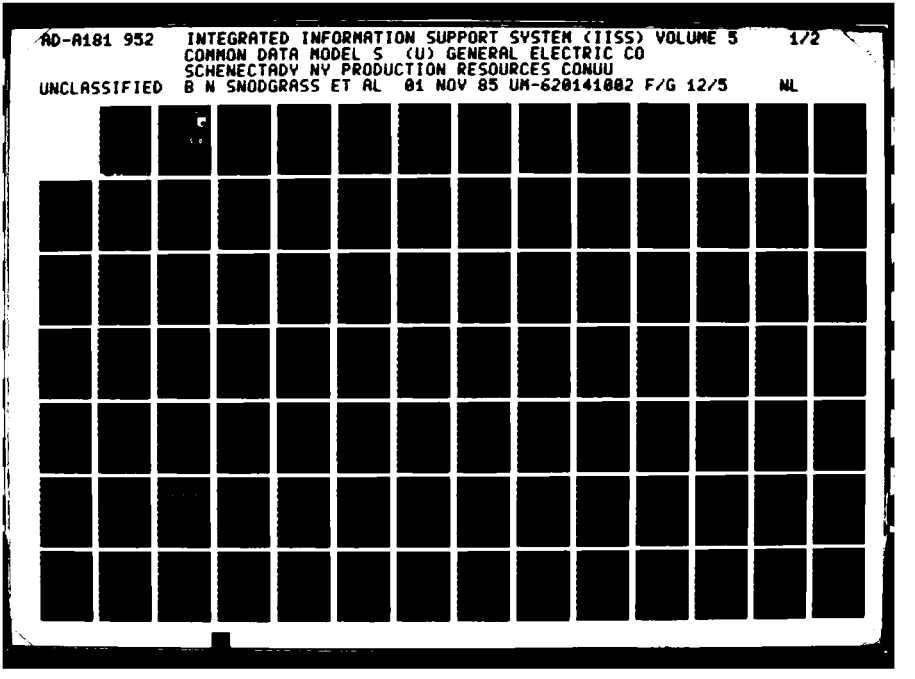
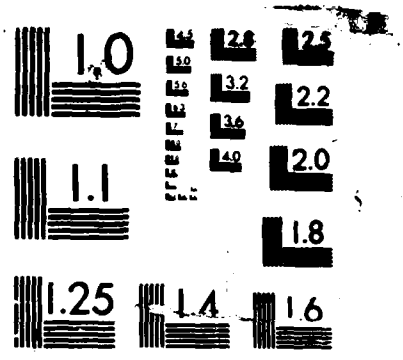

AD-A181 952 INTEGRATED INFORMATION SUPPORT SYSTEM (IISS) VOLUME 5 1/2
COMMON DATA MODEL S (U) GENERAL ELECTRIC CO
SCHENECTADY NY PRODUCTION RESOURCES CONUU
UNCLASSIFIED B N SNODGRASS ET AL 01 NOV 85 UM-620141002 F/G 12/5 NL



MICROCOPY RESOLUTION TEST CHART

(2)

AD-A181 952

DTIC FILE COPY

AFWAL-TR-86-4006
Volume V
Part 4

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 4 - Information Modeling Manual - IDEF1 Extended

General Electric Company
Production Resources Consulting
One River Road
Schenectady, New York 12345

Final Report for Period 22 September 1980 - 31 July 1985

November 1985

Approved for public release; distribution is unlimited.

PREPARED FOR:

MATERIALS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AFB, OH 45433-6533

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

DAVID L. JUDSON, PROJECT MANAGER
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

5 Aug 1986
DATE

FOR THE COMMANDER:

GERALD C. SHUMAKER, BRANCH CHIEF
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

7 Aug 86
DATE

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/MLTC, W-PAFB, OH 45433 to help us maintain a current mailing list."

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFWAL-TR-86-4006 Vol V, Part 4
6a. NAME OF PERFORMING ORGANIZATION General Electric Company Production Resources Consulting 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION AFWAL/MLTC
6c. ADDRESS (City, State and ZIP Code) 1 River Road Schenectady, NY 12345	7b. ADDRESS (City, State and ZIP Code) WPAFB, OH 45433-6533
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Materials Laboratory Air Force Systems Command, USAF 8b. OFFICE SYMBOL (If applicable) AFWAL/MLTC	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-80-C-5155
8c. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, Ohio 45433	10 SOURCE OF FUNDING NOS PROGRAM ELEMENT NO.: 78011F PROJECT NO.: 7500 TASK NO.: 62 WORK UNIT NO.: 01
11. TITLE (Include Security Classification) (See Reverse)	
12. PERSONAL AUTHOR(S) Snodgrass, R.M. Loomis, M.E. Rollins, D.R.	
13a. TYPE OF REPORT Final Technical Report 13b. TIME COVERED 22 Sept 1980 - 31 July 1985 14. DATE OF REPORT (Yr., Mo., Day) 1985 November 15. PAGE COUNT 142	
16. SUPPLEMENTARY NOTATION ICAM Project Priority 6201 The computer software contained herein are theoretical and/or references that in no way reflect Air Force-owned or -developed computer software.	

17 COSATI CODES		
FIELD	GROUP	SUB GR.
1308	0905	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This document is a modeling guide and reference manual for an extended version of the ICAM definition language for information modeling, referred to as IDEF1X. The IDEF1X syntax, procedure, and documentation requirements for developing a logical model of the semantic characteristics of data are described.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson	22b. TELEPHONE NUMBER (Include Area Code) 513-255-6976 22c. OFFICE SYMBOL AFWAL/MLTC

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

11. Title

Integrated Information Support System (IISS)
Vol V - Common Data Model Subsystem
Part 4 - Information Modeling Manual - IDEF1 Extended

PREFACE

This user's manual covers the work performed under Air Force Contract F33615-80-C-5155 (ICAM Project 6201). This contract is sponsored by the Materials Laboratory, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Gerald C. Shumaker, ICAM Program Manager, Manufacturing Technology Division, through Project Manager, Mr. David Judson. The Prime Contractor was Production Resources Consulting of the General Electric Company, Schenectady, New York, under the direction of Mr. Alan Rubenstein. The General Electric Project Manager was Mr. Myron Hurlbut of Industrial Automation Systems Department, Albany, New York.

Certain work aimed at improving Test Bed Technology has been performed by other contracts with Project 6201 performing integrating functions. This work consisted of enhancements to Test Bed software and establishment and operation of Test Bed hardware and communications for developers and other users. Documentation relating to the Test Bed from all of these contractors and projects have been integrated under Project 6201 for publication and treatment as an integrated set of documents. The particular contributors to each document are noted on the Report Documentation Page (DD1473). A listing and description of the entire project documentation system and how they are related is contained in document FTR620100001, Project Overview.

The subcontractors and their contributing activities were as follows:

TASK 4.2

Subcontractors	Role
Boeing Military Aircraft Company (BMAC)	Reviewer.
D. Appleton Company (DACOM)	Responsible for IDEF support, state-of-the-art literature search.
General Dynamics/ Ft. Worth	Responsible for factory view function and information models.

Subcontractors	Role
Illinois Institute of Technology	Responsible for factory view function research (IITRI) and information models of small and medium-size business.
North American Rockwell	Reviewer.
Northrop Corporation	Responsible for factory view function and information models.
Pritsker and Associates	Responsible for IDEF2 support.
SofTech	Responsible for IDEF0 support.

TASKS 4.3 - 4.9 (TEST BED)

Subcontractors	Role
Boeing Military Aircraft Company (BMAC)	Responsible for consultation on applications of the technology and on IBM computer technology.
Computer Technology Associates (CTA)	Assisted in the areas of communications systems, system design and integration methodology, and design of the Network Transaction Manager.
Control Data Corporation (CDC)	Responsible for the Common Data Model (CDM) implementation and part of the CDM design (shared with DACOM).
D. Appleton Company (DACOM)	Responsible for the overall CDM Subsystem design integration and test plan, as well as part of the design of the CDM (shared with CDC). DACOM also developed the Integration Methodology and did the schema mappings for the Application Subsystems.

Subcontractors	Role
Digital Equipment Corporation (DEC)	Consulting and support of the performance testing and on DEC software and computer systems operation.
McDonnell Douglas Automation Company (McAuto)	Responsible for the support and enhancements to the Network Transaction Manager Subsystem during 1984/1985 period.
On-Line Software International (OSI)	Responsible for programming the Communications Subsystem on the IBM and for consulting on the IBM.
Rath and Strong Systems Products (RSSP) (In 1985 became McCormack & Dodge)	Responsible for assistance in the implementation and use of the MRP II package (PIOS) that they supplied.
SofTech, Inc.	Responsible for the design and implementation of the Network Transaction Manager (NTM) in 1981/1984 period.
Software Performance Engineering (SPE)	Responsible for directing the work on performance evaluation and analysis.
Structural Dynamics Research Corporation (SDRC)	Responsible for the User Interface and Virtual Terminal Interface Subsystems.

Other prime contractors under other projects who have contributed to Test Bed Technology, their contributing activities and responsible projects are as follows:

Contractors	ICAM Project	Contributing Activities
Boeing Military Aircraft Company (BMAC)	1701, 2201, 2202	Enhancements for IBM node use. Technology Transfer to Integrated Sheet Metal Center (ISMC).

Contractors	ICAM Project	Contributing Activities
Control Data Corporation (CDC)	1502, 1701	IISS enhancements to Common Data Model Processor (CDMP).
D. Appleton Company (DACOM)	1502	IISS enhancements to Integration Methodology.
General Electric	1502	Operation of the Test Bed and communications equipment.
Hughes Aircraft Company (HAC)	1701	Test Bed enhancements.
Structural Dynamics Research Corporation (SDRC)	1502, 1701, 1703	IISS enhancements to User Interface/Virtual Terminal Interface (UI/VTI).
Systran	1502	Test Bed enhancements. Operation of Test Bed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Continued)

SECTION 1

INTRODUCTION

The purpose of this document is to define advanced concepts, techniques, and procedures for the development of logical models of the semantic characteristics of data. Within the business environment, these semantic data models may serve to support the management of data as a resource, the integration of information systems, and the building of computer databases.

The need for semantic data models was first recognized by the U.S. Air Force in the mid-seventies as a result of the Integrated Computer Aided Manufacturing (ICAM) Program. The objective of this program was to increase manufacturing productivity through the systematic application of computer technology. The ICAM Program identified a need for better analysis and communication techniques for people involved in improving manufacturing productivity. As a result, the ICAM Program developed a series of techniques known as the IDEF (ICAM Definition) Methods. IDEF includes three different modeling methodologies to graphically characterize the manufacturing business environment.

- IDEF0 is used to produce a "function model" which is a structured representation of the activities or processes within the environment or system.
- IDEF1 is used to produce an "information model" which represents the structure and semantics of information within the environment or system.
- IDEF2 is used to produce a "dynamics model" which represents the time varying behavioral characteristics of the environment or system.

IDEF1 was originally developed under the ICAM Program by Hughes Aircraft and D. Appleton Company (DACOM) based on internal developments of both companies as well as relational theory concepts developed by Dr. E.F. (Ted) Codd and entity-relationship modeling concepts by Dr. P.P.S. (Peter) Chen. Over the last five years, IDEF1 has been used extensively by both aerospace and non-aerospace companies.

In 1983, the U.S. Air Force initiated the Integrated

Information Support System (IISS) Project under the ICAM Program. The objective of this project was to provide the enabling technology to logically and physically integrate a network of heterogeneous computer hardware and software. The IISS approach to integration focuses on the capture, management, and use of a single semantic definition of the data resource referred to as a "Conceptual Schema". This Conceptual Schema is defined using the IDEF1 modeling technique.

This document defines an extended version of IDEF1 (referred to as IDEF1X) based on the requirements and experiences of the IISS project and applications within industry. Improvements to the technique included enhanced graphical representation, enhanced semantic richness, and simplified development procedures. Over the past five years, these extensions have been developed and tested by DACOM through various Air Force and private projects with both major aerospace corporations, such as General Dynamics, McDonnell Douglas, Rockwell International and General Electric, and with non-aerospace corporations, such as ARCO, Security Pacific National Bank and Schering Plough.

This document is structured to serve both as an IDEF1X modeling guide and as a reference manual. Section 2 discusses overall data modeling concepts. The specific syntax and semantics for an IDEF1X model are given in Section 3. Although different approaches may be used to create a model, Section 4 provides a basic procedure for model building, assuming limited automated support. The ICAM requirements for documentation and model validation techniques are presented in Section 5. A comparison of IDEF1X with IDEF1 along with a glossary and a list of references are contained in the appendices.

SECTION 2

DATA MODELING CONCEPTS

The focus of this manual is on the syntax and procedure for IDEF1X data models. However, before getting into the technical details of IDEF1X in Sections 3 and 4, this Section will discuss why data modeling is important and what are the overall objectives of the IDEF1X approach.

2.1 Managing Data as a Resource

Over the past decade, there has been a growing awareness among major corporations for the need to manage data as a resource. Perhaps one of the drivers to manage data as a resource is the requirement for flexibility in order to compete in a very dynamic business environment. Many companies must continually realign their organizations and procedures to adjust for advancements in technology and shifts in the market places. In order to realign quickly and smoothly, companies must recognize and manage the infrastructure of the business which includes understanding the data and associated knowledge required to run the business.

Many companies have formed special groups, such as Data Administration or Information Resource Management, in order to tackle the problem of managing data. The difficulty of their jobs, however, is compounded by the rapid and diverse growth of data. According to the Garner Group, a Stamford CT, market research company, the average large corporation will require online access to one trillion bytes of data by 1990, 50 times the amount of data needed in 1985. The creation and use of this data will be spread throughout the corporation. IBM has stated that by the end of 1987 as many as 14 million business professionals will use workstations to house and process their own data. Furthermore, an ICAM study showed that the data that already exists is generally inconsistent, untimely, inflexible, inaccessible, and unaligned with current business needs.

In order to manage data, we must understand its basic characteristics. Data can be thought of as a symbolic representation of facts with meanings. A single meaning can be applied to many different facts. For example, the meaning "zip code" could be applied to numerous five digit numbers. A fact without a meaning is of no value and a fact with the wrong meaning can be disastrous. Therefore, the focus of data

management must be on the meaning associated with data.

"Information" can be defined as an aggregation of data for a specific purpose or within a specific context. See Figure 2-1. This implies that many different types of information can be created from the same data. Statistically, 400 pieces of data could be combined 10 to the 869 power different ways to create various forms of information. Thus, the strategy to manage the information resource must focus on managing the meanings applied to facts, rather than attempting to control or limit the creation of information.

2.2 The Three Schema Concept

Over the years, the skill and interest in building information systems has grown tremendously. However, for the most part, the traditional approach to building systems has only focused on defining data from two distinct views, the user view and the computer view. From the user view, which will be referred to as the "external schema", the definition of data is in the context of reports and screens designed to aid individuals in doing their specific jobs. The required structure of data from a usage view changes with the business environment and the individual preferences of the user. From the computer

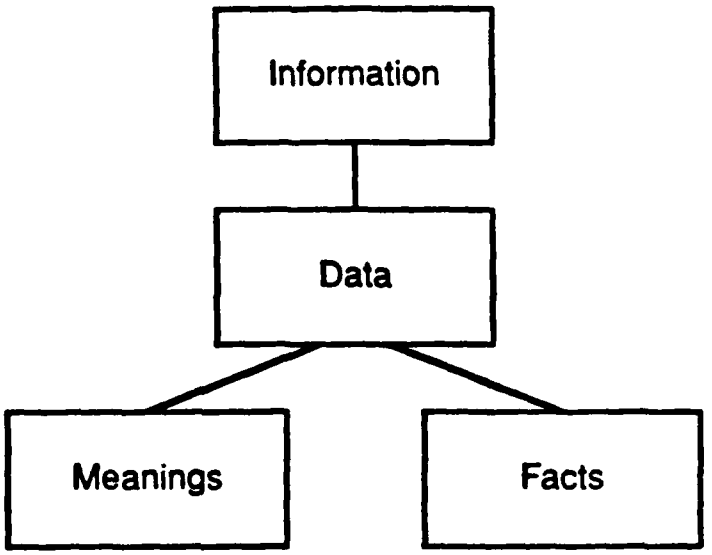


Figure 2-1. Components of Information

view, which will be referred to as the "internal schema", data is defined in terms of file structures for storage and retrieval. The required structure of data for computer storage depends upon the specific computer technology employed and the need for efficient processing of data.

These two views of data have been defined by analysts over the years on an application by application basis as specific business needs were addressed. See Figure 2-2. Typically, the internal schema defined for an initial application cannot be readily used for subsequent applications, resulting in the creation of redundant and often inconsistent definition of the same data. Data was defined by the layout of physical records and processed sequentially in early information systems. The

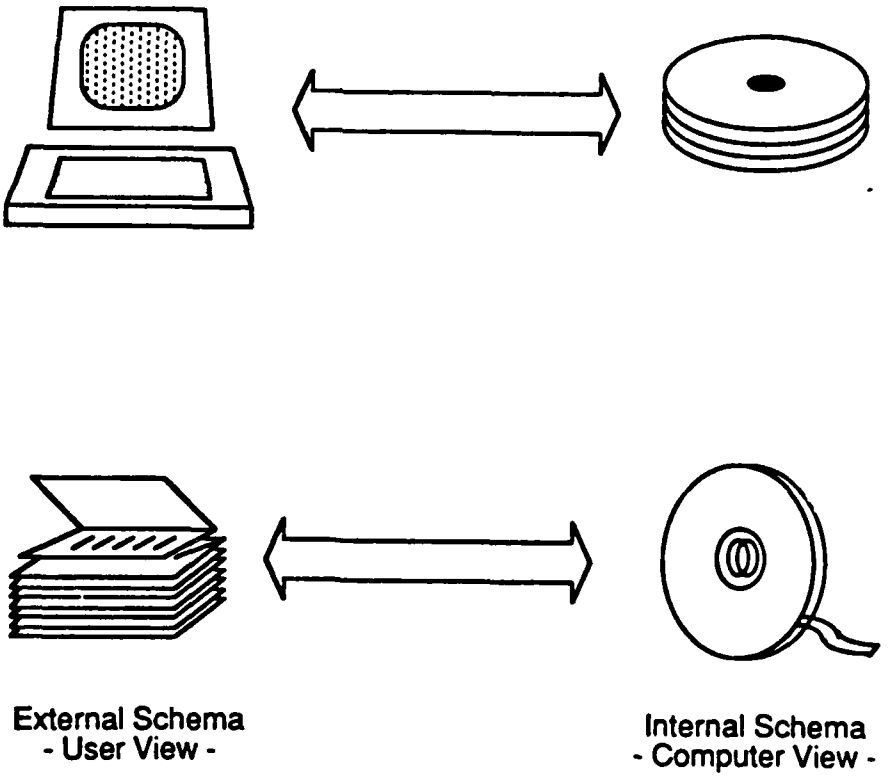


Figure 2-2. Traditional Views of Data

need for flexibility, however, lead to the introduction of Database Management Systems (DBMS's), which allow for random access of logically connected pieces of data. The logical data structures within a DBMS are typically defined as either hierarchies, networks or relations. Although DBMS's have greatly improved the shareability of data, the use of a DBMS alone does not guarantee a consistent definition of data. Furthermore, most large companies have had to develop multiple databases which are often under the control of different DBMS's and still have the problems of redundancy and inconsistency.

The recognition of this problem led the ANSI/X3/SPARC Study Group on Database Management Systems to conclude that in an ideal data management environment a third view of data is needed. This view, referred to as a "conceptual schema" is a single integrated definition of the data within an enterprise which is unbiased toward any single application of data and is independent of how the data is physically stored or accessed. See Figure 2-3. The primary objective of this conceptual schema is to provide a consistent definition of the meanings and interrelationship of data which can be used to integrate,

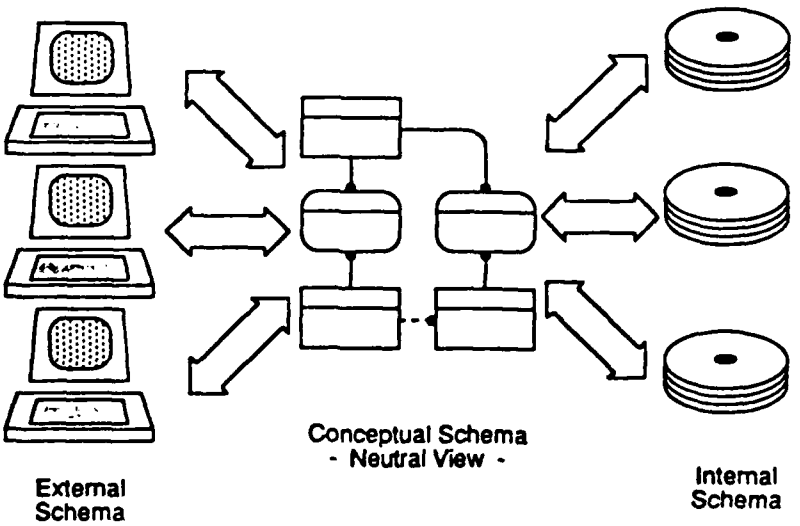


Figure 2-3. Three-Schema Approach

share, and manage the integrity of data. A conceptual schema must have three important characteristics:

1. It must be consistent with the infrastructure of the business and be true across all application areas.

2. It must be extendible, such that, new data can be defined without altering previously defined data.

3. It must be transformable to both the required user views and to a variety of data storage and access structures.

2.3 Objectives of Data Modeling

The logical data structure of a DBMS, whether hierarchical, network, or relational, cannot totally satisfy the requirements for a conceptual definition of data because it is limited in scope and biased toward the implementation strategy employed by the DBMS. Therefore, the need to define data from a conceptual view has lead to the development of semantic data modeling techniques. That is, techniques to define the meaning of data within the context of its interrelationships with other data. As illustrated in Figure 2-4, the real world, in terms of resources, ideas, events, etc., are symbolically defined within physical data stores. A semantic data model is an abstraction which defines how the stored symbols related to the real world. Thus, the model must be a true reflection of the real world.

