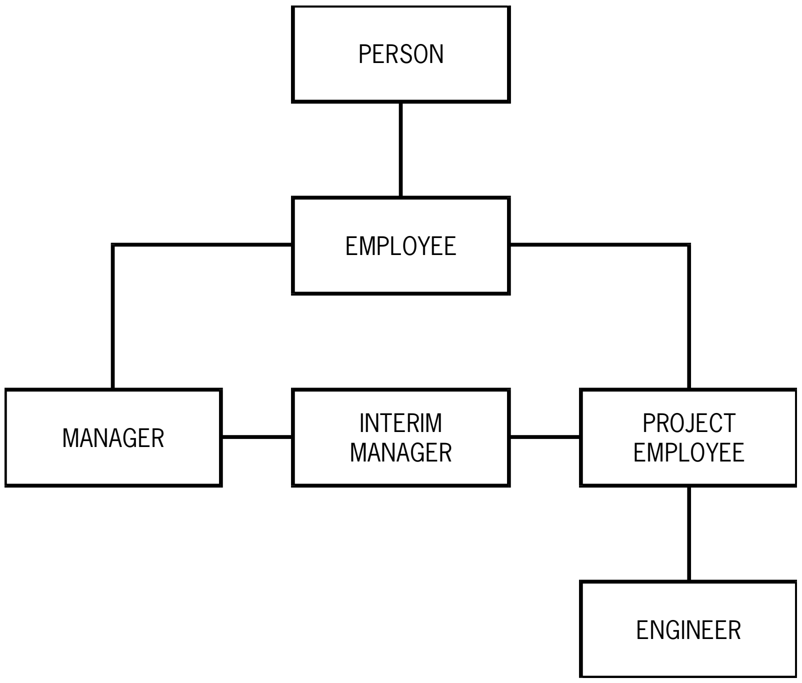

Figure 2–1. Generalization Hierarchy

Section 3
Attributes

The characteristics of an entity are known as its attributes. Attributes define a set of values that describe some aspect of the entity. The aggregation of a number of attributes defines the entity in more detail: the more attributes you ascribe to an entity, the more well-defined the entity. SIM offers two attribute types with which you can define entities: data-valued attributes (DVAs) and entity-valued attributes (EVAs).

Classes, representing entity types, are defined along with their attributes. In other words, the class PERSON, representing people, is defined with attributes such as NAME, ADDRESS, and AGE. A subclass-for instance, EMPLOYEE-is also defined with its specific attributes such as EMPLOYEE-MANAGER and EMPLOYEE-SALARY. However, EMPLOYEE, as a subclass of PERSON, automatically assumes the attributes defined for PERSON. Those attributes are inherited. Inherited attributes are discussed briefly under “Inherited Attributes” in this section.

Data-Valued Attribute (DVA)

A DVA is a characteristic that defines displayable values-that is, values you are probably familiar with as traits of data items. For instance, DVAs of EMPLOYEE could be NAME, AGE, and SALARY.

DVAs can also have other qualities. For example, they can be multivalued (MV) or single-valued (SV). These qualities are discussed later in this section under “Multivalued and Single-Valued Attributes”

Entity-Valued Attribute (EVA)

An EVA represents a directed relationship between an entity in its owning class and the entities of a target class. The target class can be any class, including the owning class of the EVA.

For the purpose of illustration, assume that you have defined a class called EMPLOYEE and a subclass called MANAGER. One EVA for entities of the EMPLOYEE class is EMPLOYEE-MANAGER. Because the values that can be assumed by EMPLOYEE-MANAGER are represented by entities in the MANAGER class, the EVA EMPLOYEE-MANAGER represents a directed relationship between EMPLOYEE entities and MANAGER entities. This relationship is illustrated in Figure 3–1.

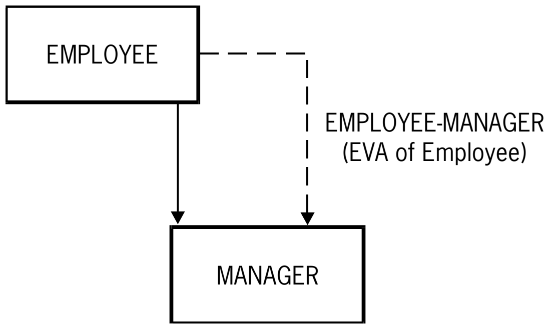

Figure 3–1. Entity-Valued Attribute (EVA)

Referential integrity is maintained because of the relationships implied by EVAs. An EVA cannot be assigned an invalid reference when it is inserted or modified through either the Interactive Query Facility (IQF), the Workstation Query Facility, or the extensions provided by the ALGOL, COBOL ANSI-74, or Pascal programming languages. Because the entities are explicitly referenced by the EVA on definition, the system checks the integrity of the modification or insertion. The link, so to speak, cannot be circumvented.