A semantic data model can be used to serve many purposes. Some key objectives include:

1. Planning of Data Resources

   A preliminary data model can be used to provide an overall view of the data required to run an enterprise. The model can then be analyzed to identify and scope projects to build shared data resources.

2. Building of Shareable Databases

   A fully developed model can be used to define an application independent view of data which can be validated by users and then transformed into a physical database design for any of the various DBMS technologies. In addition to generating databases

which are consistent and sharable, development costs can be drastically reduced through data modeling.

3. Evaluation of Vendor Software

Since a data model actually reflects the infrastructure of an organization, vendor software can be evaluated against a company's data model in order to identify possible inconsistencies between the infrastructure implied by the software and the way the company actually does business.

4. Integration of Existing Databases

By defining the contents of existing databases with semantic data models, an integrated data definition can be derived. With the proper technology, the resulting conceptual schema can be used to control transaction processing in a distributed database environment. The U.S. Air Force Integrated Information Support System (IISS) is an experimental development and demonstration of this type of technology applied to a heterogeneous DBMS environment.

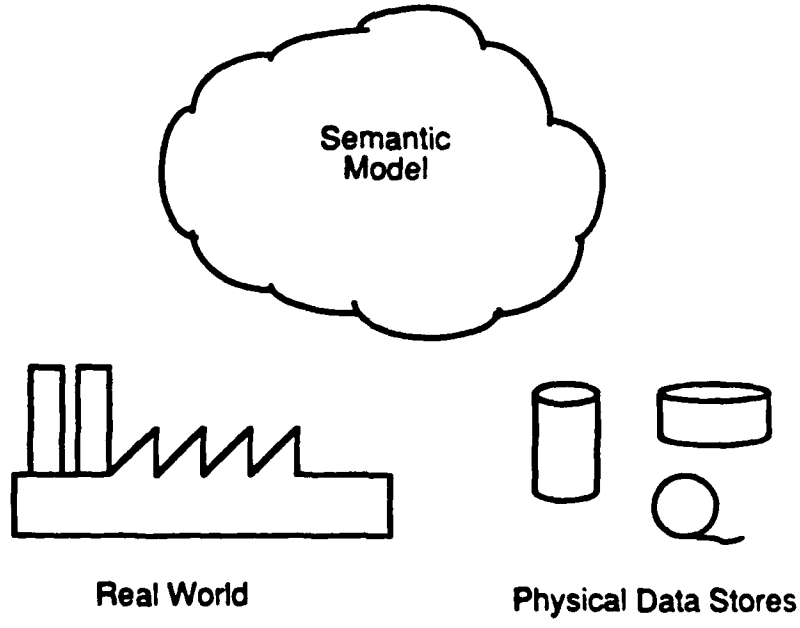


Figure 2-4. Semantic Data Models

2.4 The IDEF1X Approach

IDEF1X is the semantic data modeling technique described by this document. The IDEF1X technique was developed to meet the following requirements:

1. Support the development of conceptual schemas.

   The IDEF1X syntax supports the semantic constructs necessary in the development of a conceptual schema. A fully developed IDEF1X model has the desired characteristics of being consistent, extensible, and transformable.

2. Be a coherent language.

   IDEF1X has a simple, clean consistent structure with distinct semantic concepts. The syntax and semantics of IDEF1X are relatively easy for users to grasp, yet powerful and robust.

3. Be teachable.

   Semantic data modeling is a new concept for many IDEF1X users. Therefore, the teachability of the language was an important consideration. The language is designed to be taught to and used by business professionals and system analysts as well as data administrators and database designers. Thus, it can serve as an effective communication tool across interdisciplinary teams.

4. Be well-tested and proven.

   IDEF1X is based on years of experience with predecessor techniques and has been thoroughly tested both in Air Force development projects and in private industry.

5. Be automatable.

   IDEF1X diagrams can be generated by a variety of graphics packages. In addition, an active three-schema dictionary has been developed by the Air Force which uses the resulting conceptual schema for an application development and transaction processing in a distributed heterogeneous environment.

Commercial software is also available which supports the refinement, analysis, and configuration management of IDEF1X models.

IDEF1X uses an entity-relationship approach to semantic data modeling. The original development of IDEF1 was an extension to the entity-relationship modeling concepts of Dr. P.P.S. (Peter) Chen combined with relational theory concepts developed by Dr. E.F. (Ted) Codd. In addition to improvements in the graphical representation and modeling procedures, IDEF1X enhancements to the semantic richness include the introduction of categorization relationships (also called generalization). The IDEF1X language also incorporates commercial development work of the D. Appleton Company and The Database Design Group.

The basic constructs of an IDEF1X model are:

1. Things about which data is kept, e.g. people, places, ideas, events, etc., represented by a box;

2. Relationships between those things, represented by lines connecting the boxes; and

3. Characteristics of those things represented by attribute names within the box.

The basic constructs are shown in Figure 2-5, and expanded upon in the remained of this document.

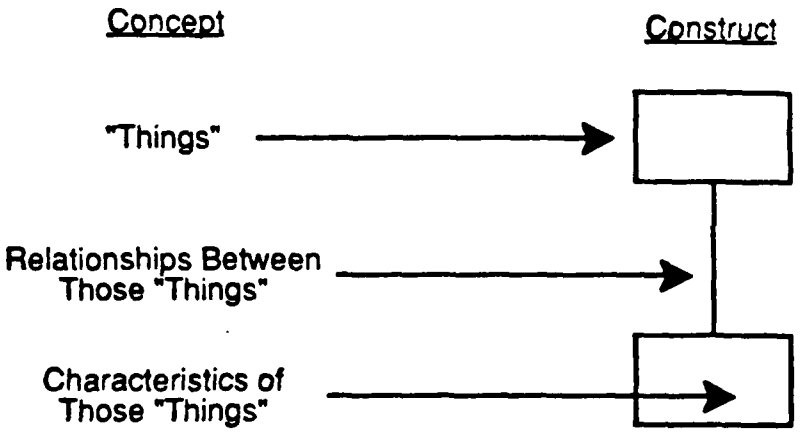


Figure 2-5. Basic Modeling Concepts

SECTION 3

IDEF1X SYNTAX AND SEMANTICS

This Section will discuss the semantics (or meaning) of each component of an IDEF1X model, the graphical syntax for representing the component, and rules governing its use. Although the components are highly interrelated, each one is discussed separately without regard for the actual sequence of construction. Section 4 discusses the procedure for building an IDEF1X model which will conform to the defined syntax and semantics.

The components of an IDEF1X model are:

1. Entities
   - Identifier-Independent Entities
   - Identifier-Dependent Entities

2. Relationships
   - Identifying Connection Relationships
   - Non-Identifying Connection Relationships
   - Categorization Relationships
   - Non-specific Relationships

3. Attributes/Keys
   - Attributes
   - Primary Keys
   - Alternate Keys
   - Foreign Keys

3.1 Entities

Entity Semantics

An "entity" represents a set of real or abstract things (people, objects, places, events, states, ideas, pairs of things, etc.) which have common attributes or characteristics. An individual member of the set is referred to as an "entity instance". A real world object or thing may be represented by more than one entity within a data model. For example, John Doe may be an instance of both the entity EMPLOYEE and BUYER. Furthermore, an entity instance may represent a combination of real world objects. For example, John and Mary could be an instance of the entity MARRIED-COUPLE.

An entity is "identifier-independent" or simply "independent" if each instance of the entity can be uniquely identified without determining its relationship to another entity. An entity is "identifier-dependent" or simply "dependent" if the unique identification of an instance of the entity depends upon its relationship to another entity.

Entity Syntax

An entity is represented as a box as shown in Figure 3-1. If the entity is identifier-dependent then the corners of the box are rounded. Each entity is assigned a unique name and number which are separated by a slash, "/", and placed above the box. The entity number is a positive integer. The entity name is a noun phrase (a noun with optional adjectives and prepositions) that describe the set of things the entity represents. The noun phrase is singular, not plural. Abbreviations and acronyms are permitted, however, the entity name must be meaningful and consistent throughout the model. A formal definition of the entity and a list of synonyms or aliases must be defined in the model glossary. Although an entity may be drawn in any number of diagrams, it only appears once within a given diagram.

Entity Rules

1. Each entity must have a unique name and the same meaning must always apply to the same name. Furthermore, the same meaning cannot apply to different names unless the names are aliases.

2. An entity has one or more attributes which are either owned by the entity or inherited through a relationship (See Foreign Keys in Section 3.7).

3. An entity has one or more attributes which uniquely identify every instance of the entity. (See Primary and Alternate Keys in Section 3.6).

4. Any entity can have any number of relationships with other entities in the model.

5. If an entire foreign key is used for all or part of an entity's primary key, then the entity is identifier-dependent. Conversely, if only a portion of a foreign key or no foreign key attribute at all is used for an entity's primary key, then the entity is identifier-independent.

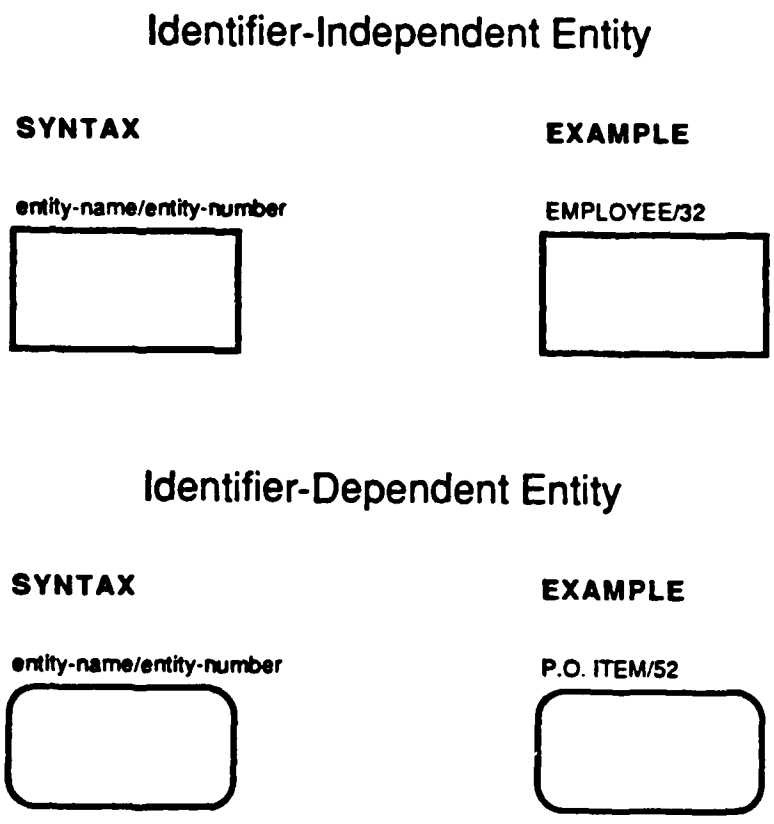


Figure 3-1. Entity Syntax

3.2 Connection Relationships

Connection Relationship Semantics

A "specific connection relationship" or simply "connection relationship" (also referred to as a "parent-child or existence-dependency relationship") is an association or connection between entities in which each instance of one entity, referred to as the parent entity, is associated with zero, one, or more instances of the second entity, referred to as the child entity, and each instance of the child entity is associated with exactly one instance of the parent entity. That is, an instance of the child entity can only exist if an associated instance of the parent entity exists. For example, a specific connection relationship would exist between the entities BUYER and PURCHASE-ORDER, if a buyer issues zero, one, or more purchase orders and each purchase order must be issued

by a single buyer. An IDEF1X model depicts the type or set of relationship between two entities. A specific instance of the relationship associates specific instances of the entities. For example, "buyer John Doe issued Purchase Order number 123" is an instance of a relationship.

The connection relationship may be further defined by specifying the cardinality of the relationship. That is, the specification of how many child entity instances may exist for each parent instance. Within IDEF1X, the following relationship cardinalities can be expressed:

1. Each parent entity instance may have zero, one or more associated child entity instances.

2. Each parent entity instance must have at least one or more associated child entity instances.

3. Each parent entity instance can have none or at most one associated child instance.

4. Each parent entity instance is associated with some exact number of child entity instances.

If an instance of the child entity is identified by its association with the parent entity, then the relationship is referred to as an "identifying relationship". For example, if one or more tasks are associated with each project and tasks are only uniquely identified within a project, then an identifying relationship would exist between the entities PROJECT and TASK. That is, the associated project must be known in order to uniquely identify one task from all other tasks. (Also see Foreign Keys in Section 3.7)

If every instance of the child entity can be uniquely identified without knowing the associated instance of the parent entity then the relationship is referred to as a "non-identifying relationship". For example, although an existence-dependency relationship may exist between the entities BUYER and PURCHASE-ORDER, purchase orders may be uniquely identified by a purchase order number without identifying the associated buyer.

Assertions which affect multiple relationships may also be defined. One type of assertion may specify a boolean constraint between two or more relationships. For example, an "exclusive OR" constraint states that for a given parent entity

instance if one type of child entity instance exists, then a second type of child entity instance will not exist. However, if both the parent and child entities refer to the same real world thing, then a potential categorization relationship exists (See Section 3.3).

Another type of constraint is a "path assertion" which constraints the specific instances of parent and child entities when two entities can be related either directly or indirectly through two different sequences of relationships. For example, the entity DEPARTMENT may have two child entities, EMPLOYEE and PROJECT. If the entities EMPLOYEE and PROJECT have a common child entity called PROJECT-ASSIGNMENT, then PROJECT-ASSIGNMENT is indirectly related to DEPARTMENT via two different relationship paths. A path assertion might state that "employees may only be assigned to projects which belong to the same department for which they work".

Connection Relationship Syntax

A specific connection relationship is depicted as a line drawn between the parent entity and the child entity with a dot at the child end of the line. The default child cardinality is zero, one or many. A "P" (for positive) is placed beside the dot to indicate a cardinality of one or more. A "Z" is placed beside the dot to indicate a cardinality of zero or one. If the cardinality is an exact number, a positive integer number is placed beside the dot. See Figure 3-2.

A solid line depicts an identifying relationship between the parent and child entities. See Figure 3-3. If an identifying relationship exists the child entity is always an identifier-dependent entity, represented by a rounded corner box, and the primary key attributes of the parent entity are also inherited primary key attributes of the child entity. (Also see Foreign Keys in Section 3.7).

The parent entity in an identifying relationship will be identifier-independent unless the parent entity is also the child entity in some other identifying relationship, in which case both the parent and child entity would be identifier-dependent. An entity may have any number of relationships with other entities. However, if the entity is a child entity in any identifying relationship, it is always shown as a identifier-dependent entity with rounded corners, regardless of its role in the other relationships.

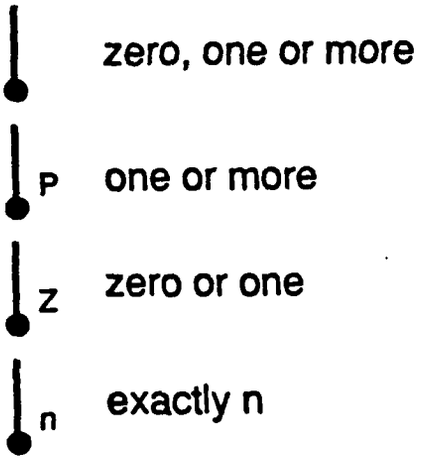
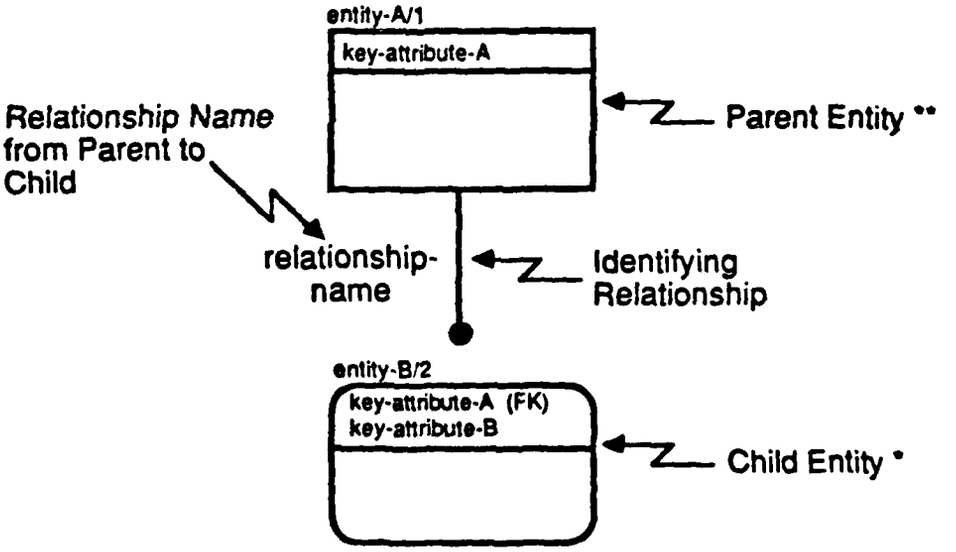


Figure 3-2. Relationship Cardinality Syntax



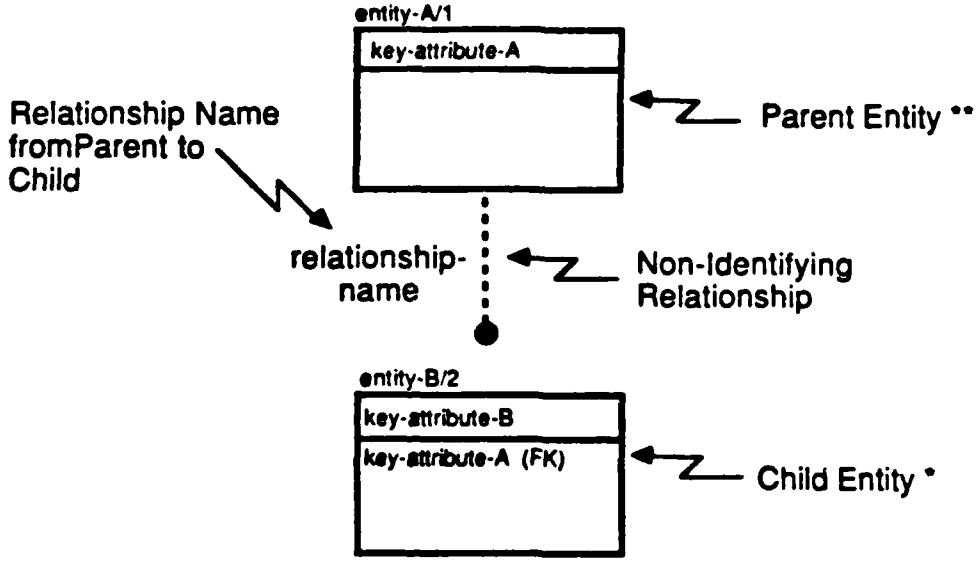
* The Child Entity in an Identifying Relationship is always an Identifier-Dependent Entity

** The Parent Entity in an Identifying Relationship may be an Identifier-Independent Entity (as shown) or an Identifier-Dependent Entity depending upon other relationships.

Figure 3-3. Identifying Relationship Syntax

A dashed line depicts a non-identifying relationship between the parent and child entities. See Figure 3-4. Both parent and child entities will be identifier-independent entities in a non-identifying relationship unless either or both are child entities in some other relationship which is an identifying relationship.

A relationship is given a name, expressed as a verb phrase (a verb with optional adverbs and prepositions) placed beside the relationship line. The name of each relationship between the same two entities must be unique, but the relationship names need not be unique within the model. The relationship name is always expressed in the parent-to-child direction, such that a sentence can be formed by combining the parent entity name, relationship name, cardinality expression, and child entity name. For example, the statement "A project consists of



* The Child Entity in a Non-Identifying Relationship will be an Identifier-Dependent Entity unless the entity is also a Child Entity in some Identifying Relationship.

** The Parent Entity in a Non-Identifying Relationship may be an Identifier-Independent Entity (as shown) or an Identifier-Dependent Entity depending upon other relationships.

Figure 3-4. Non-Identifying Relationship Syntax

one or more tasks" could be derived from a relationship showing PROJECT as the parent entity, TASK as the child entity with a "P" cardinality symbol, and "CONSISTS OF" as the relationship name. Note that the relationship must still hold true when stated from the reverse direction, although the child to-parent relationship is not named explicitly. From the previous example, it is inferred that "a task is part of exactly one project".

Connection Relationship Rules

1. A specific connection relationship is always between exactly two entities, a parent entity and a child entity.

2. An instance of a child entity must always be associated with exactly one instance of the parent entity.

3. An instance of a parent entity may be associated with zero, one or more instances of the child entity depending on the specified cardinality.

4. The child entity in an identifying relationship is always an identifier-dependent entity.

5. An entity may be associated with any number of other entities as either a child or a parent.

3.3 Categorization Relationships

Categorization Relationship Semantics

Entities are used to represent the notion of "things about which we need information". Since some real world things are categories of other real world things, some entities must, in some sense, be categories of other entities. For example, suppose employees are something about which information is needed. Although there is some information needed about all employees, additional information may be needed about salaried employees which is different from the additional information needed about hourly employees. Therefore, the entities SALARIED-EMPLOYEES and HOURLY-EMPLOYEES are categories of the entity EMPLOYEE. In an IDEF1X model, they are related to one another through a categorization relationship.