Also, when you delete an entity, all EVAs referencing that entity are automatically updated to reflect the deletion, as long as the update does not violate other integrity constraints.

EVAs can also have other qualities. For example, they can be multivalued or single-valued. These qualities are discussed later in this section under “Multivalued and Single-Valued Attributes.”

Subrole

The subrole is a very specific attribute, which is used to indicate alternate subclasses that can be applied to an entity. Each entity is specifically associated with a primary class-for instance, people who belong to the EMPLOYEE class. A subrole attribute for those people might be PROFESSION which can take possible values of MANAGER or PROJECT-EMPLOYEE. Each class can have only one subrole attribute.

In Figure 3–2, Mary Smith, a person, can have the subrole attribute of EMPLOYMENT, which takes values of either EMPLOYEE or PREVIOUS-EMPLOYEE. Because Mary is an employee, she can then also assume the subrole attribute of PROFESSION, which takes values of either PROJECT-EMPLOYEE or MANAGER.

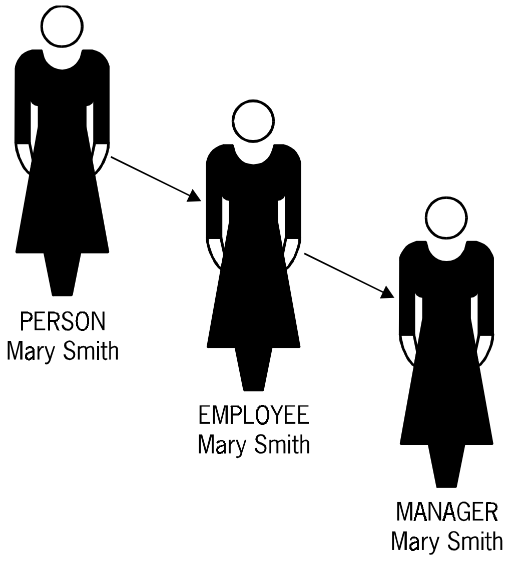

Figure 3–2. Subroles

Mary can play both roles, depending on the context in which the information about her is requested. For more information on queries and selective queries, refer to the *SIM OML Programming Guide*.

Inverse Attribute

All EVAs establish relationships that are bidirectional. An EVA names a relationship from one perspective, and its inverse EVA names the relationship from the opposing perspective. By default, SIM automatically creates inverse EVAs. However, you can declare these inverses explicitly to give them a precise definition.

In the preceding example for EVAs, the EVA, EMPLOYEE-MANAGER, points to the MANAGER class from the EMPLOYEE class. If you were to view the entities in the MANAGER class, then you might be interested in the names of the employees of a particular manager. Therefore, an appropriate name for the path from MANAGER to EMPLOYEE would be EMPLOYEES-MANAGING.

As mentioned under “Entity-Valued Attribute (EVA)” earlier in this section, other traits, such as the inverse, can be assigned to EVAs. Inverse attributes can be single-valued or multivalued. More information can be found in this section under “Multivalued and Single-Valued Attributes.”

Multivalued and Single-Valued Attributes

Multivalued attributes assume multiple values of the same type, at the same time. For example, EMPLOYEES-MANAGING is a multivalued attribute (MVA) of the class MANAGER. It is also an EVA, which makes it a multivalued EVA. Thus, in the context of this example, a manager can manage more than one employee at a time. The inverse EVA of EMPLOYEES-MANAGING, EMPLOYEE-MANAGER, is a single-valued attribute (SVA), which means that it can only assume one value at a time-that is, an employee can only have one manager at a time.