A "complete categorization relationship" is a relationship between two or more entities, in which each instance of one entity, referred to as the generic entity, is associated with

exactly one instance of one and only one of the other entities, referred to as catgeory entities. Each instance of the generic entity and its associated instance of one of the category entities represents the same real-world thing and, therefore, have the same unique identifier. From the previous example, EMPLOYEE is the generic entity and SALARIED-EMPLOYEE and HOURLY-EMPLOYEE are category entities.

Category entities for a generic entity are always mutually exclusive. That is, an instance of the generic entity can correspond to the instance of only one category entity. This implies from the example that an employee cannot be both salaried and hourly. The IDEF1X syntax does allow, however, for an incomplete set of categories. If it is possible that an instance of the generic entity is not associated with any of the category entities, then the relationship is defined as an "incomplete categorization relationship".

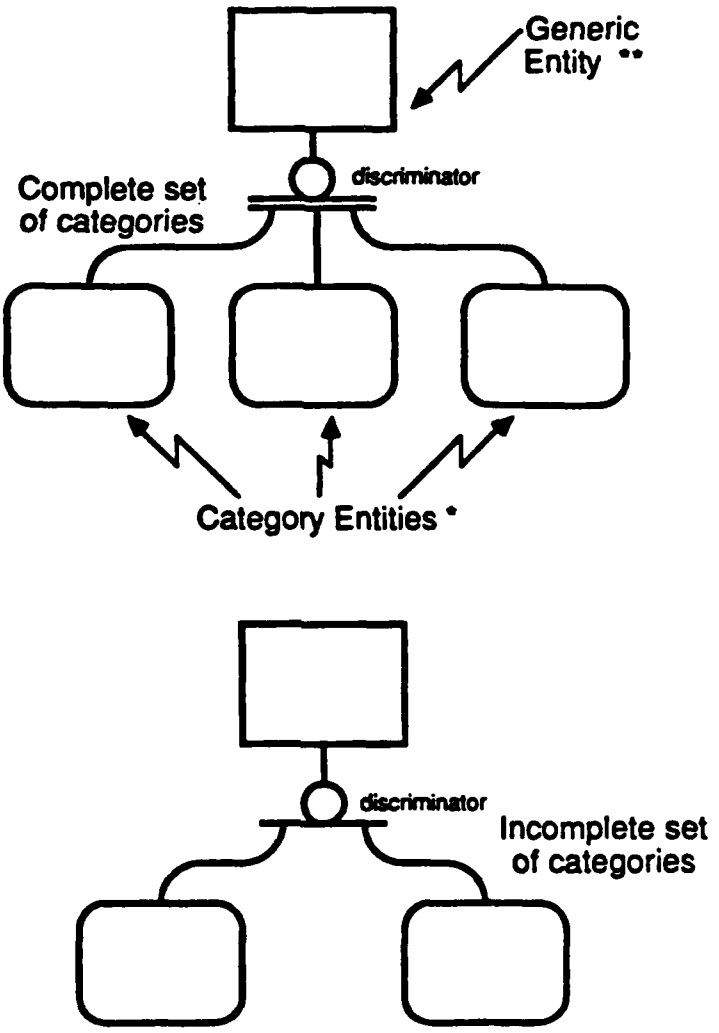
An attribute value in the generic entity instance determines to which of the possible category entities it is related. This attribute is called the "discriminator" of the categorization relationship. In the previous example, the discriminator might be named EMPLOYEE-TYPE.

Categorization Relationship Syntax

A categorization relationship is shown as a line extending from the generic entity to a circle which is underlined. Separate lines extend from the underlined circle to each of the category entities. Cardinality is not specified for the category entity since it is always zero or one. Category entities are also always identifier-dependent. See Figure 3-5. The generic entity is independent unless its identifier is inherited through some other relationship.

If the circle has a double underline, it indicates that the set of category entities is complete. A single line under the circle indicates an incomplete set of categories.

The name of the generic entity attribute used as the discriminator is written beside the circle. Although the relationship itself is not named explicitly, the generic entity to category entity relationship can be read as "can be". For example, an EMPLOYEE can be a SALARIED-EMPLOYEE. If the complete set of categories is referenced, the relationship may be read as "must be". For example, an EMPLOYEE must be a SALARIED-EMPLOYEE or an HOURLY-EMPLOYEE. The relationship is



* Category Entities will always be Identifier-Dependent Entities.

** The Generic Entity may be an Identifier-Independent Entity (as shown) or an Identifier-Dependent Entity depending upon other relationships.

Figure 3-5. Categoriztion Relationship Syntax

read as "is a/an" from the reverse direction. For example, an HOURLY-EMPLOYEE is an EMPLOYEE.

The generic entity and each category entity must have the same key attributes. However, role names may be used in the category entities. (Also, see Foreign Keys in Section 3.7)

Categorization Relationship Rules

1. A category entity can have only one generic entity. That is, it can only be a member of the set of categories for one categorization relationship.

2. A category entity in one categorization relationship may be a generic entity in another categorization relationship.

3. An entity may have any number of categorization relationships in which it is the generic entity. (For example, FEMALE-EMPLOYEE and MALE-EMPLOYEE may be a second set of categories for the generic entity EMPLOYEE.)

4. A category entity cannot be a child entity in an identifying connection relationship.

5. The primary key attribute(s) of a category entity must be the same as the primary key attribute(s) of the generic entity.

6. All instances of a category entity have the same discriminator value and all instances of different categories must have different discriminator values.

3.4 Non-Specific Relationships

Non-Specific Relationship Semantics

Both parent-child connection and categorization relationships are considered to be "specific relationships" because they defined precisely how instances of one entity relate to instances of another entity. In a fully refined IDEF1X model, all associations between entities must be expressed as specific relationships. However, in the initial development of a model, it is often helpful to identify "non-specific relationship" between two entities. These non-specific relationships are refined in later development phases of the model. The procedure for resolving non-specific relationships is discussed in Sec-

tion 4.4.1

A non-specific relationship, also referred to as a "many to many relationship", is an association between two entities in which each instance of the first entity is associated with zero, one, or many instances of the second entity and each instance of the second entity is associated with zero, one, or many instances of the first entity. For example, if an employee can be assigned to many projects and a project can have many employees assigned, then the connection between the entities EMPLOYEE and PROJECT can be expressed as a non-specific relationship. This non-specific relationship can be replaced with specific relationships later on in the model development by introducing a third entity, such as PROJECT-ASSIGNMENT, which is a common child entity in specific connection relationships with the EMPLOYEE and PROJECT entities. The new relationships would specify that an employee has zero, one, or more project assignments and that a project has zero, one or more project assignments. Each project assignment is for exactly one employee and exactly one project. Entities introduced to resolve non- specific relationship are sometimes called "intersection" or "associative" entities.

A non-specific relationship may be further defined by specifying the cardinality from both directions of the relationship. Any combination of cardinalities may be used to specify a non-specific relationship. That is, for each instance of the first entity, there are either:

- zero, one or more;
- one or more;
- zero or one; or
- an exact number

of instances of the second entity, and for each instance of the second entity there are either:

- zero, one or more;
- one or more;
- zero or one, or
- an exact number

of instances of the first entity. Note that if a cardinality of "exactly one" exists at either end of the relationship, the relationship is specific rather than non-specific.

Non-Specific Relationship Syntax

A non-specific relationship is depicted as a line drawn between the two associated entities with a dot at each end of the line. See Figure 3-6. Cardinality may be expressed at both ends of the relationship as shown in Figure 3-2. A "P"

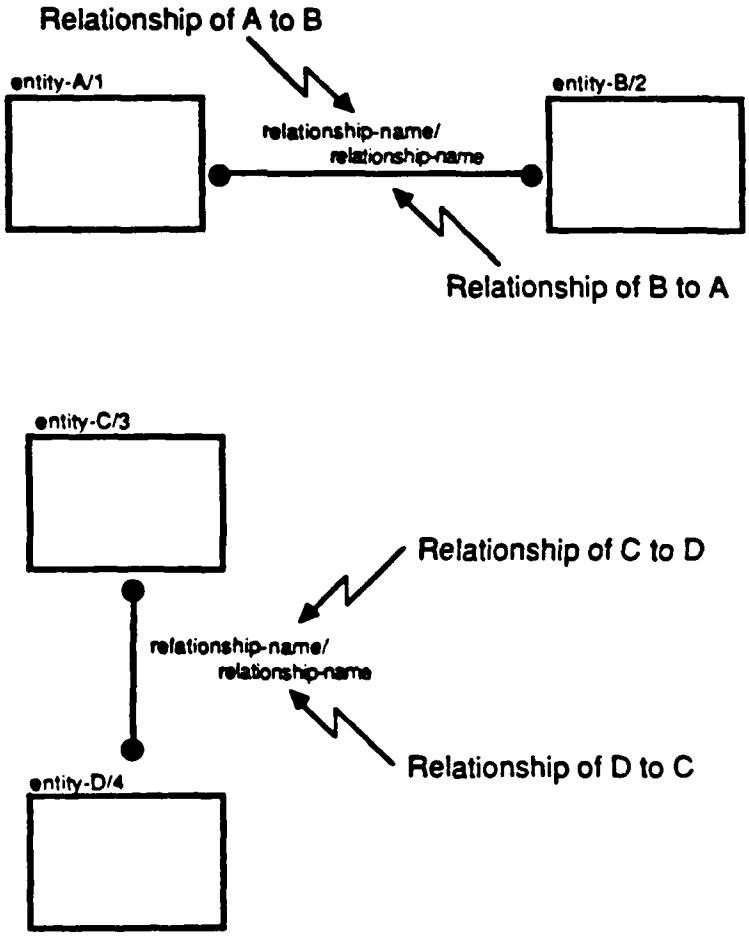


Figure 3-6. Non-Specific Relationship Syntax

(for positive) placed beside a dot indicates that for each instance of the entity at the other end of the relationship there are one or more instances of the entity at the end with the "P". A "Z" placed beside a dot indicates that for each instance of the entity at the other end of the relationship there are zero or one instances of the entity at the end with the "Z". In a similar fashion, a positive integer number or minimum and maximum positive integer range may be placed beside a dot to specify an exact cardinality. The default cardinality is zero, one, or more.

A non-specific relationship is named in both directions. The relationship names are expressed as a verb phrase (a verb with optional adverbs and prepositions) placed beside the relationship line and separated by a slash, "/". The order of the relationship names depends on the relative position of the entities. The first name expresses the relationship from either the left entity to the right entity, if the entities are arranged horizontally, or the top entity to the bottom entity, if they are arranged vertically. The second name expresses the relationship from the other direction, that is either the right entity to the left entity or the bottom entity to the top entity again depending on the orientation. The relationship is labeled such that sentences can be formed by combining the entity names with the relationship names. For example, the statements "A project has zero, one, or more employees" and "An employee is assigned zero, one, or more projects" can be derived from a non-specific relationship labeled "has/is assigned" between the entities PROJECT and EMPLOYEE. (The sequence assumes the entity PROJECT appears above or to the left of the entity EMPLOYEE.)

Non-Specific Relationship Rules

1. A non-specific relationship is always between exactly two entities.

2. An instance of either entity may be associated with zero, one or more instances of the other entity depending on the specified cardinality.

3. A non-specific relationship must be replaced by specific relationships in order to fully develop a model.

3.5 Attributes

Attribute Semantics

An "attribute" represents a type of characteristic or property associated with a set of real or abstract things (people, objects, places, events, states, ideas, pairs of things, etc.). An "attribute instance" is a specific characteristic of an individual member of the set. An attribute instance is defined by both the type of characteristic and its value, referred to as an "attribute value". Within an IDEF1X model, attributes are associated with specific entities. An instance of an entity, then, must have a single specific value for each associated attribute. For example, EMPLOYEE-NAME and BIRTH-DATE may be attributes associated with the entity EMPLOYEE. An instance of the entity EMPLOYEE could have the attribute values of "Jenny Lynne" and "February 27, 1953".

An entity must have an attribute or combination of attributes whose values uniquely identify every instance of the entity. These attributes form the "primary-key" of the entity. (See Section 3.6). For example, the attribute EMPLOYEE-NUMBER might serve as the primary key for the entity EMPLOYEE, while the attributes EMPLOYEE-NAME and BIRTH-DATE would be other attributes.

Within an IDEF1X model, every attribute is owned by only one entity and every instance of the entity must have a value for every attribute associated with the entity. That is, the attribute must be applicable to every member of the set of things represented by the entity. The attribute MONTHLY-SALARY, for example, would apply to some instances of the entity EMPLOYEE but probably not all. Therefore, a separate but related entity called SALARIED-EMPLOYEE might be identified in order to establish ownership for the attribute MONTHLY-SALARY. Since an actual employee who was salaried would represent an instance of both the EMPLOYEE and SALARIED-EMPLOYEE entities, attributes common to all employees, such as EMPLOYEE-NAME and BIRTH-DATE, need not be an attribute of the SALARIED-EMPLOYEE entity.

In addition to attributes "owned" by the entity, that is a basic characteristic of the things the entity represents, an attribute may be "inherited" by the entity through a specific connection or categorization relationship in which it is a child or category entity. (See Section 3.7). For example, if every employee is assigned to a department, then the attribute

DEPARTMENT-NUMBER could be an attribute of EMPLOYEE which is inherited through the relationship of the entity EMPLOYEE to the entity DEPARTMENT. The entity DEPARTMENT would be the owner of the attribute DEPARTMENT-NUMBER. Only primary key attributes may be inherited through a relationship. The attribute DEPARTMENT-NAME, for example, would not be an inherited attribute of EMPLOYEE if it was not part of the primary key for the entity DEPARTMENT.

Attribute Syntax

Each attribute is identified by a unique name expressed as a noun phrase (a noun with optional adjectives and prepositions) that describes the characteristic represented by the attribute. The noun phrase is singular, not plural. Abbreviations and acronyms are permitted, however, the attribute name must be meaningful and consistent throughout the model. A formal definition of the attribute and a list of synonyms or aliases must be defined in the model of glossary.

Attributes are shown by listing their names, one line per attribute, inside the associated entity box. Attributes which define the primary key are placed at the top of the list and separated from the other attributes by a horizontal line. See Figure 3-7.

Attribute Rules

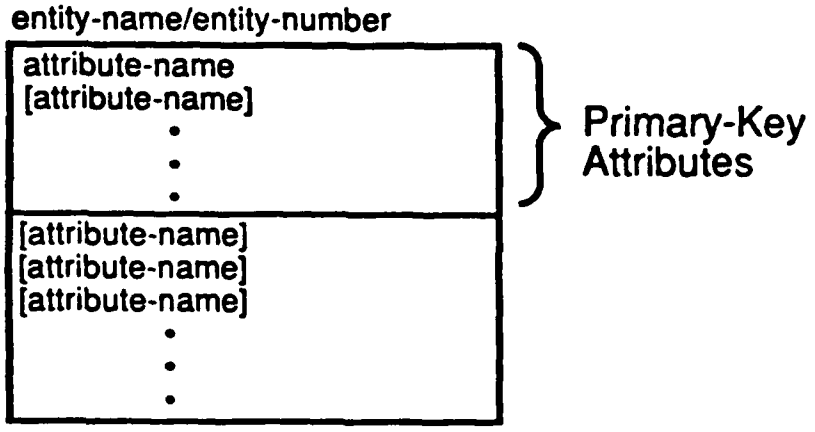
1. Each attribute must have a unique name and the same meaning must always apply to the same name. Furthermore, the same meaning cannot apply to different names unless the names are aliases.

2. An entity can own any number of attributes. Every attribute is owned by exactly one entity (referred to as the Single-Owner Rule).

3. An entity can have any number of inherited attributes. However, an inherited attribute must be part of the primary key of a related parent entity or generic entity.

4. Every instance of an entity must have a value for every attribute (referred to as the No-Null Rule).

5. No instance of an entity can have more than one value for an attribute associated with the entity (referred to as the No-Repeat Rule).



EXAMPLE

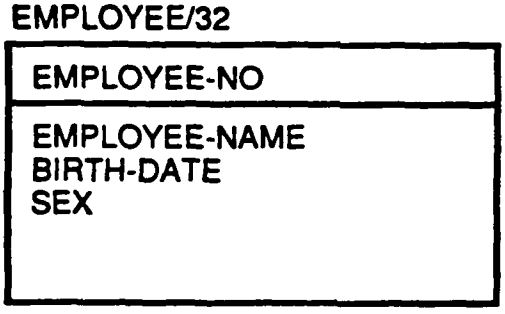


Figure 3-7. Attribute and Primary Key Syntax

3.6 Primary and Alternate Keys

Primary and Alternate Key Semantics

A candidate key of an entity is one or more attributes, whose value uniquely identifies every instance of the entity. For example, the attribute PURCHASE-ORDER-NUMBER may uniquely identify an instance of the entity PURCHASE-ORDER. A combination of the attributes ACCOUNT-NUMBER and CHECK-NUMBER may uniquely identify an instance of the entity CHECK.

Every entity must have at least one candidate key. In some cases, an entity may have more than one attribute or group of attributes which uniquely identify instances of the entity. For example, the attributes EMPLOYEE-NUMBER and SOCIAL-SECURITY-NUMBER may both uniquely identify an instance of the entity EMPLOYEE. If more than one candidate key exists, then one candidate key is designated as the "primary key" and the other candidate keys are designated as "alternate keys". If only one candidate key exists, then it is, of course, the primary key.

Primary and Alternate Key Syntax

Attributes which define the primary key are placed at the top of the attribute list within an entity box and separated from the other attributes by a horizontal line. See Figure 3-7.

Each alternate key is assigned a unique integer number and is shown by placing the note "AK" plus the alternate key number in parentheses, e.g. "(AK1)", to the right of each of the attributes in the key. (See Figure 3-8). An individual attribute may be identified as part of more than one alternate key. A primary key attribute may also serve as part of an alternate key.

Primary and Alternate Key Rules

1. Every entity must have a primary key.

2. Any entity may have any number of alternate keys.

3. A primary or alternate key may consist of a single attribute or combination of attributes.

4. An individual attribute may be part of more than one key, either primary or alternate.

5. Attributes which form primary and alternate keys of an entity may be either owned by the entity or inherited through a relationship. (See Foreign Keys in Section 3.7).

6. Primary and alternate keys must contain only those attributes that contribute to unique identification (i.e., if any attribute were not included as part of the key then every instance of the entity could not be uniquely identi-

fied, referred to as the Smallest-Key Rule).

7. If the primary key is composed of more than one attribute, the value of every non-key attribute must be functionally dependent upon the entire primary key, i.e., if the primary key is known, the value of each non-key attribute is known and no non-key attribute value can be determined by just part of the primary key (referred to as the Full-Functional-Dependency Rule).

8. Every non-key attribute must be only functionally dependent upon the primary and alternate keys, i.e., no non-key attribute's value can be determined by another non-key attribute value (referred to as the No-Transitive-Dependency Rule).

attribute-name (AKn [,AKm] ...)

Where n,m,etc., uniquely identify each Alternate Key that includes the associated attribute and where an Alternate Key consists of all the attributes with the same identifier.

EXAMPLE

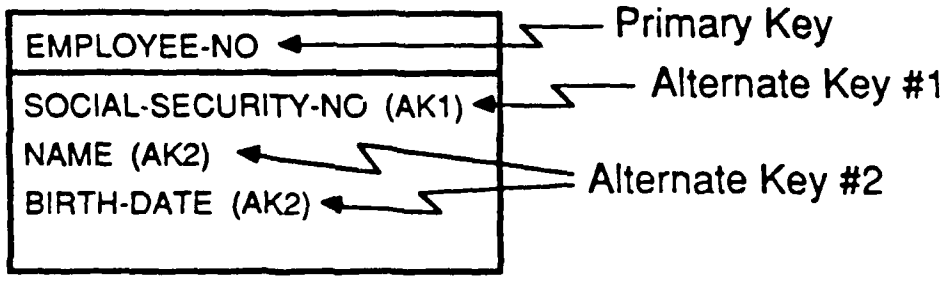


Figure 3-8. Alternate Key Syntax

3.7 Foreign Keys

Foreign Key Semantics

If a specific connection or categorization relationship exists between two entities, then the attributes which form the primary key of the parent or generic entity are inherited as attributes of the child or category entity. These inherited attributes are referred to as "Foreign Keys". For example, if a connection relationship exists between the entity PROJECT as a parent and the entity TASK as a child, then the primary key attributes of PROJECT would be inherited attributes of the entity TASK. For example, if the attribute PROJECT-ID were the primary key of PROJECT, then PROJECT-ID would also be an inherited attribute or Foreign Key of TASK.

An inherited attribute may be used as either a portion or total primary key, alternate key, or non-key attribute within an entity. If all the primary key attributes of a parent entity are inherited as part of the primary key of the child entity, then the relationship through which the attributes were inherited is an "identifying relationship". If any of the inherited attributes are not part of the primary key, then the relationship is a "non-identifying relationship. See Section 3.2. For example, if tasks were only uniquely numbered within a project, then the inherited attribute PROJECT-ID would be combined with the owned attribute TASK-NUMBER to define the primary key of TASK. The entity PROJECT would have an identifying relationship with the entity TASK. If on the other hand, the attribute TASK-NUMBER is always unique, even between projects, then the inherited attribute PROJECT-ID would be a non-key attribute of the entity TASK. In this case, the entity PROJECT would have a non-identifying relationship with the entity TASK.

In a categorization relationship, both the generic entity and the category entities represent the same real-world thing. Therefore, the primary key for all category entities is inherited through the categorization relationship from the primary key of the generic entity. For example, if SALARIED-EMPLOYEE and HOURLY-EMPLOYEE are category entities and EMPLOYEE is the generic entity, then if the attribute EMPLOYEE-NUMBER is the primary key for the entity EMPLOYEE, it would also be the primary key for the entities SALARIED-EMPLOYEE and HOURLY-EMPLOYEE.

In some cases, a child entity may have multiple relationships to the same parent entity. The primary key of the parent entity would appear as inherited attributes in the child entity for each relationship. For a given instance of the child entity, the value of the inherited attributes may be different for each relationship, i.e. two different instances of the parent entity may be referenced. A bill of material structure, for example, can be represented by two entities PART and ASSEMBLE-STRUCTURE. The entity PART has a dual relationship as a parent entity to the entity ASSEMBLE-STRUCTURE. The same part sometimes acts a component from which assemblies are made, i.e., a part may be a component in one or more assemblies, and sometimes acts as an assembly into which components are assembled, i.e., a part may be an assembly for one or more component parts. If the primary key for the entity PART is PART-NO, then PART-NO would appear twice in the entity ASSEMBLE-STRUCTURE as an inherited attribute.

When a single attribute is inherited more than once, a "role name" must be assigned to each occurrence. From the previous example, role names of COMPONENT-NO and ASSEMBLE-NO could be assigned to distinguish between the two inherited PART-NO attributes. Although not required, role names may also be used with single occurrences of inherited attributes to more precisely convey its meaning within the context of the child entity.

Foreign Key Syntax

A foreign key is shown by placing the names of the inherited attributes inside the entity box and by following each with the letters "FK" in parentheses, i.e., "(FK)". See Figure 3-9. If the inherited attribute belongs to the primary key of the child entity, it is placed above the horizontal line and the entity is drawn with rounded corners to indicate that the identifier (primary key) of entity is dependent upon an attribute inherited through a relationship. If the inherited attribute does not belong to the primary key of the child entity, it is drawn below the line. Inherited attributes may also be part of an alternate key.

Role names, like attribute names, are noun phrases. A role name is followed by the name of the inherited attribute, separated by a period. See Figure 3-10.

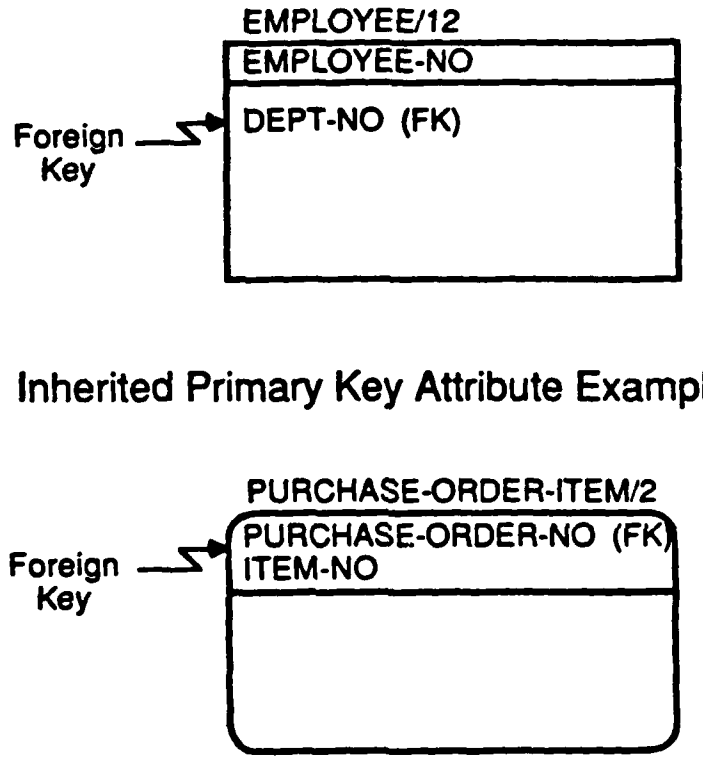


Figure 3-9. Foreign Key Syntax Examples

Foreign Key Rules

1. Every entity must contain a separate foreign key for each specific connection or categorization relationship in which it is the child or category entity.

2. The primary key of a generic entity must be inherited as the primary key for each category entity.

3. An entity must not contain two entire foreign keys that identify the same instance of the same parent or generic entity for every instance of the child or category entity (otherwise, only one relationship exists and only one foreign key is needed).

ROLE NAME SYNTAX

role-name . attribute-name (FK)

EXAMPLE

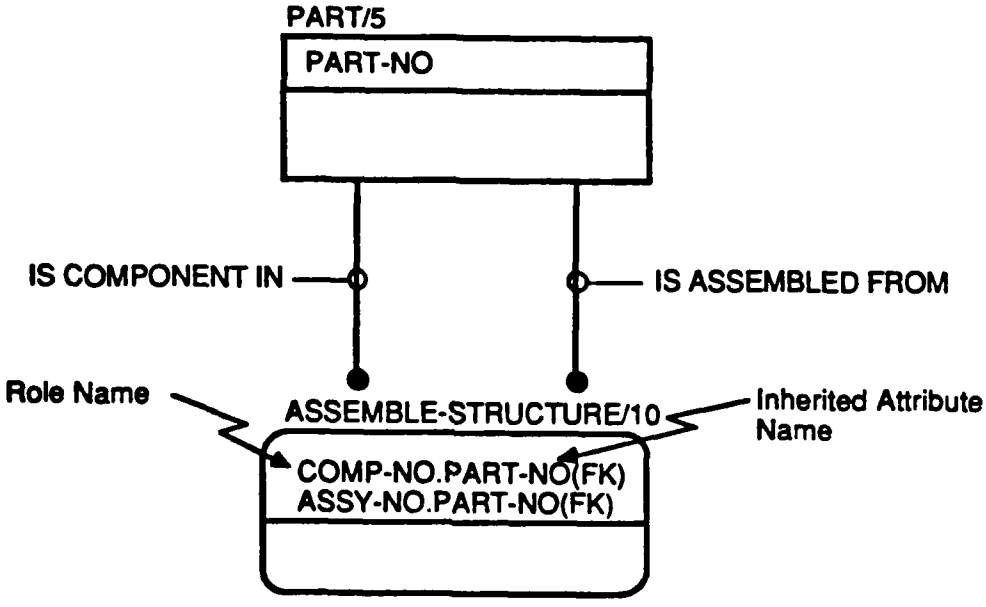


Figure 3-10. Role Name Syntax

4. Every inherited attribute of a child or category entity must represent an attribute in the primary key of a related parent or generic entity. Conversely, every primary key attribute of a parent or generic entity must be an inherited attribute in the related child or category entity.

5. Each role name assigned to an inherited attribute must be unique and the same meaning must always apply to the same name. Furthermore, the same meaning cannot apply to different names unless the names are aliases.

6. A single inherited attribute may be part of more than one foreign key provided that the attribute always has the same value for both foreign keys in any given instance of the entity.

SECTION 4

MODELING PROCEDURES

4.1 Phase Zero - Project Initiation

The IDEF1X data model must be described and defined in terms of both its limitations and its ambitions. The modeler is one of the primary influences in the development of the scope of the model. Together, the modeler and the project manager unfold the plan for reaching the objectives of Phase Zero. These objectives include:

- Project definition - a general statement of what has to be done, why, and how it will get done.
- Source material - a plan for the acquisition of source material, including indexing and filing.
- Author conventions - a fundamental declaration of the conventions (optional methods) by which the author chooses to make and manage the model.

The products of these objectives, coupled with other descriptive and explanatory information, become the products of the Phase Zero effort.

4.1.1 Establish Modeling Objectives

The modeling objective is comprised of two statements:

- Statement of purpose - a statement defining concerns of the model, i.e., its contextual limits.
- Statement of scope - a statement expressing the functional boundaries of the model.

One of the primary concerns, which will be answered as a result of the establishment of the modeling objective, is the concern over the time-frame reference for the model. Will it be a model of the current activities (i.e., an AS-IS model) or will it be a model of what is intended after future changes are made (i.e., a TO-BE model)? Formal description of a problem domain for an IDEF1X modeling project may include the review, construction, modification, and/or elaboration of one or more IDEF0 (activity) models. For this reason, both the modeler and

the project manager must be versed to some degree in the authorship and use of IDEF0 models. Typically, an IDEF0 model already exists, which can serve as a basis for the problem domain.

Although the intent behind data modeling is to establish an unbiased view of the underlying data infrastructure which supports the entire enterprise, it is important for each model to have an established scope which helps identify the specific data of interest. This scope may be related to a type of user (e.g. a buyer or design engineer) a business function (e.g. engineering drawing release or shop order scheduling) or a type of data (e.g. geometric product definition data or financial data). The statement of scope together with the statement of purpose defines the modeling objective. The following is an example of a modeling objective:

> "The purpose of this model is to define the current (AS-IS) data used by a manufacturing cell supervisor to manufacture and test composite aircraft parts."

Although the scope may be limited to a single type of user, other users must be involved in the modeling process to ensure development of an unbiased view.

4.1.2 Develop Modeling Plan

The modeling plan outlines the tasks to be accomplished and the sequence in which they should be accomplished. These are laid out in conformance with the overall tasks of the modeling effort:

- Project planning
- Data collection
- Entity definition
- Relationship definition
- Key attribute definition
- Nonkey attribute population
- Model validation
- Acceptance review

The modeling plan serves as a basis to assign tasks, schedule milestones, and estimate cost for the modeling effort.

4.1.3 Organize Team

The value of a model is measured not against some absolute

norm, but in terms of its acceptability to experts and laymen within the community for which it is built. This is accomplished through two mechanisms. First, a constant review by experts of the evolving model provides a measure of validity of that model within the particular environment of those experts. Second, a periodic review of the model by a committee of experts and laymen provides for a "corporate" consensus to the model. During the modeling process, it is not uncommon to discover inconsistencies in the way various departments do business. These inconsistencies must be resolved in order to produce data models that represent the enterprise in an acceptable and integrated fashion.

To the extent possible, the builders of a model should be held responsible for what the model says. Nothing is assumed to have been left to the model reader's imagination. Nor is the reader at liberty to draw conclusions outside the scope of the statement of the model. This forces a modeler to carefully consider each piece of data added to the model, so that no imagination is required in the interpretation of the model.

The team organization is constructed to support these basic principles and to provide required project controls. The IDEF1X team organization has five primary roles:

- Project Manager
- Modeler
- Sources of Information
- Subject Matter Experts
- Acceptance Review Committee

The purpose of a role assignment, irrespective of the assignee, is the determination of responsibility. Each of these roles is defined on the pages that follow.

One person may serve in more than one capacity on the team, but it is wise to remember that if there are insufficient points of view taken into account when building the model, the model may represent a very narrow perspective. It may end up only partially serving to reach the objectives of the modeling project.

In the cases of the project manager and the modeler, there must be a lead, or principal, individual who fulfills the role. Although it is the modeler's ultimate goal to have the model approved by the review committee, the modeler reports to the project manager, not the review committee. In this way the

otherwise conflicting interests of the modeler, review committee, and project manager are disentangled. The project manager is always placed in a position of control, but the various technical discussions and approvals are automatically delegated to the qualified participants. Figure 4-1 illustrates the functional project organization, with the project manager at the nucleus of all project activity.

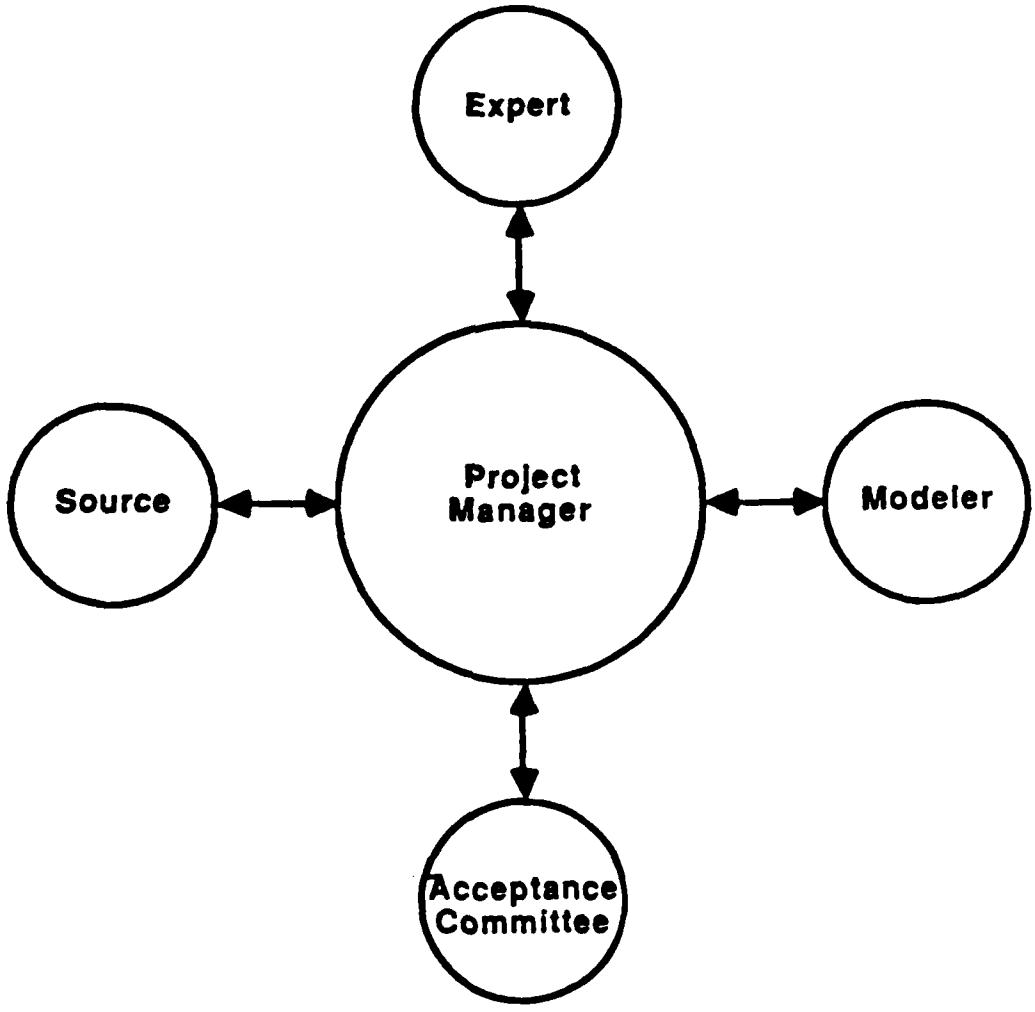


Figure 4-1. Team Organization

Project Manager Role

The project manager is the person identified as having administrative control over the modeling project. The project manager performs four essential functions in the modeling effort.

First of all, the project manager selects the modeler. As a major part of this function, the project manager and the modeler must reach an agreement on the ground rules to be followedin the modeling effort. These include the use of this methodology, the extent of control the project manager expects to exercise over the modeler, and the scope and orientation of the model to be developed.

The second function performed by the project manager is to identify the sources of information on which the modeler will draw to build the model. These sources may either be people particularly knowledgeable in some aspect of the business area, or documents that record, instigate, or report aspects of that business area. From a modeling standpoint, personnel who can interpret and explain the information they deal with are more desirable. However, documents which record that information are usually less expensive to obtain. The project manager must be in a position to provide these sources to the modelers. Sources are initially identified in modeling Phase Zero, but the list must be reviewed and revised as the effort progresses, since the information required will tend to change as the model grows.

Next, the project manager selects experts on whose knowledge and understanding the modeler will draw for validation of the evolving model. Validation, as discussed below under Expert, means concurrence that the model acceptably reflects the subject being modeled. The experts will be given portions of the model and asked to review and comment based on their particular knowledge. Clearly, more of an expert's time will be absorbed in the modeling effort than the time we would set aside for a source of basic information. The initial list of experts is established during Phase Zero, but will be reviewed and revised throughout the modeling effort as the need arises.

Finally, the project manager forms and convenes the acceptance review committee. This committee, under the chairmanship of the project manager, periodically meets to consider issues of substance requiring arbitration and to review portions of the model for formal acceptance. The project manager sits on

the committee as its non-voting chairman, thereby providing the needed link between the modeler and the committee. Although the modeler is not a member of the committee, the project manager will frequently invite the modeler to attend a committee meeting to provide background information or to explain difficult technical points. The first meeting of the committee is held during Phase Zero, and thereafter at the discretion of the project manager.

Modeler Role

The modeler records the model on the basis of source material he is able to gather. It is the modeler's function to apply modeling techniques to the problem posed by the project manager. The modeler performs four primary functions: source data collection, education and training, model recording, and model control. The modeler is the central clearinghouse for both modeling methodology information and information about the model itself.

Before the modeler's primary functions begin, the modeler and the project manager study and establish the scope of the modeling effort. The modeler then outlines a project plan, i.e., the tasks required to reach the stated objectives. The project manager provides the modeler with a list of information sources and a list of experts on whom the modeler may rely. The modeler must ensure that the necessary lines of communication are established with all participants.

Source data are collected by the modeler from the various sources identified by the project manager. The nature of these data will depend largely on the modeling phase being exercised. Both people and documents will serve as sources of information throughout the modeling effort. The modeler must be particularly aware that each piece of source data represents a particular view of the data in the enterprise. Each producer and each user of data has a distinct view of that data. The modeler is striving to see, through the eyes of the sources, the underlying meaning and structure of the data. Each source provides a perspective, a view of the data sought. By combining these views, by comparing and contrasting the various perspectives, the modeler develops an image of the underlying reality. Each document may be seen as a microcosmic implementation of a system, meeting the rules of the underlying data model. The modeler attempts to capture all of these rules and represent them in a way that can be read, understood, and agreed upon by experts and informed laymen.

The modeler's second function is to provide assistance with the modeling technique to those who may require it. This will fall generally into three categories: general orientation for review committee members, sources, and some experts; model readership skills for some sources and experts; and modeling skills for some experts and modelers, as required.

The third function is recording the model. The modeler records the data model by means of textual and graphic descriptions. Standard forms for capturing and displaying model information are presented in Section 5.3.

The modeler also controls the development of the model. Files of derived source information are maintained to provide appropriate backup for decisions made by the modeler, and to allow a record of participation. This record of participation provides the modeler with an indication of the degree to which the anticipated scope is being covered. By knowing who has provided information in what areas, and the quality of those interactions, the modeler can estimate the degree to which current modeling efforts have been effective in meeting the original goals.

The modeler is also responsible for periodically organizing the content of the model into some number of reader kits for distribution to reviewers. A reader kit is a collection of information about the model, organized to facilitate its review and the collection of comments from the information experts. Kits are discussed further in Section 5.2.

Source Roles

Source information for an IDEF1X model comes from every quarter within the enterprise. These sources are often people who have a particular knowledge of the management or operation of some business process and whose contact with the model may be limited to a few short minutes of interview time. Yet these sources form the heart of the modeling process. Their contribution is modeled, and their perception provides the modeler with the needed insight to construct a valid, useful model. Sources must be sought out and used to best advantage wherever they may be found.

The project manager identifies sources of information that may be effective, based on the modeler's statement of need. As the modeling effort progresses, needs change and the list of

sources must be revised. Whereas the modeler must be careful to account for the information provided by each source, both the modeler and source should be aware that any particular contribution is necessarily biased. Each source perceives the world a little differently, and it is the modeler's responsibility to sort out these varying views. This is especially true of source documents.

Documents record the state of a minute portion of the enterprise at some point in time. However, the information on a document is arranged for the convenience of its users, and seldom directly reflects the underlying data structure. Redundancy of data is the most common example of this, but the occurrence of serendipitous data on a document is also a source of frequent and frustrating confusion. Documents are valuable sources of information for the model, but they require a great deal of interpretation, understanding, and corroboration to be used effectively.

If the data model is being developed to either integrate or replace existing databases, then the existing database designs should be referenced as a source document. However, like other documents, existing database designs do not generally reflect the underlying data structure and require interpretation.

People used as sources, on the other hand, can often extend themselves beyond their direct use of information to tell the modeler how that information is derived, interpreted, or used. By asking appropriate questions, the modeler can use this information to assist in understanding how the perception of one source may relate to that of another source.

Expert Role

An expert is a person appointed by the project manager who has a particular knowledge of some aspect of the manufacturing area being modeled, and whose expertise will allow valuable critical comments of the progressing model. The impact that appropriate experts can have on the modeling effort cannot be overemphasized. Both the modeler and the project manager should seriously consider the selection of each expert.

Experts are called on to critically review portions of the evolving model. This is accomplished through the exercise of some number of validation cycles, and by the use of reader kits. These kits provide the expert with a related collection of information presented to tell a story. In this fashion, the

expert is provided the information in an easily digestible form and is challenged to fill in the blanks or complete the story. Although the kit is largely based on modeler interpretation of information from informed sources, the comments of experts may also be expected to provide high quality source material for the refinement of the model. The particular expertise of these people makes them uniquely qualified to assist the modeler in constructing and refining the model. The modeler must take every opportunity to solicit such input, and this is why the kits of information must present the expert with concise, clear problems to solve relative to the modeling effort.

The primary job of the expert is to validate the model. Expert validation is the principal means of achieving an informed consensus of experts. That is, a valid model is one agreed to by experts informed about the model. Note that it is not necessary for a model to be "right" for it to be valid. If the majority of experts in the field agree that the model appropriately and completely represents the area of concern, then the model is considered to be valid. Dissenting opinions are always noted, and it is assumed by the discipline that models are invalid until proved otherwise. This is why expert participation is so vital to the modeling effort. When the modeler first constructs a portion of the model, he is saying, "I have reviewed the facts and concluded the following..." When that portion is submitted to experts for review, he asks, "Am I right?" Expert comments are then taken into account in revising that portion of the model with which the experts do not agree, always bearing in mind that a consensus is being sought.

Experts, more than any other nonmodeling participants, require training to be effective. In fact, one of the modeler's responsibilities is to ensure that experts have an adequate understanding of the modeling methodology and process. Principally, experts require good model readership skills, but it may be helpful to train an expert in some of the rudiments of model authorship. By providing experts with a basic understanding of modeling, the project is assured of useful input from those experts. Further, the stepwise, incremental nature of the modeling process presents experts with the modeling methodology in small doses. This tends to enhance the expert's ability to understand and contribute to the modeling effort.