Because many attributes usually only take one value at a time-for instance, an employee usually only has one name, one salary, and one employee identification number-attributes default to single-valued. You must explicitly define them to be multivalued when you create the definitions. For more information on defining data, refer to the *SIM ODL Programming Guide.*

Immediate Attributes

Immediate attributes are those attributes that are explicitly defined for a class of entities. Some examples of immediate attributes are NAME, BIRTH-DATE, and AGE for the PERSON class; PROFESSION, EMPLOYEE-HIRE-DATE, and EMPLOYEE-SALARY for the EMPLOYEE class; PROJECT-NO, PROJECT-TITLE, and PROJECT-MANAGER for the PROJECT class.

Inherited Attributes

Inherited attributes are those attributes that are inherited by a subclass from its superclass. For example, because EMPLOYEE is a defined subclass of PERSON, EMPLOYEE inherits the attributes of PERSON. PERSON constitutes a generic representation of an entity. EMPLOYEE is more specific and therefore possesses characteristics that are not applicable to the class PERSON.

This relationship is illustrated in Figure 3–3.

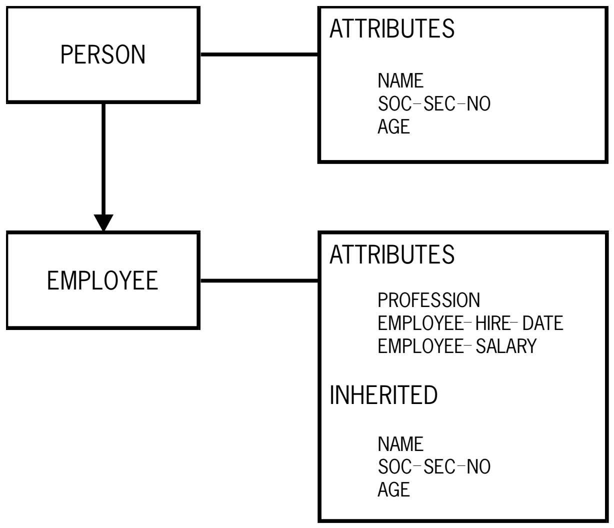

Figure 3–3. Inherited Attributes

Extended Attributes

Extended attributes are attributes related to a class through EVAs. As described earlier in this section under “Entity-Valued Attribute (EVA),” an EVA points to the class for which it is defined or to another class. The example used was EMPLOYEE-MANAGER as an attribute of EMPLOYEE. That EVA points to the MANAGER class, is single-valued, and assumes a manager in that class as its value.

As a result of the relationship established between the two classes by the EVA, the attributes of MANAGER are extended attributes of EMPLOYEE. That is, a link is formed from EMPLOYEE to MANAGER, providing access to managers’ attributes through the EMPLOYEE class. Because of this link, you can form queries that deal with attributes such as RETRIEVE NAME OF EMPLOYEE-MANAGER and RETRIEVE NAME OF SPOUSE OF EMPLOYEE-MANAGER. This capability can be important when developing queries into a SIM database and is discussed in the *SIM OML Programming Guide.*

Attribute Options

Attributes can be further defined by applying various options that SIM offers. There are three types of options available, as discussed in the following paragraphs:

- Cardinality option
- Initial value option
- Read-only option

Cardinality Option

Cardinality specifically refers to the number of elements in a set. The cardinality of an attribute designates whether the attribute can assume a single value or many values at once. This distinction is described earlier in this section under “Multivalued and Single-Valued Attributes.” An attribute can be either single-valued or multivalued, meaning that it can assume either one value at a time or many values at a time. The default is single-valued.

Several options are available for both SVAs and MVAs. Others are available only for MVAs. Options for both types of attributes are UNIQUE and REQUIRED. Options for MVAs only are DISTINCT and MAX.

UNIQUE

The UNIQUE option designates that the values of that attribute cannot be repeated for the entities of a particular class. For example, if you define the attribute PROJECT-NO of the PROJECT class as a UNIQUE attribute, you ensure that no two project numbers can be the same. If you attempt to make two project numbers identical, the last project number entered is disallowed.

The UNIQUE option applies to both MVAs and SVAs, both data-valued and entity-valued.

REQUIRED

The REQUIRED option designates that the attribute must have a value. With EVAs you can use this option on either the attribute itself or its inverse attribute. You cannot use the REQUIRED option on both.

DISTINCT

The DISTINCT option designates that values of a multivalued attribute for a given entity are different from each other. For example, if you define an MVA named CHILDREN whose values are the names of a person's children, you could define CHILDREN to be DISTINCT. This option requires that the values of the children for a person be distinct from one another. In other words, one person could not have two children named John.