Acceptance Review Committee Role

The acceptance review committee is formed of experts and

informed laymen in the area addressed by the modeling effort. The project manager forms the committee and sits as its chairman. It is the function of the review committee to provide guidance and arbitration in the modeling effort, and to pass final judgement over the ultimate product of the effort: an IDEF1X data model. Since this model is one part of a complex series of events to determine and implement systematic improvements in the productivity of the enterprise, it is important that the committee include ample representation from providers, processors, and end users of the data represented. Very often, this will mean that policy planners and data processing experts will be included on the committee. These people are primarily concerned with eventual uses to which the model will be put. Further, it may be advantageous to include experts from business areas outside of, but related to, the area under study. These experts often can contribute valuable insight into how the data model will affect, or be affected by, ongoing work in other areas.

It is not uncommon for those who serve as experts to also serve as members of the review committee. No conflict of interest, in fact, should be anticipated. An expert is often only exposed to restricted portions of the model at various intermediate stages. The review committee, by contrast, must pass judgment on the entire model. It is much less common for individuals who serve in the role of source to also sit on the committee, as their knowledge is usually restricted enough in coverage to exclude them from practical contribution to the committee. Finally, it is ill-advised for modelers to sit on the committee, as a severe conflict of interest is clearly evident. Further, the role of modeler is to record the model without bias, and the role of the committee is to ensure that the model in fact represents their particular enterprise.

The end product of this segment of the project definition is the documentation of specific assignments made by the project manager to fulfill each of the functional role requirements of the modeling technique.

4.1.4 Collect Source Material

One of the first problems confronting the modeler is to determine of what sort of material needs to be gathered, and from what sources is should be gathered. Not infrequently, the scope and context of the IDEF1X model will be determined based on an analysis of an IDEF0 function model. Once the analysis of the functions and pipelines between functions is completed,

target functions within the enterprise represented by the function model can be identified. A target function node is one that represents a concentration of information in use, which is representative of the problem domain.

Once the target functional areas have been identified and the primary information categories of interest selected, individuals within functions can be selected to participate in the data gathering process. This data gathering can be accomplished in several ways, including interviews with knowledgeable individuals; observation of activities, evaluation of documents, policies and procedures, and application specific information models, etc. This requires translation of the target function nodes into their equivalent, or contributing, modeling participants. Once the groups participating in a target function have been identified, the project manager can proceed to identify individuals or specific observable areas that can be used as sources of material for the model.

Source material may take a variety of forms and may be fairly widespread throughout an organization. Source materials may include:

- Interview results
- Observation results
- Policies and procedures
- Outputs of existing systems (reports and screens)
- Inputs to existing systems (data entry forms and screens)
- Database/file specifications for existing systems

Regardless of the method used, the objective of the modeler at this point is to establish a plan for the collection of representative documentation reflecting the information pertinent to the purpose and viewpoint of the model. Once collected, each piece of this documentation should be marked in such a way that could be traced to its source. This documentation, along with the added documentation that is discovered through the course of the modeling, will constantly be referred to in the various phases of model development. The modeler will study and search for source material that lends credibility to the basic structural characteristics of the model and to

the meaning of the data represented.

As discussed in Section 2, the objective of data modeling is to define a single consistent enterprise view of the data resource which is referred to as the Conceptual Schema in the ANSI/SPARC architecture. Source documents, for the most part, represent either External Schema or Internal Schema which must map to the Conceptual Schema but are biased toward their particular use. User reports, for example, are an External Schema view of the data which might serve as source documentation. File descriptions and database designs represent Internal Schema views of data and may also be used as source documentation. Although the data structure will be greatly simplified through the modeling process, the resulting data model must be mappable back to the External and Internal Schema from which it was developed.

A sound data collection plan is of paramount importance to accomplish the objective successfully. This data collection plan must reflect what kind of data is of importance, where that data is available, and who will supply it.

4.1.5 Adopt Author Conventions

Author conventions are those latitudes granted to the modeler (author) to assist in the development of the model, its review kits, and other presentations. Their purpose is specifically for the enhancement of the presentation of the material. They may be used anywhere to facilitate a better understanding and appreciation of any portion of the model. For example, a standard naming convention may be adopted for entity and attribute names.

Author conventions may take on various forms and appear in various places. But the most important aspect of all of this is what author conventions are not.

- Author conventions are not formal extensions of the technique
- Author conventions are not violations of the technique

Author conventions are developed to serve specific needs. Each convention must be documented as it is developed and included in the Phase Zero documentation that is distributed for review.

4.2 Phase One - Entity Definition

The objective of Phase One is to identify and define the entities that fall within the problem domain being modeled. The first step in this process is the identification of entities.

4.2.1 Identify Entities

An "entity" within the context of an IDEF1X Model represents a set of "things" which have data associated with them. Where, a "thing" may be an individual, a physical substance, an event, a state, a deed, an idea, a notion, a point, a place, etc. Members of the set represented by the entity have a common set of attributes or characteristics. For example, all members of the set of employees have an employee number, name, and other common attributes. An individual member of an entity set is referred to as an instance of the entity. For example, the employee named Jerry with employee number 789 is an instance of the entity EMPLOYEE. Entities are always named with by a singular, generic noun and must be an attribute (key) which will uniquely identify each of its instances.

Most of the entities can be directly or indirectly identified from the source material collected during Phase Zero. If the modeling effort is expanding or refining a previous data model, appropriate entities should be selected from the prior model. For entities not previously defined, the modeler must first identify within the list of source material names those things which represent potentially viable entities. One way this can be simplified is to identify the occurrences of all nouns in the list. For example, terms such as part, vehicle, machine, drawing, etc., would at this stage be considered potentially viable as entities. Another method is to identify those terms ending with the use of the word "code" or "number," for example, part number, purchase order number, routing number, etc. The phrase or word preceding the word "code" or "number" could also be considered at this stage as a potentially viable entity. For the remainder of the items on this list, the modeler must ask whether the word represents an object or thing about which information is known, or is information about an object or thing. Those items that fall into the category of being objects about which information is known may also be viable entities.

Entities result from a synthesis of basic entity instances, which become members of the entity. This means that

some number of entity instances, all of whose characteristics are the same type, are represented as an entity. An example of this concept is shown in Figure 4-2. Each instance of an entity is a member of the entity, each with the same kind of identifying information.

In order to help separate entities from non-entities, the modeler should ask the following questions about each candidate entity:

- Can it be described? (Does it have qualities?)

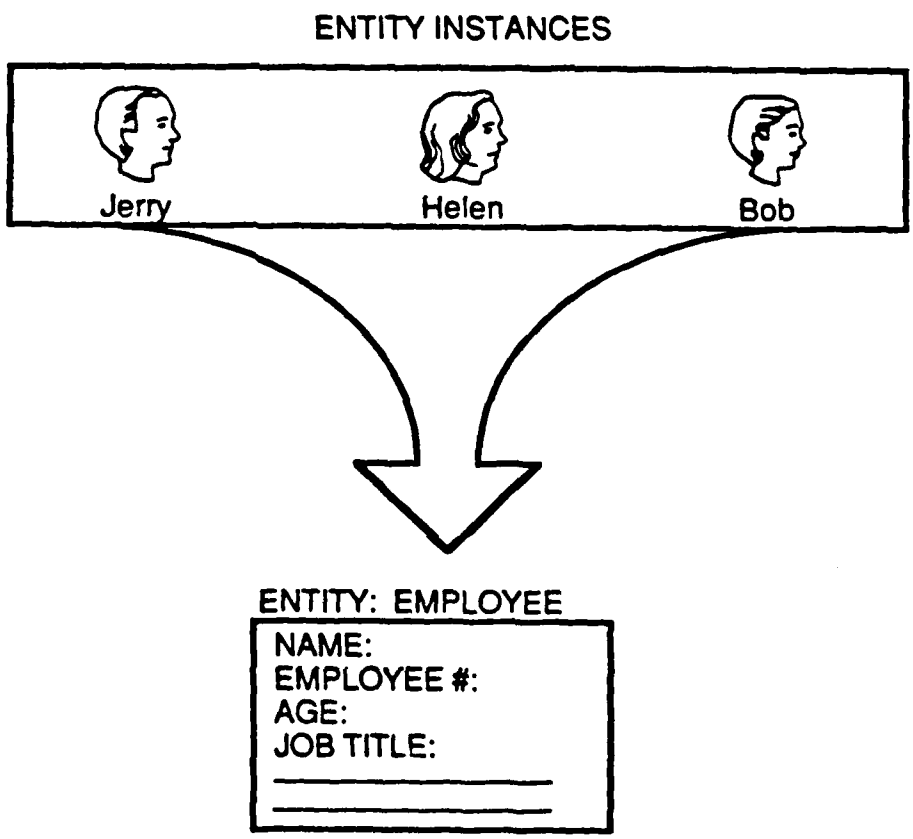


Figure 4-2. Synthesizing an Entity

- Are there several instances of these?
- Can one instance be separated/identified from another?
- Does it refer to or describe something? (A "yes" answer implies an attribute rather than an entity).

At the end of this analysis, the modeler has defined the initial entity pool. This pool contains all of the names of entities within the context of the model known at this point. As the modeler is building the entity pool, he assigns a discrete identification number to each entry and records a reference to its source. In this way, traceability of the information is maintained. The integrity of the pool remains intact, and the management of the pool is relatively easy. A sample of an entity pool is shown in Figure 4-3.

In all likelihood, not all names on the list will remain as entities by the end of Phase Four. In addition, a number of new entities will be added to this list and become a part of the information model as the modeling progresses and the understanding of the information improves.

Entity names discovered in phases further downstream must be added to the entity pool and assigned a unique identification number. One of the products of the Phase One effort is the entity pool. It must be up to date to remain viable.

4.2.2. Define Entities

The next product to emerge out of the Phase One efforts is the beginning of the entity glossary. During Phase One, the glossary is merely a collection of the entity definitions.

The components of an entity definition include:

1. ENTITY NAME

   Entity name is the unique name by which the entity will be recognized in the IDEF1X model. It should be descriptive in nature. Although abbreviations and acronyms are permitted, the entity name must be meaningful.

Number	Entity Name	Source Material Log Number
E-1	Backorder	2
E-2	Bill of Lading	2
E-3	Carrier	2
E-4	Clock Card	3
E-5	Commodity	2
E-6	Contractor	4
E-7	Delivery	2
E-8	Department	2
E-9	Deviation Waiver	6
E-10	Deviation Waiver Request	6
E-11	Division	4
E-12	Employee	10
E-13	Employee Assignment	10
E-14	Employee Skill	10
E-15	End Item Requirement	6
E-16	Group	6
E-17	Inspection Tag	12
E-18	Inventory Adjustment	6
E-19	Invoice	11
E-20	Issue From Stock	12
E-21	Job Card	12
E-22	Labor Report	12
E-23	Machine Queue	14
E-24	Master Schedule	14
E-25	Material	14
E-26	Material Availability	15
E-27	Material Handling Equipment	15
E-28	Material Inventory	15
E-29	Material Move Authorization	15
E-30	Material Requirement	15
E-31	Material Requisition	15
E-32	Material Requisition Item	15

Figure 4-3. Sample Entity Pool

2. ENTITY DEFINITION

This is a definition of the entity that is most commonly used in the enterprise. It is not intended to be a dictionary. Since the meaning of the information reflected in the model is specific to the viewpoint of the model and the context of the model defined in Phase Zero, it would be meaningless (if not totally confusing) to include definitions outside of the Phase Zero scope. However, there may be slight connotative differences in the way that the entity is defined, primarily based on contextual usage. Whenever these occur, or whenever there are alternate definitions (which are not necessarily the most common from the viewpoint of the model), these should also be recorded. It is up to the reviewers to identify what definition should be associated with the term used to identify the entity. The Phase One definition process is the mechanism used to force the evolvement of a commonly accepted definition.

3. ENTITY SYNONYMS

This is a list of other names by which the entity might be known. The only rule pertaining to this is that the definition associated with the entity name must apply exactly and precisely to each of the synonyms in the synonym list.

Entity definitions are most easily organized and completed by first going after the ones that require the least amount of research. Thus, the volume of glossary pages will surge in the shortest period of time. Then the modeler can conduct the research required to fully define the rest of the names in the pool. Good management of the time and effort required to gather and define the information will ensure that modeling continues at a reasonable pace.

4.3 Phase Two - Relationship Definition

The objective of Phase Two is to identify and define the basic relationships between entities. At this stage of modeling, some relationships may be non-specific and will require additional refinement in subsequent phases. The primary outputs from Phase Two are:

- Relationship matrix

- Relationship definitions
- Entity-level diagrams

4.3.1 Identify Related Entities

A "relationship" can be defined as simply an association or connection between two entities. More precisely, this is called a "binary relationship". IDEF1X is restricted to binary relationships because they are easier to define and understand than "n-ary" relationships. They also have a straightforward graphical representation. The disadvantage is a certain awkwardness in representing n-ary relationships. But there is no loss of power since any n-ary relationships can be expressed using n binary relationships.

A relationship instance is the meaningful association or connection between two entity instances. For example, an instance of the entity OPERATOR, whose name is John Doe and operator number is 862, is assigned to an instance of the entity MACHINE, whose type is drill press and machine number is 12678. An IDEF1X relationship represents the set of the same type of relationship instances between two specific entities. However, the same two entities may have more than one type of relationship.

The objective of the IDEF1X model is not to depict all possible relationships but to define the interconnection between entities in terms of existnce dependency (parent-child) relationships. That is, an association between a parent entity type and a child entity type, in which each instance of the parent is associated with zero, one, or more instances of the child and each instance of the child is associated with exactly one instance of the parent. That is, the existence of the child entity is dependent upon the existence of the parent entity. For example, a BUYER issues zero, one or more PURCHASE-ORDERS, and a PURCHASE-ORDER is issued by one BUYER.

If the parent and child entity represent the same real-world object, then the parent entity is a generic entity and the child is a category entity. For each instance of the category entity, there is always one instance of the generic entity. For each instance of the generic entity, there may be zero or one instances of the category. For example, a SALARIED-EMPLOYEE is an EMPLOYEE. An EMPLOYEE may or may not be a SALARIED-EMPLOYEE. Several category entities may be asso-

ciated with a generic entity in a categorization but only one category must apply to a given instance of the generic entity. For example, a categorization relationship might be used to represent the fact that an EMPLOYEE may be either a SALARIED-EMPLOYEE or an HOURLY-EMPLOYEE, but not both.

In the initial development of the model, it may not be possible to represent all relationships as parent-child or categorization relationships. Therefore, in Phase Two non-specific relationship may be specified. Non-specific relationships take the general form of zero, one, or more to zero, one, or more (N:M). Neither entity is dependent upon the other for its existence.

The first step in Phase Two is to identify the relationships that are observed between members of the various entities. This task may require the development of a relationship matrix as shown in Figure 4-4. A relationship matrix is merely a two-dimensional array, having a horizontal and a vertical axis. One set of predetermined factors (in this case all the entities) is recorded along one of the axes, and second set of factors (in this case, also all the entities) is recorded along the other axis. An "X" placed in the inter-secting points where any of the two axes meet is used to indicate a possible relationship between the entities involved. At this point, the nature of the relationship is unimportant; the fact that a relationship may exist is sufficient.

The general tendency for new modelers is to over specify the relationships between entities. Remember, the goal is to ultimately define the model in terms of parent-child relationships. Avoid identifying indirect relationships. For example, if a DEPARTMENT is responsible for one or more PROJECTS and each PROJECT initiates one or more PROJECT-TASKS, then a relationship between DEPARTMENT and PROJECT TASK is not needed since all PROJECT-TASKS are related to a PROJECT and all PROJECTS are related to a DEPARTMENT.

More experienced modelers may prefer to sketch entity-level diagrams rather than actually construct the relationship matrix. However, it is important to define relationships as they are identified.

Entity-Relationship Matrix Example

	Buyer	Requester	Approver	Purchase Requisition	Purchase Req. Item
Buyer		X		X	
Requester	X			X	
Approver				X	
Purchase Requisition	X	X	X		X
Purchase Req. Item				X	

An Entity-Relationship Matrix only reflects that a relationship of some kind may exist.

Figure 4-4. Entity/Relationship Matrix

4.3.2 Define Relationships

The next step is to define the relationships which have been identified. These definitions include:

- Indication of dependencies
- Relationship name
- Narrative statements about the relationship

As a result of defining the relationships, some relationships may be dropped and new relationships added.

In order to establish dependency, the relationship between two entities must be examined in both directions. This is done by determining cardinality at each end of the relationship. To determine the cardinality, assume the existence of an instance of one of the entities. Then determine how many specific instances of the second entity could be related to the first. Repeat this analysis reversing the entities.

For example, consider the relationship between the entities CLASS and STUDENT. An individual STUDENT may be enrolled in zero, one, or more CLASSES. Analyzing from the other direction, an individual CLASS may have zero, one, or more STUDENTS. Therefore, a many to many relationship exists between CLASS and STUDENT with a cardinality of zero, one, or more at each end of the relationship. (Note: this relationship is non-specific since a cardinality of "exactly one" does not exist at either end of the relationship. The non-specific relationship must be resolved later in the modeling process.)

Take the relationship between the entities BUYER and PURCHASE ORDER as another example. An individual BUYER may issue zero, one, or many PURCHASE-ORDERS. An individual PURCHASE-ORDER is always issued by a single BUYER. Therefore, a one to many relationship exist between BUYER and PURCHASE-ORDER with a cardinality of one at the BUYER end of the relationship and a cardinaltiy of zero, one, or more at the PURCHASE-ORDER end of the relationship. (Note: this is a specific relationship since an "exactly one" cardinality exists at the BUYER end of the relationship, i.e. BUYER is a parent entity to PURCHASE-ORDER).

Once the relationship dependencies have been established, the modeler must then select a name and may develop a defini-

tion for the relationship. The relationship name is a short phrase, typically a verb with a conjunction to the second entity mentioned. This phrase reflects the meaning of the relationship represented. Frequently, the relationship name is simply a single verb; however, adverbs and prepositions also appear frequently in relationship names. Once a relationship name is selected, the modeler should be able to read the relationships and produce a meaningful sentence defining or describing the relationship between the two entities.

In the case of the specific relationship form, there is always a parent entity and a child entity; the relationship name is interpreted from the parent end first, then from the child to the parent. If a categorization relationship exists between the entities, this implies both entities refer to the same real-world object and the cardinality at the child end (or category entity) is always zero, or one. The relationship name may be omitted since the name "may be a" is implied. For example, EMPLOYEE may be a SALARIED-EMPLOYEE.

In the case of the nonspecific relationship form, there are two relationship names, one for each entity, separated by a "/" mark. In this case, the relationship names are interpreted from top to bottom or from left to right, depending on the relative positions of the entities on the diagram, and then in reverse.

Relationship names must carry meaning. There must be some substance in what they express. The full meaning, in fact, the modeler's rationale in selecting a specific relationship name, may be documented textually by a relationship definition. The relationship definition is a textual statement explaining the relationship meaning. The same rules of definition that apply to the entity definitions also apply to the relationship definition:

- They must be specific
- They must be concise
- They must be meaningful

For example, if a one to zero or one relationship was defined between two entities such as OPERATOR and WORKSTATION, the relationship name might read "is currently assigned to".

This relationship could be supported by the following definition:

> "Each operator may be assigned to some number of workstations during any shift, but this relationship reflects the one the operator is assigned to at the moment."

4.3.3 Construct Entity-Level Diagrams

As relationships are being defined, the modeler may begin to construct entity-level diagrams to graphically depict the relationships. An example of an entity-level diagram is shown in Figure 4-5. At this stage of modeling, all entities are shown as square boxes and non-specific relationships are permitted. The number and scope of entity-level diagrams mayvary depending on the size of model and the focus of individual reviewers. If feasible, a single diagram depicting all entities and their relationships is helpful for establishing context and ensuring consistency. If multiple diagrams are generated, the modeler must take care that the diagrams are consistent with one another as well as with the entity and relationship definitions. The combination of entity-level diagrams should depict all defined relationships.

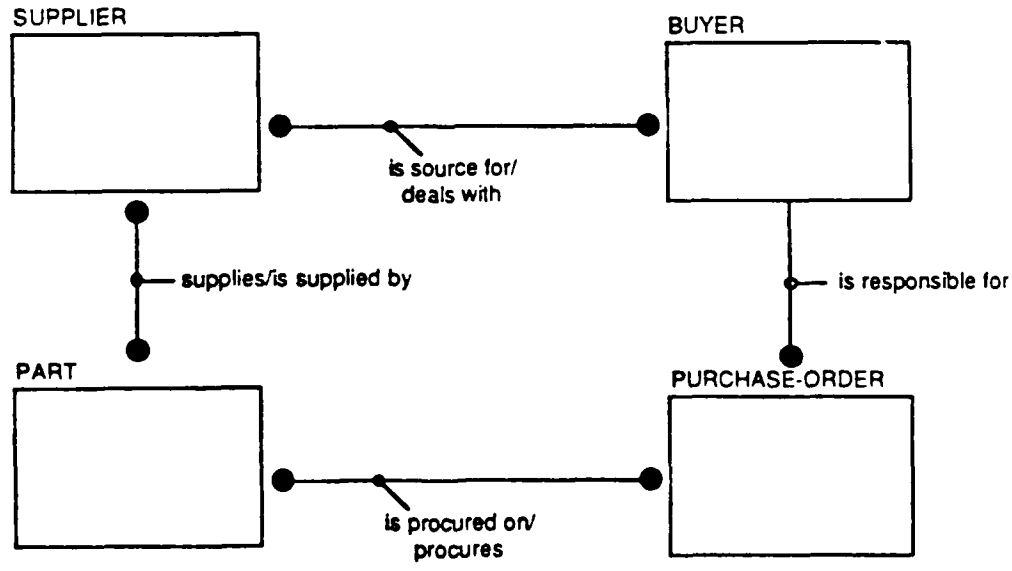


Figure 4-5. Entity-Level Diagram

A special case of the entity-level diagram focuses on a single entity and is referred to simply as an "Entity Diagram." An example is shown in Figure 4-6. The generation of an entity diagram for each and every entity is optional, but specific guidelines should be followed if they are used:

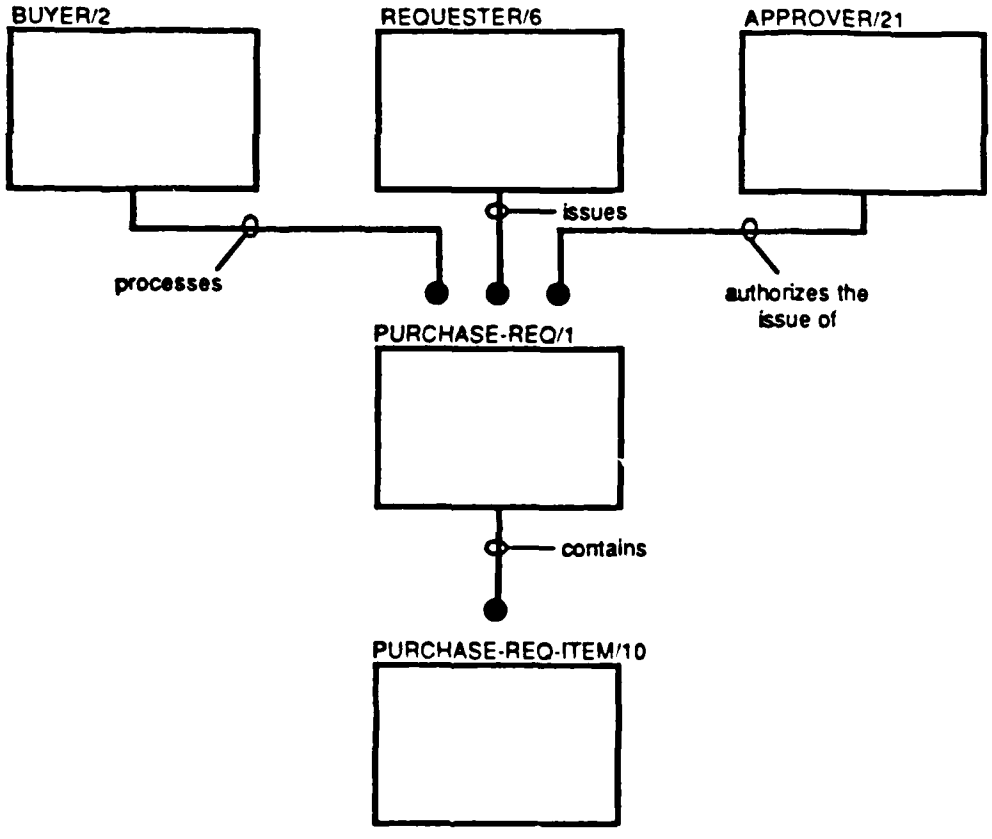


Figure 4-6. Phase Two (Entity-Level) Diagram Example

1. The subject entity will always appear in the approximate center of the page.

2. The parent or generic entities should be placed above the subject entity.

3. The child or category entities should be placed below the subject entity.

4. Nonspecific relationship forms are frequently shown to the sides of the subject entity box.

5. The relationship lines radiate from the subject entity box to the related entities. The only associations shown on the diagram are those between the subject entity and the related entities.

6. Every relationship line has a label; in the case of nonspecific relationship, the line has two labels, separated by a "/".

At this point, the information available for each entity includes the following:

1. The entity definition

2. The relationship names and optional definitions (for both parent and child relationships)

3. Depiction in one or more entity-level diagrams

The information about an entity can be expanded by the addition of reference diagrams, at the modeler's discretion. Reference diagrams (diagrams for exposition only, sometimes called FEOs) are an optional feature available to the modeler, to which individual modeler conventions may be applied. These diagrams are platforms for discussion between the modeler and the reviewers. They offer a unique capability to the modeler to document rationale, discuss problems, analyze alternatives, and look into any of the various aspects of model development. One example of a reference diagram is shown in Figure 4-7. This figure depicts the alternatives available in the selection of a relationship and is marked with the modeler's preference.

Another type of reference diagram, illustrated by Figure example, the modeler has identified the problem and its complexities for the reviewer's attention.

An "FEO" Used to Illustrate Alternatives

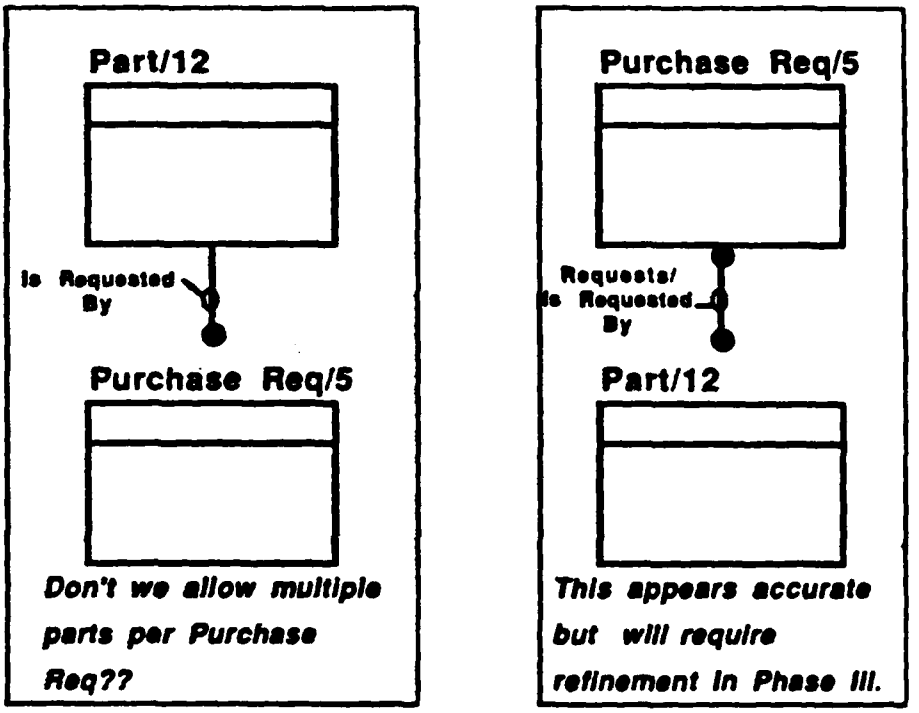


Figure 4-7. Reference Diagram (FEO)

By this stage, the modeler has compiled sufficient information to begin the formal validation through kits and walk-throughs. (See Sections 5.2 and 5.4)

4.4 Phase Three - Key Definitions

The objectives of Phase Three are to:

- Refine the non-specific relationships from Phase Two.
- Define key attributes for each entity.
- Migrate primary keys to establish foreign keys.
- Validate relationships and keys.

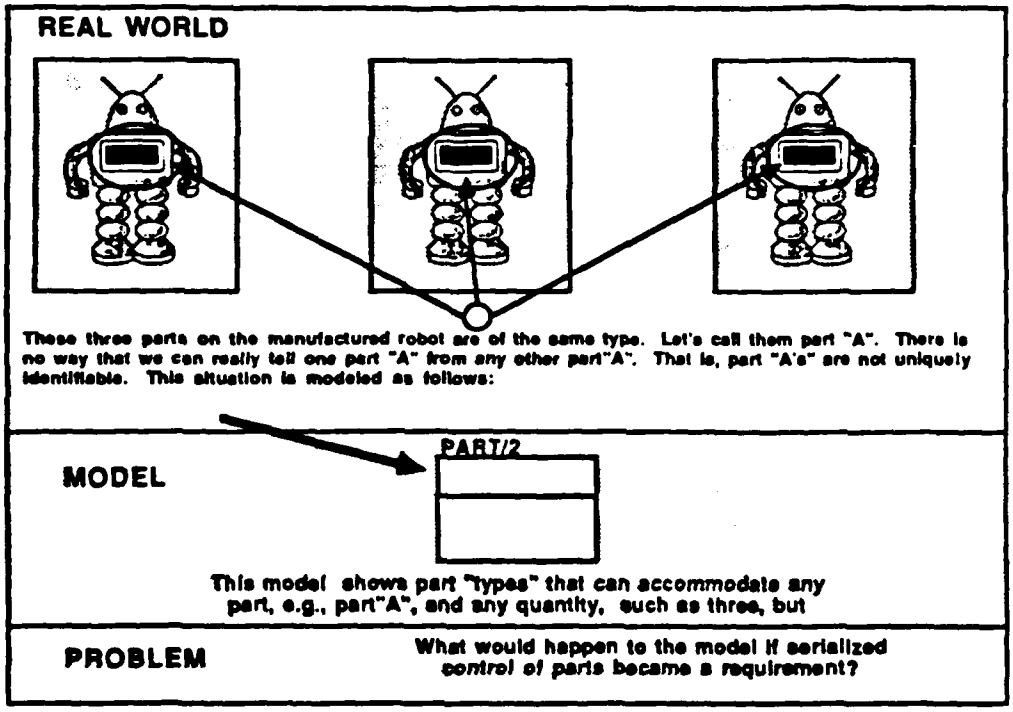


Figure 4-8. Example Reference Diagram

The results of Phase Three are depicted in one or more Phase Three (key-level) diagrams. In addition to adding key attribute definitions, Phase Three will expand and refine entity and relationship definitions.

4.4.1 Resolve Non-Specific Relationships

The first step in this phase is to ensure that all non-specific relationships observed in Phase Two have been refined. Phase Three requires that only a specific relationship form be used; either a specific connection (parent-child) relationship or categorization relationship. To meet this requirement, the modeler will employ the use of refinement alternatives. Refinement alternative diagrams are normally divided into two parts: the left part deals with the subject (the non-specific relationship to be refined), and the right part deals with the refinement alternative. An example of a refinement alternative dealing with a many-to-many resolution is exhibited in Figure 4-9.

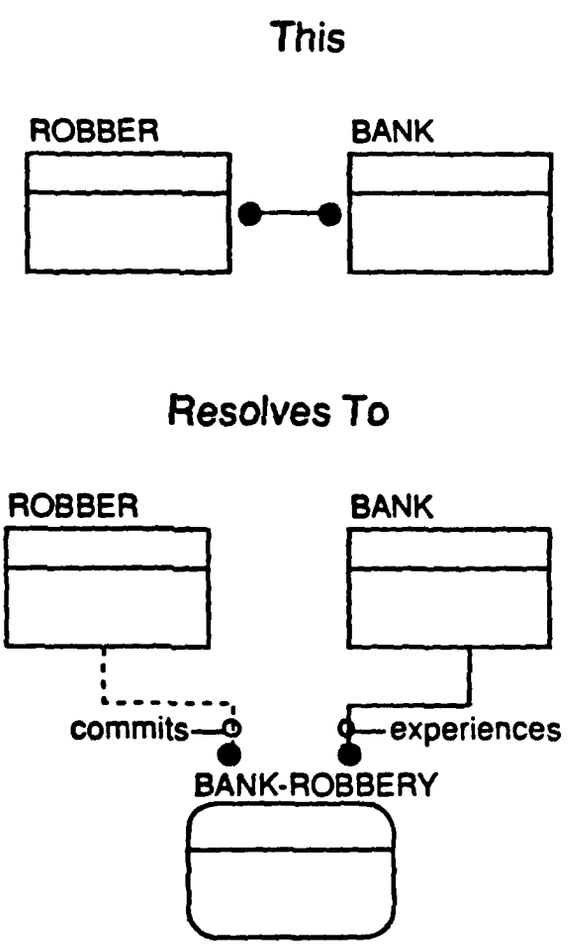


Figure 4-9. Non-Specific Relationship Refinement

The process of refining relationships translates or converts each non-specific relationship into two specific relationships. New entities evolve out of this process. The non-specific relationship shown in Figure 4-9 indicates that a ROBBER may rob many BANKS and a BANK may be robbed by many ROBBERs. However, we cannot identify which ROBBER robbed which BANK until we introduce a third entity, BANK-ROBBERY, to resolve the non-specific relationship. Each instance of the entity BANK-ROBBERY relates to exactly one BANK and one ROBBER.

In earlier phases, we have been working with what we might informally call the "natural entities." A natural entity is one that we will probably see evidenced in the source data list or in the source material log. A natural entity would include such names as the following:

1. Purchase Order

2. Employee

3. Buyer

It is during Phase Three that we begin to see the appearance of "associative entities" or what may informally be called "intersection entities." Intersection entities are used to resolve non-specific relationship and generally represent order pairs of things which have the same basic characteristics (unique identifier, attributes, etc.) as natural entities. Although the entity BANK-ROBBERY in the previous example might be considered a natural entity, it really represents the pairing of ROBBERS with BANKS. One of the subtle differences between the natural and intersection entities is in the entity names. Typically, the entity name for natural entities is a singular common noun. On the other hand, the entity name of the intersection entities may be a compound noun.

The intersection entity is more abstract in nature, and normally results from the application of rules governing the validity of entities that are first applied in Phase Three. The first of these rules is the rule requiring refinement of all non-specific relationships. This process of refinement is the first major step in stabilizing the integrated data structure.

This process of refinement involves a number of basic steps:

1. The development of one or more refinement alternatives for each non-specific relationship.

2. The selection by the modeler of a preferred alternative, which will be reflected in the Phase Three model.

3. The updating of Phase One information to include new entities resulting from the refinement.

4. The updating of Phase Two information to define relationships associated with the new entities.

4.4.2 Depict Function Views

The volume and complexity level of the data model at this point maybe appreciable. It was quite natural during Phase One to evaluate each entity independently of the other entities. At that juncture the entities were simply definitions of words. In Phase Two, it may have been practical to depict all relationships in a single diagram because the total volume of entities and relationships is typically not too large. In Phase Three, however, the volume of entities and the complexity of relationships being reflected in the model are normally such that an individual can no longer construct a total mental image of the meaning of the model. For this reason, the model may be reviewed and validated from multiple perspectives. These perspectives enable the evaluation of the model in a fashion more directly related to the functional aspects of the enterprise being modeled. These perspectives are represented by a "function view". Each function view is depicted in a single diagram. Its purpose is to establish limited context within which portions of the model can be evaluated at one sitting.

Function views can be instrumental in the evaluation and validation of the data model. The modeler must exercise some care in the determination or selection of topics illustrated in a function view. Two methods that have been used are the following:

1. Select sample source material as the topic of a function view, e.g., purchase order.

2. Relate the function view to job categories or specific processes, represented by the organizational departments or functional areas identified as sources in Phase Zero.

For example, in Figure 4-10 the data within the sample function view can be used to reconstruct a purchase order or to reconstruct a report about some number of purchase orders. When constructing a function view, the author must have the topic in mind so that it can be precisely expressed.

4.4.3 Identify Key Attributes

Phase Three of the IDEF1X methodology deals with the identification and definition of elements of data about entity instances referred to as candidate keys, primary keys, alternate keys, and foreign keys. The purpose of this step is to identify attribute values that uniquely identify each instance of an entity.

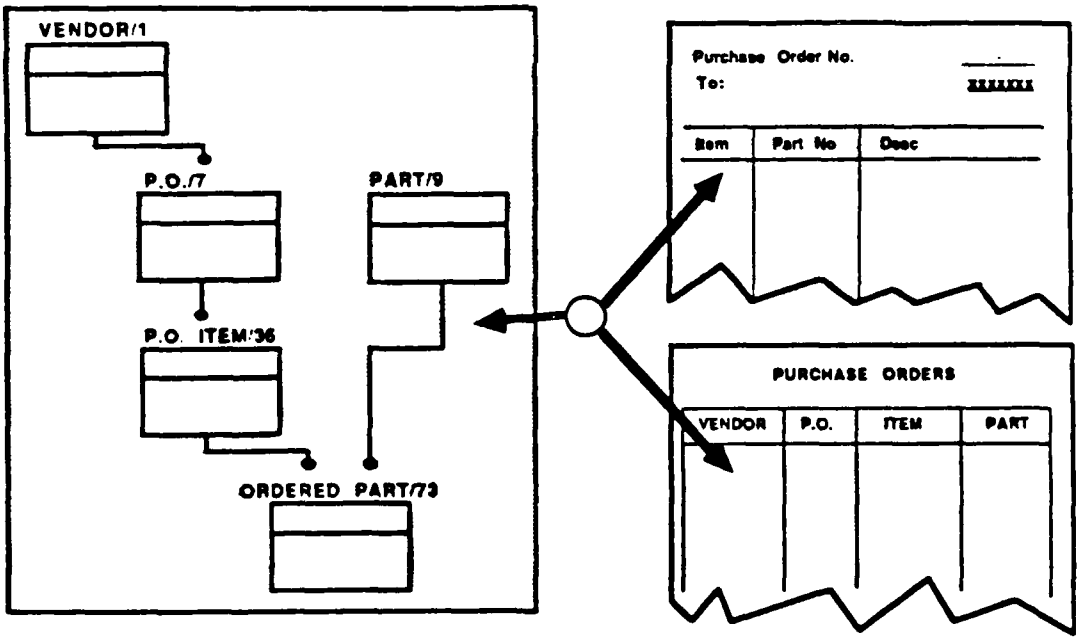


Figure 4-10. Scope of a Function View

It is important at this point that the definition and the meaning of the terms attribute instance and attribute be emphasized. An attribute instance is a property or characteristic of an entity instance. Attribute instances are composed of a name and a value. In other words, an attribute instance is one element of information that is known about a particular entity instance. Attribute instances are descriptors; that is, they tend to be adjective-like in nature.

An example of some attribute instances and their respective entity instances is shown in Figure 4-11. Note that the first entity instance, or individual, is identified with an employee number of "1," that the name associated with the entity instance is "Smith," and that the job of the entity instance is "operator." These attribute instances, taken all together, uniquely describe the entity instance and separate that entity instance from other similar entity instances. Every attribute instance has both a type and a value. The unique combination of attribute instances describes a specific entity instance. An attribute represents a collection of attribute instances of the same type that apply to all the entity instances of the same entity. Attribute names are typically singular descriptive nouns. In the example of the Employee entity, there are several attributes, including the following:

- Employee number
- Employee name
- Employee job/position

An example of how attribute instances are represented as attributes is also shown in Figure 4-11. The attribute instances belong to the entity instances. But the attributes themselves belong to the entity. Thus, an ownership association is established between an entity and some number of attributes.

An attribute has only one owner. An owner is the entity in which the attribute originates. In our example, the owner of the EMPLOYEE-NUMBER attribute would be the EMPLOYEE entity. Although attributes have only one owner, the owner can share the attribute with other entities. How this works will be discussed in detail in later segments.

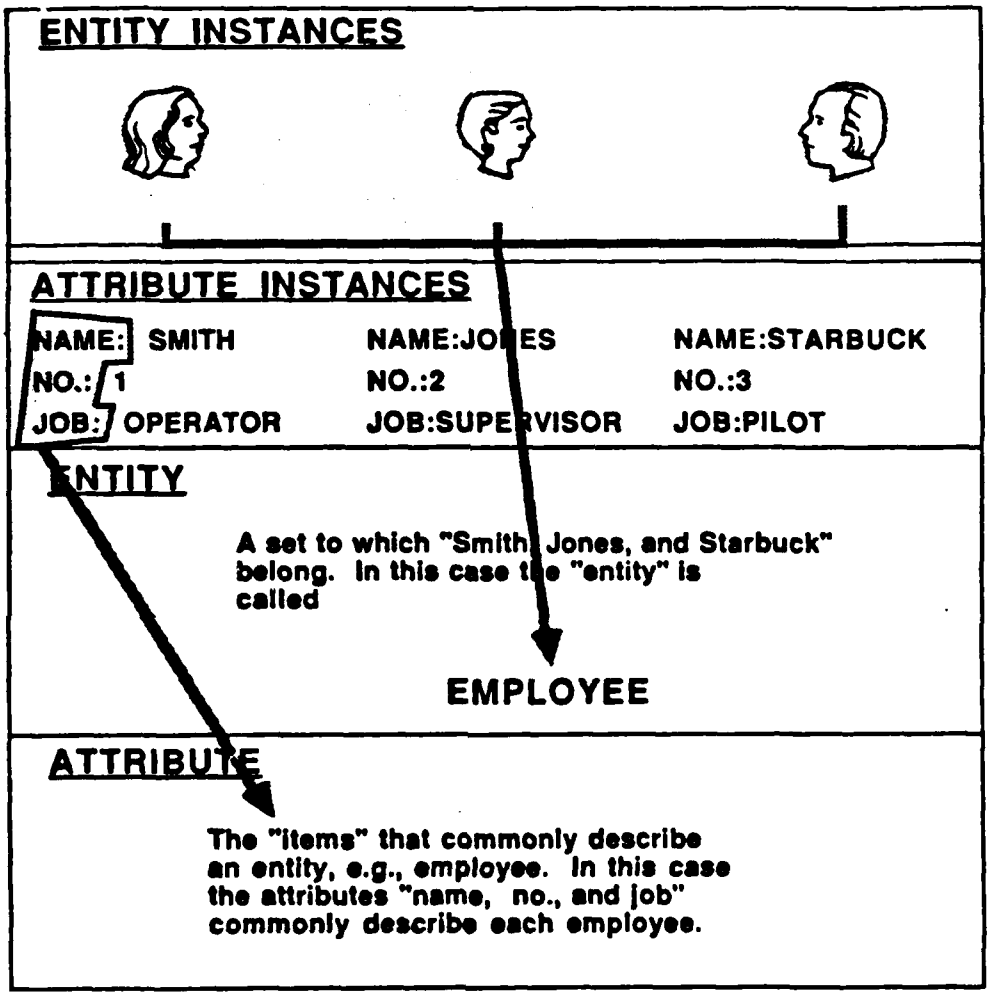


Figure 4-11. Attribute Examples

An attribute represents the use of an attribute instance to describe a specific property of a specific entity instance. Additionally, some attributes represent the use of an attribute instance to help uniquely identify a specific entity instance. These are informally referred to as key attributes.