By comparison, if you applied the UNIQUE option to CHILDREN, then only one person in the database could have a child named John. Clearly, this restriction would not be acceptable.

The DISTINCT option applies only to MVAs, both data-valued and entity-valued.

MAX

The MAX option assigns an upper limit on the number of values an MVA can assume. By default, there is no limit on the number of values assumable by an MVA.

Initial Value Option

The initial value option allows you to establish a default value for a data-valued attribute. For example, if a company rule states that a new employee always starts at level 5, then for the attribute LEVEL in class EMPLOYEE, you assign the initial value 5. That value is the resident value for that attribute until changed, either by updating the level number or by inserting a different level when first entering a new employee's data.

Read-Only Option

The read-only option protects the value of the attribute from update. In fact, if designated as read-only, that attribute value cannot be inserted, changed, or deleted by a user.

If this option is applied to an existing attribute for which data currently exists, it prevents that data from being changed. Also, the read-only option is used automatically by SIM in certain circumstances when you are converting existing Data Management Systems II (DMSII) databases to SIM.

Note: *The options described do not apply to class attributes, user-defined data types, or elements of a compound data type. These attribute options do apply, however, to both DVAs and EVAs. Data types are discussed in Section 4, "Data Types."*

Section 4
Data Types

Data types define a set of values that can be assumed by a particular data-valued attribute (DVA). Programming languages offer various data typing capabilities through picture statements, such as in COBOL, and in variable declarations in Pascal.

An advanced feature of SIM is integrity maintenance supported by a powerful data typing capability. DVAs with strong data types establish distinct rules for those attributes. Those rules help capture the semantics of each attribute, thus preventing undesirable operations.

SIM offers a number of data types: primitive, string, compound, symbolic, and user-defined.

Primitive Types

Primitive data types are those types that are probably most familiar to users. They include the following types:

- Integer

  Integer data types can only assume integer values. The sign of the number can be included. The default for the sign is plus (+). Ordering of the data is implied by its numeric values, that is, from highest to lowest. The range of allowable values for integers is between 549755813887 and –549755813887, inclusive.

- Real

  Real data types can assume floating-point numeric values, for example, 5.14 or –6.234. Real numbers can also be designated in the following format:

  ```
  <mantissa>E<exponent>
  ```

  The <mantissa> construct refers to the significant digits, the E indicates the exponent, and the <exponent> construct is expressed as *[+/–]<integer>*. For example, 5.14E+8 is equivalent to 5.14 multiplied by 10 raised to the power of 8.

  The ordering of real data is implied by the numeric value. The allowable range of values for real data types is between 4.31359146674E68 and –4.31359146674E68, inclusive.

- Character

  Character data types assume any characters of the underlying character set, which can be either EBCDIC or Kanji.

Note: *Kanji refers to the Japanese character set, also supported by SIM.*

- Boolean

  Boolean data types can assume either TRUE or FALSE as values. Boolean data types are not ordered.

- Time

  Time data types can assume time values in the format of HH:MM[:SS], where HH represents hours, MM represents minutes, and SS represents seconds. The range of allowable values for hours is between 00 and 23, inclusive. The range for minutes and seconds is between 00 and 59, inclusive. Seconds are optional.

  The system-defined function CURRENT-TIME gives the current time of the day stored in the system. You can use this function to assign values to a time data type attribute.

- Date

  Date data types can assume date values in the format of MM/DD/YY or MM/DD/YYYY, where MM represents month, DD represents day, and YY and YYYY represent year. A specification of YY assumes 19YY. The allowable range of values is between 1/1/0001 and 12/31/9999, inclusive. SIM enforces all the semantics of dates, including recognition of months with less than 31 days and recognition of years that include a leap day.

- Number

  Number data types can assume fixed point fractional or integer numeric values. You can store any numerical value in a format you designate in the following convention:

  ```
  <digit 1>,<digit 2>
  ```

  The <digit 1> construct represents the total number of digits, the <digit 2> construct represents the number of digits to the right of the decimal point, and the character S can be included to show that a number is signed. If S is not included, then the number is assumed to be positive. For instance, the first of the following examples can assume 123456.78, and the second can assume –123456789:

  ```
  NUMBER (8,2)

  NUMBER (S 9)
  ```

String Types

String data types consist of characters. The default character set is the EBCDIC character set. The maximum length of a string must be defined with an unsigned integer. The maximum allowable length for an EBCDIC string is 4095 characters. Strings can be a fixed or variable length and are ordered when the base character set is EBCDIC.