Phase Three focuses on the identification of the key attributes within the context of our model. In Phase Four the nonkey attributes will be identified and defined.

One or more key attributes form a candidate key of an entity. A candidate key is defined as one or more key attributes used to uniquely identify each instance of an entity. An employee number is an example of one attribute being used as a candidate key of an entity. Each employee is identified from all the other employees by an employee number. Therefore, the EMPLOYEE-NUMBER attribute is a candidate key, which we can say uniquely identifies each member of the EMPLOYEE entity.

Some entities have more than one group of attributes that can be used to distinguish one entity instance from another. For example, consider the EMPLOYEE entity with the EMPLOYEE-NUMBER and SOCIAL-SECURITY-NUMBER attributes, either of which by itself is a candidate key. For such an entity one candidate key is selected for use in key migration and is designated as the primary key. The others are called alternate keys. If an entity has only one candidate key, it is automatically the primary key. So, every entity has a primary key, and some also have alternate keys. Either type can be used to uniquely identify entity instances, but only the primary key is used in key migration.

In the model diagram, a horizontal line is drawn through the subject entity box and the primary key is shown within the box, above that line. If there is more than one attribute in a primary key (e.g., project number and task number are both needed to identify project tasks), they all appear above the line. If an entity has an alternate key, it is assigned a unique alternate key number. In the diagram this number appears in parentheses following each attribute that is part of the alternate key. If an attribute belongs to more than one alternate key, each of the numbers appears in the parentheses. If an attribute belongs to both an alternate key and the primary key, it appears above the horizontal line followed by its alternate key number. If it does not belong to the primary key, it appears below the line. Examples of the various key forms are shown in Figure 4-12.

The process of identifying keys consists of:

1. Identifying the candidate key(s) for an entity.

2. Selecting one as the primary key for the entity.

Since some candidate keys may be the result of migration, key identification is an iterative process. Start with all the en-

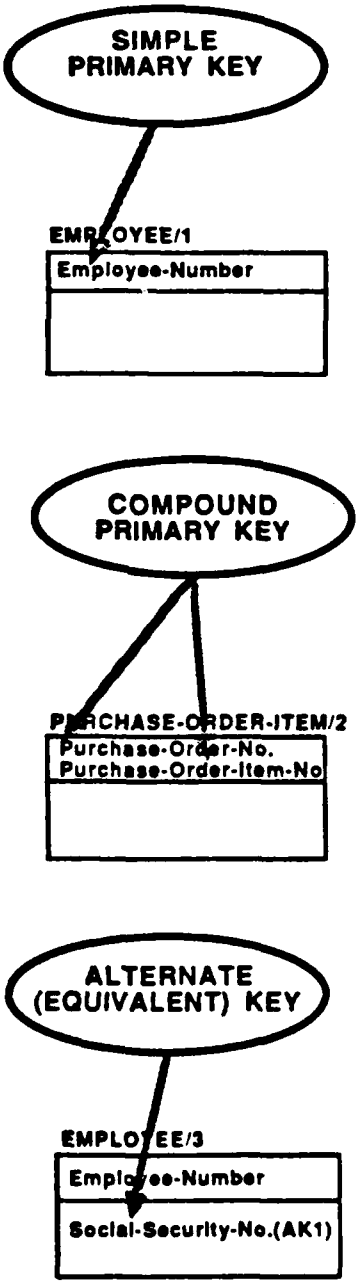


Figure 4-12. Key Forms

tities that are not a child or category in any relationship. These are usually the ones whose candidate keys are most obvious. These are also the starting points for key migration because they do not contain any foreign keys.

4.4.4 Migrate Keys

Key migration is the process of replicating one entity's primary key in another related entity. The replica is called a foreign key. The foreign key value in each instance of the second entity is identical to the primary key value in the related instance of the first entity. This is how an attribute that is owned by one entity comes to be shared by another. Three rules govern key migration:

1. Migration always occurs from the parent or generic entity to the child or category entity in a relationship.

2. The entire primary key (that is, all attributes that are members of the primary key) must migrate once for each relationship shared by the entity pair.

3. Alternate key and nonkey attributes never migrate.

Each attribute in a foreign key matches an attribute in the primary key of the parent or generic entity. In a category relationship the primary key of the category entity must be identical to that of the generic entity. In other relationships the foreign key attribute may be part of the primary key of the child entity, but it does not have to be. Foreign key attributes are not considered to be owned by the entities in which they appear, because they are reflections of attributes in the parent entities. Thus, each attribute in an entity is either owned by that entity or belongs to a foreign key in that entity.

In the model diagrams, foreign keys are noted much the same as alternate keys, i.e., "(FK)" appears behind each attribute that belongs to the foreign key. If the attribute also belongs to the primary key, it is above the horizontal line; if not, it is below.

If the primary key of a child entity contains all the attributes in a foreign key, the child entity is said to be "identifier dependent" on the parent entity, and the relationship is called an "identifying relationship". If any at-

tributes in a foreign key do not belong to the child's primary key, the child is not identifier dependent on the parent, and the relationship is called "nonidentifying". In Phase Three and Four diagrams, only identifying relationships are shown as solid lines; non-identifying relationships are shown as dashed lines.

An entity that is the child in one or more identifying relationships is called an "identifier-dependent entity". One that is the child in only non-identifying relationships (or is not the child in any relationships) is called an "identifier-independent entity". In Phase Three and Four diagrams, only identifier-independent entities are shown as boxes with square corners; dependent entities are shown as boxes with rounded corners.

An example of key migration of an attribute from a parent entity to a child entity is shown in Figure 4-13.

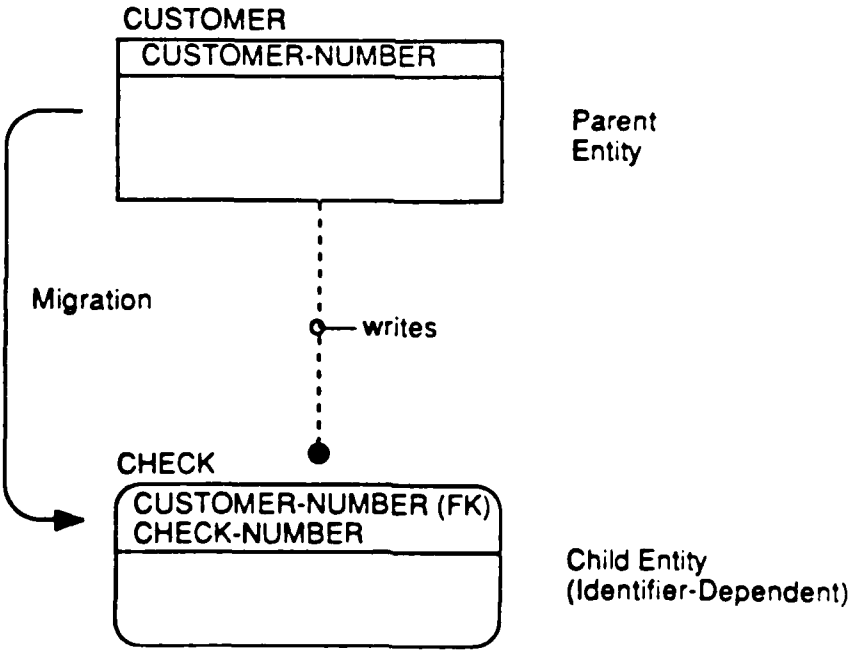


Figure 4-13. Key Migration to an Indentifier-Dependent Entity

In this example the CUSTOMER-NUMBER attribute (the primary key of the CUSTOMER entity) migrates to (is a foreign key in) the CHECK entity. It is then used in the CHECK entity as a member of its primary key in conjunction with another attribute called CHECK-NUMBER, which is owned by CHECK. The two attributes (CUSTOMER-NUMBER and CHECK-NUMBER) together form the primary key for the CHECK.

An example of key migration of an attribute from an identifier-independent entity to another identifier-independent entity is shown in Figure 4-14. In this example, the DEPARTMENT-NO attributes migrates to EMPLOYEE. However, the primary key of EMPLOYEE is EMP-ID. Therefore, DEPT-NO appears as a foreign key below the key attribute line. The relationship line is dashed since it is a non-identifying relationship.

The same attribute can generate more than one foreign key in the same child entity. This occurs when the attribute migrates through two or more relationships into the child entity. In some cases, each child instance must have the same value for that attribute in both foreign keys. When this is so, the attribute appears only once in the entity and is identified as a foreign key. In other cases, a child instance may (or must) have different values in each foreign key. In these cases, the

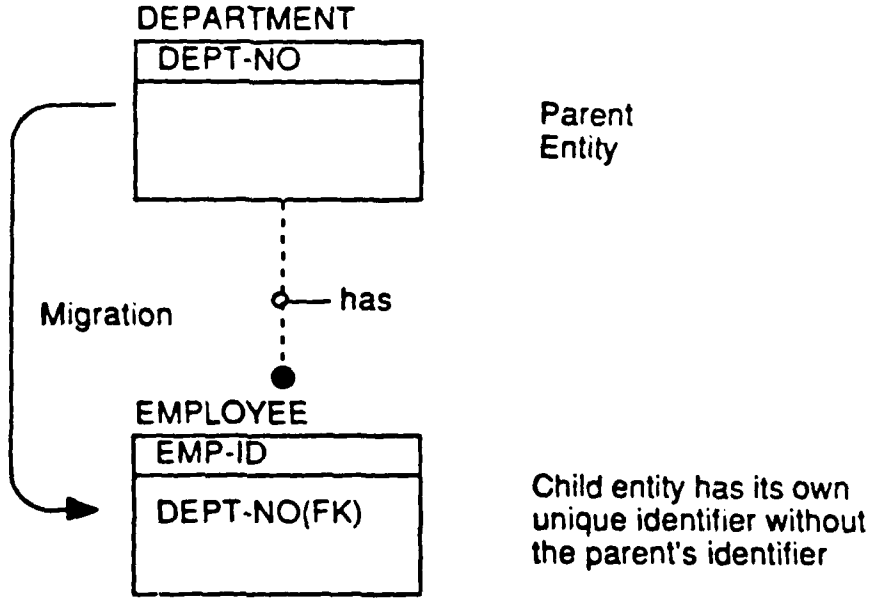


Figure 4-14. Migration to an Identifier-Independent Entity

attribute appears more than once in the entity and it becomes necessary to distinguish one occurrence from another. To do so, each is given a role name that suggests how it differs from the others. Figure 4-15 shows an example of this.

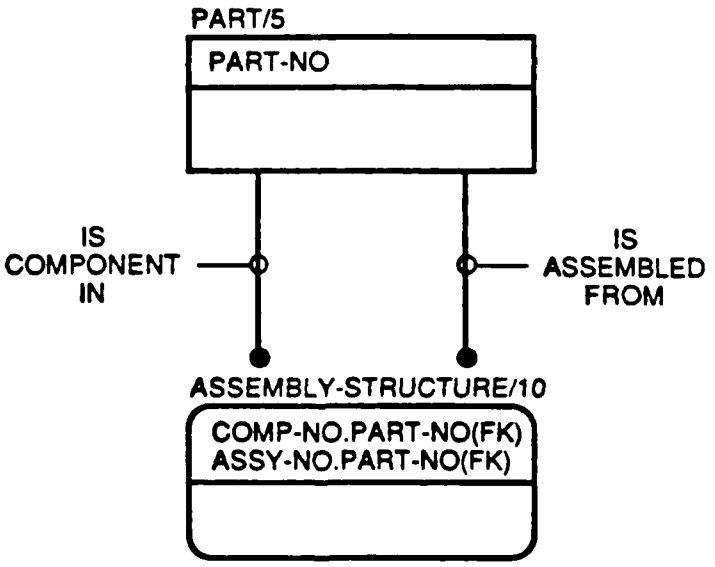
4.4.5 Validate Key and Relationships

Basic rules governing the identification and migration of keys are:

1. The use of non-specific relationship syntax is prohibited.

2. Key migration from parent (or generic) entities to child (or category) entities is mandatory.

3. The use of an attribute that might have more than one value at a time for a given entity instance is prohibited. (No-Repeat Rule)



Each of the migrated "PART-NO" keys is given an additional "ROLE" name identifying its function in the child. The role name is separated from the foreign key name by a period.

Figure 4-15. Attribute Role Names

4. The use of an attribute that could be null (i.e., have no value) in an entity instance is prohibited. (No-Null Rule)

5. Entities with compound keys cannot be split into multiple entities with simpler keys (Smallest - Key Rule).

6. Assertions are required for dual relationship paths between two entities.

We have already discussed the first two rules in previous sections, so we will turn our attention to the last group of rules at this point.

Figure 4-16 shows a diagram dealing with the application of the "No-Repeat Rule". Notice that the subject of the diagram shows both the PURCHASE-ORDER-NUMBER and PURCHASE-ORDER-ITEM-NUMBER as members of the primary key of PURCHASE-ORDER.

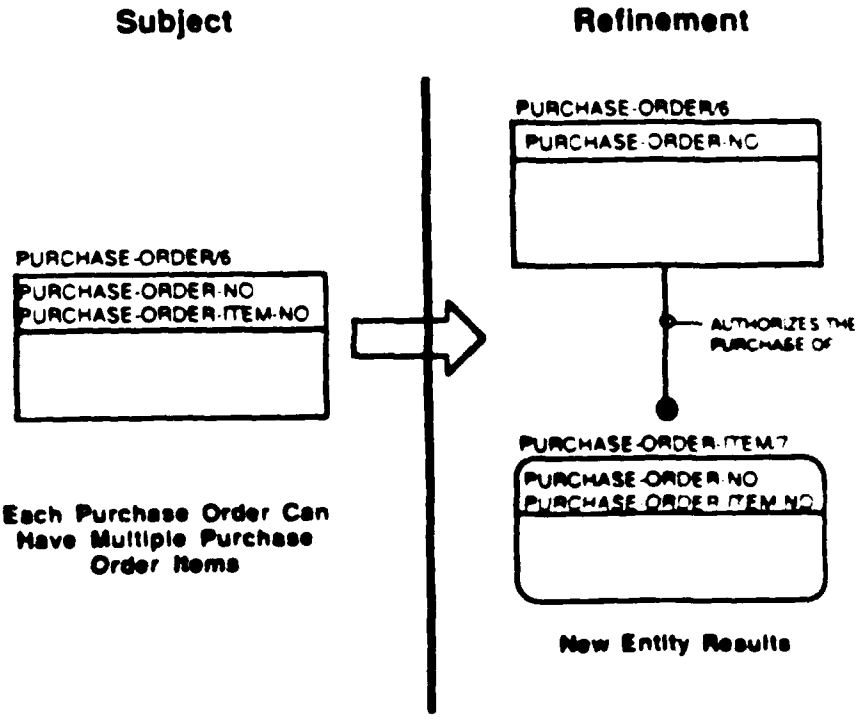


Figure 4-16 No-Repeat Rule Refinement

However, evaluation of the way PURCHASE-ORDER-ITEM-NUMBER is used will show that a single PURCHASE-ORDER (entity instance) can be many PURCHASE-ORDER-ITEM-NUMBER, one for each item being ordered. To properly depict this in the data model, a new entity called PURCHASE-ORDER-ITEM would have to be created, and the relationship label, syntax, and definition added. Then, the true characteristics of the association between purchase orders and purchase order items begin to emerge.

Figure 4-17 shows a refinement alternative diagram dealing with the application of the "No-Null Rule". Note that PART-NUMBER has migrated to PURCHASE-ORDER-ITEM. This association was established because purchase order items are linked in some way with the parts. However, the diagram as shown asserts that every purchase order item is associated with exactly one part number. Investigation (or perhaps reviewer comment) reveals that not all purchase order items are associated with parts. Some may be associated with services or other commodities that have no part numbers. This prohibits the migration of PART-NUMBER directly to the PURCHASE-ORDER-ITEM entity and requires the establishment of a new entity called ORDERED-PART in our example.

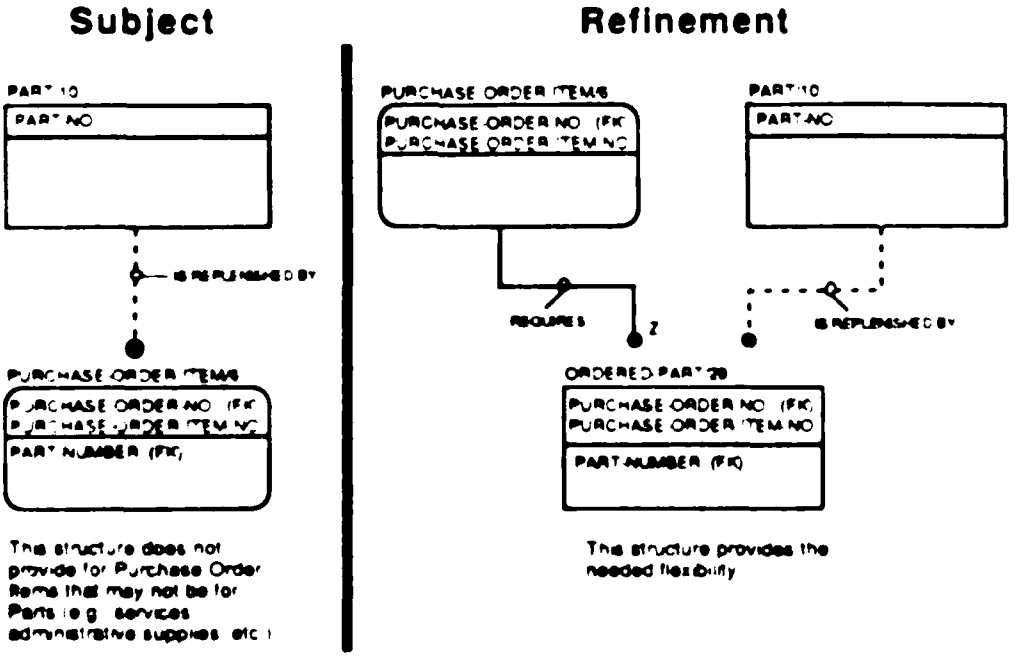


Figure 4-17 "No-Null" Rule Refinement

Once a new entity is established, key migration must occur as mandated by the migration rule, and the modeler will once again validate the entity-relationship structure with the application of the No-Null and No-Repeat Rules.

Each compound key should be examined to make sure it complies with the Smallest-Key Rule. This rule requires that no entity with a compound key can be split into two or more entities, with simpler keys (fewer components), without losing some information. This rule is a combination and extension of the fourth and fifth normal forms in relational theory. Other rules of normalization, such as Full-Functional-Dependency and No-Transitive-Dependency cannot be applied until non-key attributes are applied to the model in Phase Four.

In Phase Two, the tendency to specify redundant relationships was mentioned. However, the Phase Two analysis was primarily judgemental on the part of the modeler. With keys established, the modeler can now be more rigorous in the analysis. A dual path of relationships exists anytime there is a child entity with two relationships which ultimately lead back (through one or more relationships) to a common "root" parent entity. When dual paths exist, a "path assertion" is required to define whether the paths are equal, unequal, or indeterminate. The paths are equal if, for each instance of the child entity, both relationship paths always lead to the same root parent entity instance. The paths are unequal if, for each instance of the child entity, both relationships paths always lead to different instances of the root parent. The paths are indeterminate if they are equal for some child entity instances and unequal for others. If one of the paths consist of only a single relationship and the paths are equal, then the single relationship path is redundant and should be removed.

The simplest case of dual path relationship is one in which both paths consist of a single relationship. An example of this structure was shown in Figure 4-15. Since each instance of PART-USAGE may relate to two different instances of PART, no redundancy exists. The path assertion in this case would require the paths to be unequal, since a PART cannot be assembled into itself.

If one of the paths consists of multiple relationships and the other consists of a single relationship, the structure is referred to as a "triad". An example triad is shown in Figure 4-18. In this case, EMPLOYEE relates to DIVISIONs both directly and indirectly through DEPARTMENT. If the assertion is that

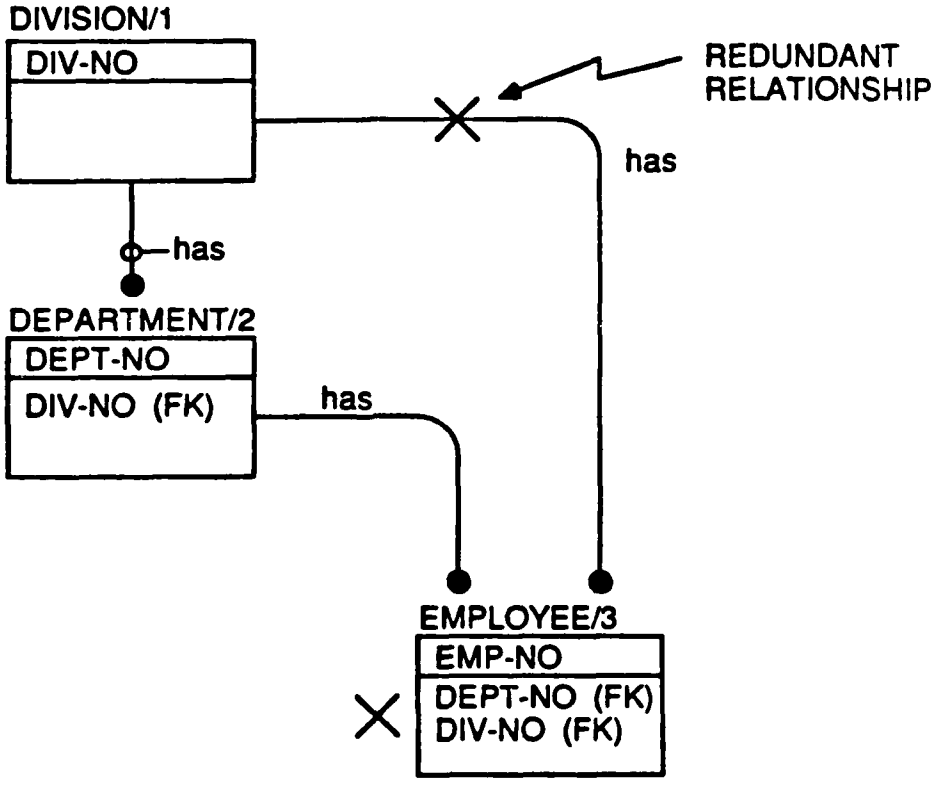


Figure 4-18. Example Triad

the DIVISION that an EMPLOYEE belongs to is the same DIVISION as his DEPARTMENT (i.e. equal parts) then the relationship between DIVISION and EMPLOYEE is redundant and should be removed. Note that, if we had asserted that some but not all EMPLOYEEs could, in fact, belong to two different DIVISIONS, another entity, such as LOANED-EMPLOYEE, would have to be added to satisfy application of the No-Null Rule to DIV-NO as a foreign key EMPLOYEE.