The alternative character set is Kanji. You can define Kanji strings. In this case, the maximum allowable length is 2047 characters.

String data types can also be based on user defined types. In the following example, the string CAPSTRING is based on the user-defined type UPPERCASE:

```
Type UPPERCASE = CHAR ("A".."Z");
TYPE CAPSTRING = STRING [20] OF UPPERCASE;

CLASS C
     (NAME: CAPSTRING; % contains uppercase only)
```

Compound Types

Compound data types consist of one or more other data types. Each type that is a component of the compound must be named and can be either a primitive type, a string, a symbolic, another compound, or a user-defined type. These elements are considered components of the compound.

For example, you can define a compound data type PHONE-NUMBER, as shown in the following example:

```
PHONE-NUMBER
    |
    |_ area-code: NUMBER (3)
    |_ prefix   : NUMBER (3)
    |_ suffix   : NUMBER (4)
```

Symbolic Types

Symbolic data types allow you to assign symbolic values to attributes. The values must be legal identifiers, following standard SIM naming conventions. The values must be unique identifiers in the database. A symbolic data type cannot share the values used by another symbolic in the same database unless that symbolic is declared as a subset of another. Doing so could be accomplished by using a range operator.

The following is an example of an attribute defined as a symbolic data type:

```
Degree: SYMBOLIC (HS, BA, BS, MA, MS, PHD) ORDERED
```

This example illustrates that a degree held by an employee can assume one of the values shown: either HS (High School), BA (Bachelor of Arts), BS (Bachelor of Science), MA (Master of Arts), MS (Master of Science), or PHD (Doctor of Philosophy). Note that by default a symbolic type is not considered to be ordered. However, the values assigned to this symbolic can be ordered by use of an ORDERED option.

Another typical example is that of the DAY-OF-WEEK attribute, which you can define as a symbolic type having the values of MON, TUES, WED, THU, FRI, SAT, SUN. You can define a subrange for an attribute called WORKING-DAYS, which would include the symbolic values MON, TUES, WED, THU, FRI from the DAY-OF-WEEK attribute. This definition is legal because WORKING-DAYS is a subset of DAY-OF-WEEK. Subranges are discussed in the following text.

Subranges

In the case of a value list, like the list for DAY-OF-WEEK, you can designate a subrange of values. This subrange can be applied to another attribute, for instance, WORKING-DAYS. When this assignment is done, the subrange functions as a data type. The subrange is a means of uniquely restricting the type of data that can be assumed by a designated attribute. The subrange is a selected part of a value list that has already been defined for another data type, often a symbolic data type. The subrange can consist of any of the following types:

- Integer
- Real
- Character
- Time
- Date
- Number
- String
- Symbolic
- User-defined (if the base type is not compound or Kanji)

When a string subrange is used, it must be compatible with the base character type of the corresponding string. When a user-defined subrange is used, it must be compatible with the base data type of the user-defined type whose values serve as the range.

User-Defined Types

You can define your own data types to suit particular needs. User-defined types can be useful in defining objects that consist of several components, are used repeatedly, or are special case items, such as DOLLAR or CURRENCY.

Any primitive, compound, symbolic, or other user-defined type can be used as the value for a user-defined data type. If you use another user-defined data type as a value, it must be defined first.

One advantage of user-defined types is that they allow you to attach more semantic meaning to data descriptions. In the following example, AGE and HEIGHT are both defined as integers. Physically, they are alike and can be compared. However, anyone looking at these data descriptions can see that they are semantically different.

```
User-Defined Data Type:

  AGE   : Integer

  HEIGHT: Integer
```

In some instances, user-defined types also can place integrity constraints on your data. In the following example, AGE is defined as a symbolic, with a set of subranges. Having a subrange enforces some integrity constraints because only valid data can fit in the subrange. Data not within the defined subrange are not allowed.

```
User-Defined Data Type:

      AGE    = Integer

      JUNIOR = Age(1.  .  .17)

      ADULT  = Age(18.  .  .99)

      SENIOR = Age(56.  .  .99)
```

Section 5
Basic Schema of a SIM Database

Now that you have an understanding of the building blocks of SIM, you might be wondering how they all fit together. This section defines and illustrates a schema by using an example SIM database called ORGANIZATION.

The schema is the logical description of your database representation of a real-world system or application environment. It describes database objects and their relationships.

You can think of the schema as a map the system uses when performing actions on the database or when traversing the database. You can store the schema of a SIM database in the InfoExec Advanced Data Dictionary System (ADDS). You can modify the schema by updating the individual objects and characteristics you have previously defined.

The schema is static and does not change except through definite design and modification procedures. The contents of your database are dynamic and can be changed through *ad hoc* or programmatic update facilities.

By adding more information to the schema, you create a more powerful and flexible system because, with more information, the database management system (DBMS) can assume more responsibility for the manipulation of database objects. The benefit of using SIM is that you can develop more flexible databases than with other DBMSs because it is possible to define objects and relationships with more detail.

You can also include comments in your schema. These comments can be read by interactive users or by application programmers through reports provided by InfoExec products. Comments are strings of no more than 255 characters. A comment can be assigned to the identifiers of classes, attributes, user-defined types, and index files. You can create and assign comments by using the text definition capabilities of ADDS.

For more information on defining the database and comments for individual structures, refer to the *SIM ODL Programming Guide.*

Figure 5–1 shows the relationships between the classes in the ORGANIZATION database. This database is used in examples throughout the overview. In the figure, the following conventions are used:

- Each oval represents a class.
- Class names are indicated within each oval.
- Broken lines between classes represent superclass-subclass relationships.
- Light lines with arrows represent single-valued EVA relationships between classes.

- Heavy lines with arrows represent multivalued EVA relationships between classes.
- EVA names are indicated above the light and heavy lines.

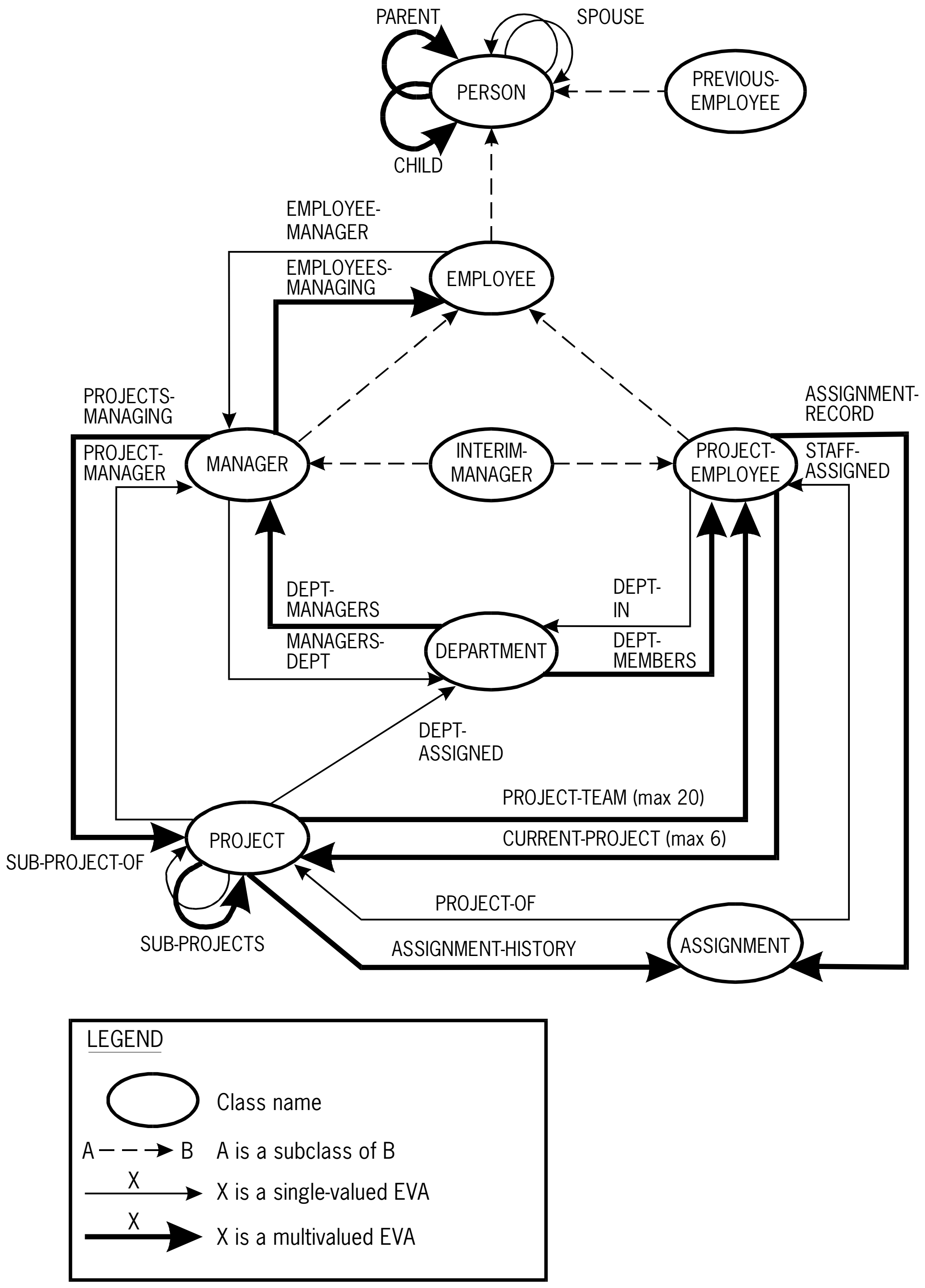

Figure 5–1. Relationships between the Classes in the ORGANIZATION Database

The following tables describe the schema used for the ORGANIZATION database. Names or descriptions that are given in uppercase letters describe classes, attributes, and, in some cases, allowable values. Names or descriptions that are initially capitalized or are given in lowercase letters indicate data types, options, or values.

Numbers within square brackets describe the number of characters or digits that can comprise a string or number. Words or values within parentheses indicate symbolic values and ranges of those values.

User-Defined Types

Table 5–1 describes the user-defined types used in the ORGANIZATION database.

Table 5–1. User-Defined Types

Attribute Name	Data Type	Values and Options
FLOOR	Symbolic	(FIRST-FLOOR, SECOND-FLOOR, THIRD-FLOOR, FOURTH-FLOOR, FIFTH-FLOOR)
DEGREE	Ordered Symbolic	(HS, BA, BS, MA, MS, PHD)
ADDRESS	STREET	: string [30]
	CITY	: string [20]
	STATE	: string [2]
	ZIPCODE	: number [9]
NAME-OF-PERSON	FIRST-NAME	: string [20]
	MID-INITIAL	: string [1]
	LAST-NAME	: string [20]
NEXT-OF-KIN	RELATIONSHIP	: string [20]
	NAME-OF-KIN	: NAME-OF-PERSON
	PHONE-NO	: string [10]
RANK-OF-MANAGER	Symbolic	(SUPERVISOR, DEPT-MANAGER, DIV-MANAGER, EXECUTIVE); ordered

Classes and Attributes

Tables 5–2 through 5–10 describe the classes and attributes in the ORGANIZATION database.

PERSON Class

Table 5–2. PERSON Class Attributes

Attribute Name	Data Type	Values and Options
EMPLOYED	Subrole	(EMPLOYEE, PREVIOUS-EMPLOYEE)
PERSON-ID	Integer	Unique; required
NAME	NAME-OF-PERSON	Required
BIRTH-DATE	Date	Required
AGE	Integer	(17 to 99)
MARITAL-STATUS	Symbolic	(SINGLE, MARRIED, DIVORCED); initial value is SINGLE
CURRENT-RESIDENCE	Address	Required
NEXT-OF-KIN	NEXT-OF-KIN	
US-CITIZEN	Boolean	Required
GENDER	Symbolic	(MALE, FEMALE)
SPOUSE	PERSON	Inverse is SPOUSE
CHILD	PERSON	Inverse is PARENT
PARENT	PERSON	Inverse is CHILD
EDUCATION		Multivalued
	DEGREE-OBTAINED	DEGREE
	YEAR-OBTAINED	Date
	GPA	Real

Notes:

- *PERSON-ID is ID-INDEX on PERSON.*
- *EDUCATION is EDUCATION-INDEX on PERSON.*

PREVIOUS-EMPLOYEE Subclass of PERSON

Table 5–3. PREVIOUS-EMPLOYEE Subclass

Attribute Name	Data Type	Values and Options
LEAVE-STATUS	Symbolic	(DISABILITY, LEAVE, RETIRED, QUIT); required
TERMINATION-REASON	String [30]	
LAST-WORK-DATE	Date	
HIRE-DATE	Date	

EMPLOYEE Subclass of PERSON

Table 5–4. EMPLOYEE Subclass

Attribute Name	Data Type	Values and Options
PROFESSION	Subrole	(PROJECT-EMPLOYEE, MANAGER); multivalued
EMPLOYEE-ID	Integer	Unique; required
EMPLOYEE-HIRE-DATE	Date	
EMPLOYEE-SALARY	Real	
EMPLOYEE-STATUS	Symbolic	(EXEMPT, NON-EXEMPT)
EMPLOYEE-MANAGER	MANAGER	Inverse is EMPLOYEES-MANAGING

MANAGER Subclass of EMPLOYEE

Table 5–5. MANAGER Subclass

Attribute Name	Data Type	Values and Options
MANAGER-TITLE	RANK-OF-MANAGER	Required
BONUS	Real	
EMPLOYEES-MANAGING	EMPLOYEE	Inverse is EMPLOYEE-MANAGER; multivalued
PROJECTS-MANAGING	PROJECT	Inverse is PROJECT-MANAGER; multivalued
MANAGERS-DEPT	DEPARTMENT	Inverse is DEPT-MANAGERS
TEMP-STATUS	Subrole	(INTERIM-MANAGER)

PROJECT-EMPLOYEE Subclass of EMPLOYEE

Table 5–6. PROJECT-EMPLOYEE Subclass

Attribute Name	Data Type	Values and Options
TITLE	Symbolic	(SENIOR, JUNIOR, STAFF, SPECIALIST); required
OVERALL-RATING	Number [3,1]	
CURRENT-PROJECT	PROJECT	Inverse is PROJECT-TEAM; Multivalued (MAX 6); required
DEPT-IN	DEPARTMENT	Inverse is DEPT-MEMBERS; required
ASSIGNMENT-RECORD	ASSIGNMENT	Inverse is STAFF-ASSIGNED; multivalued; required
TEMP-STATUS-2	Subrole	(INTERIM-MANAGER)

INTERIM-MANAGER Subclass of MANAGER and PROJECT-EMPLOYEE

Table 5–7. INTERIM-MANAGER Subclass

Attribute Name	Data Type	Values and Options
INTERIM-HISTORY		Multivalued
	START-DATE	Date
	END-DATE	Date

PROJECT Class

Table 5–8. PROJECT Class

Attribute Name	Data Type	Values and Options
PROJECT-NO	Integer	Unique; required
PROJECT-TITLE	String [20]	
PROJECT-TEAM	PROJECT-EMPLOYEE	Inverse is CURRENT-PROJECT; multivalued
DEPT-ASSIGNED	DEPARTMENT	
PROJECT-MANAGER	MANAGER	Inverse is PROJECTS-MANAGING
ASSIGNMENT-HISTORY	ASSIGNMENT	Inverse is PROJECT-OF; multivalued
SUB-PROJECT-OF	PROJECT	Inverse is SUB-PROJECTS
SUB-PROJECTS	PROJECT	Inverse is SUB-PROJECTS-OF; multivalued

DEPARTMENT Class

Table 5–9. DEPARTMENT Class

Attribute Name	Data Type	Values and Options
DEPT-TITLE	String [20]	
DEPT-NO	Integer	Unique; required
DEPT-LOCATION	FLOOR	
DEPT-MEMBERS	PROJECT-EMPLOYEE	Inverse is DEPT-IN; multivalued
DEPT-MANAGERS	MANAGER	Inverse is MANAGERS-DEPT; multivalued

ASSIGNMENT Class

Table 5–10. ASSIGNMENT Class

Attribute Name	Data Type	Values and Options
START-DATE	Date	
END-DATE	Date	
EST-PERSON-HOURS	Number [3]	
RATING	Number [3,1]	
STAFF-ASSIGNED	PROJECT-EMPLOYEE	Inverse is ASSIGNMENT-RECORD
PROJECT-OF	PROJECT	Inverse is ASSIGNMENT-HISTORY
ASSIGNMENT-NO	Integer	Unique; required

Index