Assertions may also be applied to dual path relationships when both paths evolve more than one relationship. Figure 4-19 illustrates an example where two relationship paths exist between DEPARTMENT and TASK-ASSIGNMENT. If an EMPLOYEE can only be assigned to a PROJECT which is managed by his DEPARTMENT, then the paths are equal. If an EMPLOYEE can only be assigned to a PROJECT which is not managed by his DEPARTMENT, then the

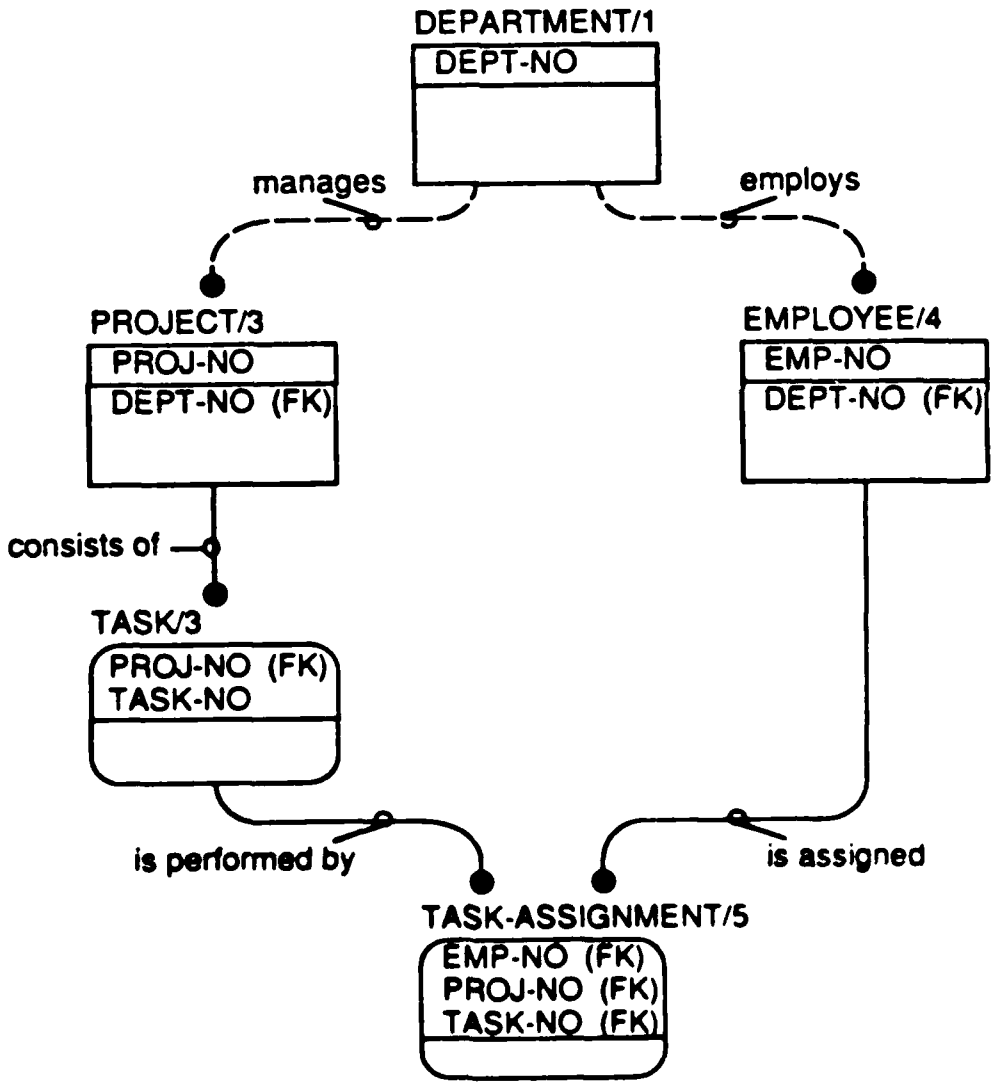


Figure 4-19. Path Assertions

paths are unequal. If an EMPLOYEE can be assigned to a PROJECT regardless of the managing DEPARTMENT, then the paths are indeterminate. Indeterminate paths are generally assumed unless an assertion is specified. Assertions should be attached as notes to the Phase Three diagrams and included in the child entity definition.

As primary key members are identified, entries are made into an attribute pool. An entity/attribute matrix may be used to identify the distribution and use of attributes throughout the model. The matrix has the following characteristics:

1. All entity names are depicted on the side.

2. All attribute names are depicted at the top.

3. The use of attributes by entities is depicted in the adjoining vectors, as appropriate, using codes such as the following:

   "O" = Owner
   "K" = Primary key
   "I" = Inherited

A sample of an entity/attribute matrix is shown in Figure 4-20. This matrix is a principal tool in maintaining model continuity.

4.4.6 Define Key Attributes

Once the keys have been identified for the model, it is time to define the attributes that have been used as keys. In Phase Three, definitions are developed for key attributes only. The same basic guidelines for these definitions apply: they must be precise, specific, complete, and universally understandable.

Attribute definitions are always associated with the entity that owns the attribute. That is, they are always members of the owner entity documentation set. Therefore, it is simply a matter of identifying those attributes owned by each entity, and used in that entity's primary key or alternate key. In the example shown in Figure 4-20 those attributes are coded "OK" on the entity/attribute matrix.

Entity	Attribute	Purchase Req. No.	Buyer Code	Vendor Code	Order Code	Change Number	Ship to Location	Vendor Name	Vendor Address	Confirmation Code	Confirmation Name	Extra Copy Code	Requester Name	Department Code	Ship Via	Buyer Name	Purchase Order No.	Purchase Req Issue Date	Q. C. Att. Code	Taxable Code	Resale Code	Pattern Number	Payment Terms
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Purchase Requistion	1	OK	I																				
Buyer	2		OK																				
Vendor	3			OK																			
Purchase Order	4		I	I													OK						
Requester	6												OK										
Part	9																						
Purchase Req. Item	10	IK																					
Purchase Req. Line	12	IK																					
Approver	21			IK																			
Part Source	22																						

Figure 4-20. Entity/Attribute Matrix

The attribute definition consists of:

- attribute name
- attribute definition
- attribute synonyms

4.4.7 Depict Phase Three Results

As a result of key identification and migration, the Function View diagrams may now be updated to reflect and refine relationships. The Phase Three Function View diagrams should also depict:

- Primary, alternate, and foreign key attributes
- Identifier-independent (square corner) and identifier-dependent (rounded corner) entities

- Identifying (solid line) and non-identifying (dashed-line) relationships.

An example of a Phase Three Function View is shown in Figure 4-21. Much of the information generated by Phase Three analysis may be reported by entity. Each entity documentation set consists of:

- A definition of the entity,
- A list of primary, alternate, and foreign key attributes,

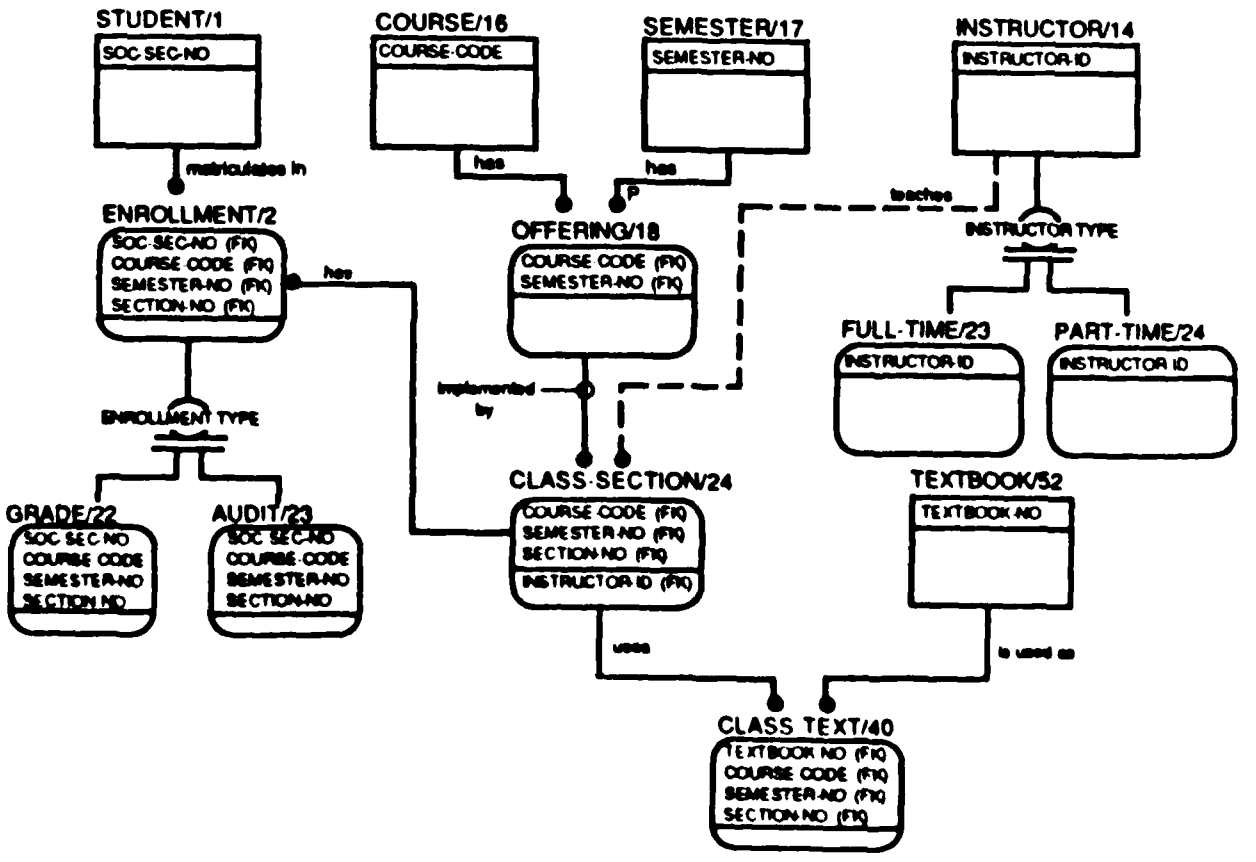


Figure 4-21. Example of Phase III Function View Diagram

- A definition for owned key attributes,
- A list of relationships in which the entity is a generic entity,
- A list of relationships in which the entity is a category entity,
- A list of identifying relationships in which the entity is a parent,
- A list of identifying relationships in which the entity is a child,
- A list of non-identifying relationships in which the entity is a parent, and
- A list of non-identifying relationships in which the entity is a child.
- A definition of dual path assertions (if appropriate)

Optionally, the modeler may also wish to construct an individual diagram for entity following the same approach as the optional Entity Diagram in Phase Two.

Along with a tabular listing of relationship definitions, a cross reference back to the associated entities is helpful. Owned and shared attributes should also be cross-referenced in the Phase Three reports.

4.5 Phase Four - Attribute Definition

Phase Four is the final stage of model developing. The objectives of this plan are to:

- Develop an attribute pool
- Establish attribute ownership
- Define non-key attributes
- Validate and refine the data structure

The results of Phase Four are depicted in one or more Phase Four (attribute-level) diagrams. At the end of Phase Four, the data model is fully refined (corresponding to fifth normal form in relational theory). The model is supported by a complete set of definitions and cross-references for all entities, attributes (key and non-key), and relationships.

4.5.1 Identify Nonkey Attributes

The construction of an attribute pool was begun in Phase Three with the identification of keys. The first step in Phase Four is to expand the attribute pool to include nonkey attributes. An attribute pool is a collection of potentially viable attribute names. Each name in the attribute pool occurs only once, and each is assigned a unique identifying number.

The process of constructing the attribute pool is similar in nature to construction of the entity pool. For the entity pool in Phase One, we extracted names that appeared to be object nouns from the Phase Zero source data list. Now we will return to the source data list and extract those names that appear to be descriptive nouns. Descriptive nouns (nouns that are used to describe objects) commonly represent attributes. Figure 4-22 shows an example attribute pool.

Many of the names on the source data list from Phase Zero were entered into the entity pool in Phase One as potential entities. Some of those names, however, may have been recognized by Phase Three as not qualifying as entities. In all probability, these are attributes. In addition, many of those names that were not selected from the list in the first place are probably attributes. The list, then, in conjunction with the knowledge gained during Phase One and Phase Two, is the basis for establishment of the attribute pool. The attribute pool is a list of potentially viable attributes observed within the context of the model. This list, in all likelihood, will be appreciably larger than the entity pool.

The attribute pool is the source of attribute names that are used in the model. In the event that attributes are discovered in later phases of the modeling effort, the attributes are added to the attribute pool and assigned a unique identifying number; they then progress to their intended use in the model.

4.5.2 Establish Attribute Ownership

The next step requires that each nonkey attribute be assigned to one owner entity. The owner entity for many of them will be obvious. For example, the modeler should be able to readily associate the VENDOR-NAME attribute with the VENDOR entity. However, some attributes may cause the modeler difficulty in locating their owner entities.

Number	Attribute Name	Source Data Number
1	Purchase Requisition Number	1
2	Buyer Code	2
3	Vendor Name	3
4	Order Code	4
5	Change Number	5
6	Ship to Location	6
7	Vendor Name	8
8	Vendor Address	8
9	Configuration Code	9
10	Configurer's Name	9
11	Extra Copy Code	10
12	Requester Name	11,42
13	Department Code	12
14	Ship Via	13
15	Buyer Name	14
16	Purchase Order Number	15
17	Purchase Requisition Issue Date	16
18	Quality Control Approval Code	17
19	Taxable Code	19
20	Resale Code	20
21	Pattern Number	21
22	Payment Terms	22
23	Freight on Board Delivery Location	18
24	Purchase Requisition Item Number	23
25	Quantity Ordered	24
26	Quantity Unit Measure	25
27	Part Number	26
28	Part Description	27
29	Unit Price	28
30	Price Unit of Measure	29
31	Purchase Requisition Line Code	31
32	Requested Delivery Date	32
33	Requested Delivery Quantity	33
34	Commodity Code	30

Figure 4-22. Sample Attribute Pool

If the modeler is not certain of the owner entity of an attribute, he may refer to the source material from which the attribute was extracted. This will aid in the determination of the owner. In Phase Zero, the source data list was established and became the foundation for the attribute pool. The source data list points the modeler to the locations where the attribute values represented are used in the original source material. By analyzing the usage of the attribute in the source material, the modeler will be able to more easily determine the owner entity in the data model. The modeler should keep in mind that the governing factor for determining ownership of the attributes is the occurrence attribute instances represented by the attribute values reflected in the source material. As each attribute is assigned to its owner entity, the assignment should be recorded.

4.5.3 *Define Attributes*

A definition must be developed for each of the attributes identified in Phase Four. The principles governing other definitions used in the data model, and particularly those in Phase Three, apply here as well. The definitions developed must be precise, specific, complete, and universally understandable. These attribute definitions are produced in the same format as the attribute definitions from Phase Three.

Attribute definition include:

- attribute name
- attribute definition
- attribute synonym(s)/aliases

Each attribute must be given a unique name since within an IDEF1X model the "same name - same meaning rule" applies to both entities and attributes. Therefore, the modeler may wish to adopt a standard approach for the attribute names. However, user recognizable/natural English names are encouraged for readability to support validation. Attribute names which must satisfy strict programming language rules, e.g. seven character FORTRAN variable names should always be identified as aliases if included at all.

Within the attribute definition, the modeler may wish to identify the attribute format, e.g. alpha-numeric code, text, money, date, etc. The domain of acceptable values may also be

specified in definition in terms of a list, e.g. Monday, Tuesday, Wednesday, Thursday, or Friday, or a range, e.g. greater than zero but less than 10. Assertions which involve multiple attributes may also be specified in definition. For example, the attribute EMPLOYEE-SALARY must be greater than $20,000 when EMPLOYEE-JOBCODE equals twenty.

4.5.4 Refine Model

The modeler is now ready to begin the Phase Four refinement of relationships. The same basic rules applied in Phase Three also apply to this refinement. The application of the No-Null and No-Repeat Rules introduced in Phase Three are now applied to both the key and nonkey attributes. As a result, the modeler can expect to find some new entities. As these entities are identified, the key migration rule must be applied, just as it was in Phase Three.

The only difference in applying the No-Null and No-Repeat Rules in Phase Four is that these rules are applied primarily to the nonkey attributes. Figure 4-23 illustrates the application of the No-Null Rule to a nonkey attribute. Figure

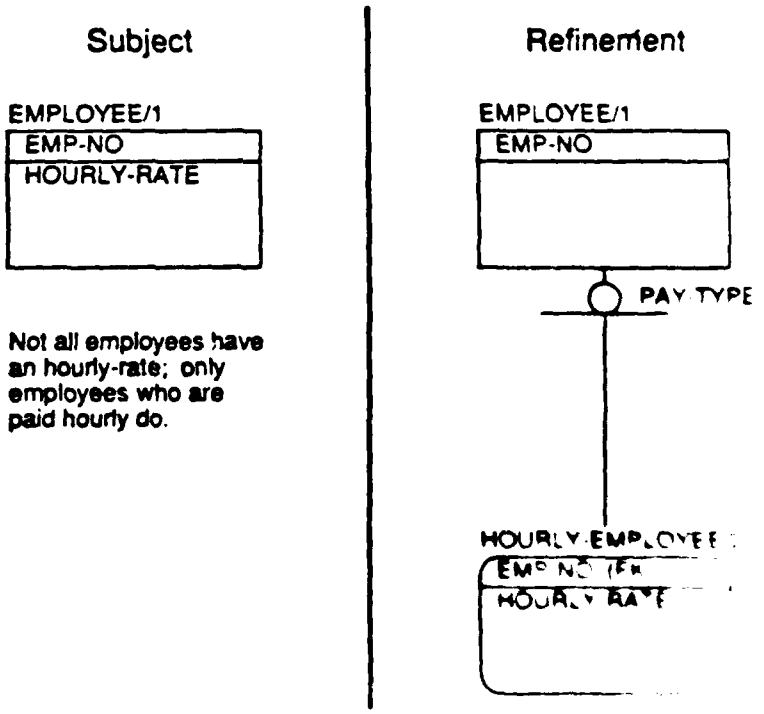
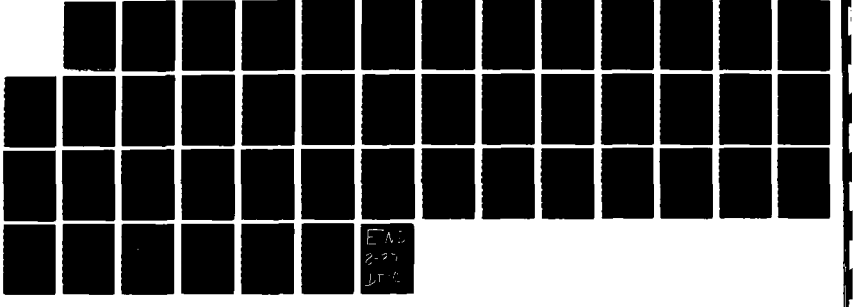
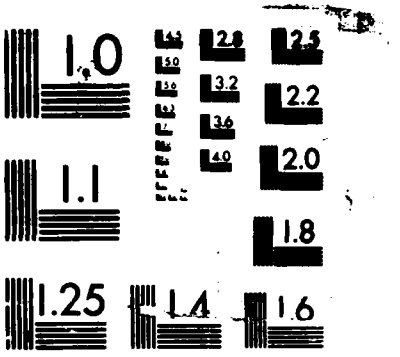


Figure 4-23. Phase IV - Applying the N

AD-A181 952 INTEGRATED INFORMATION SUPPORT SYSTEM (IISS) VOLUME 5 2/2
COMMON DATA MODEL S (U) GENERAL ELECTRIC CO
SCHENECTADY NY PRODUCTION RESOURCES CONUU
UNCLASSIFIED B N SNODGRASS ET AL 01 NOV 85 UM-620141002 F/G 12/5 NL





MICROCOPY RESOLUTION TEST CHART

4-24 illustrates the application of the No-Repeat Rule to a nonkey attribute.

An alternative to immediately creating new entities for attributes that violate the refinement rules is to mark the violators when they are found and create new entities later. Violators of the No-Null Rule can be marked by placing an "N" (for the No-Null Rule) or an "R" (for the No-Repeat Rule) in parentheses following their names in attribute diagrams.

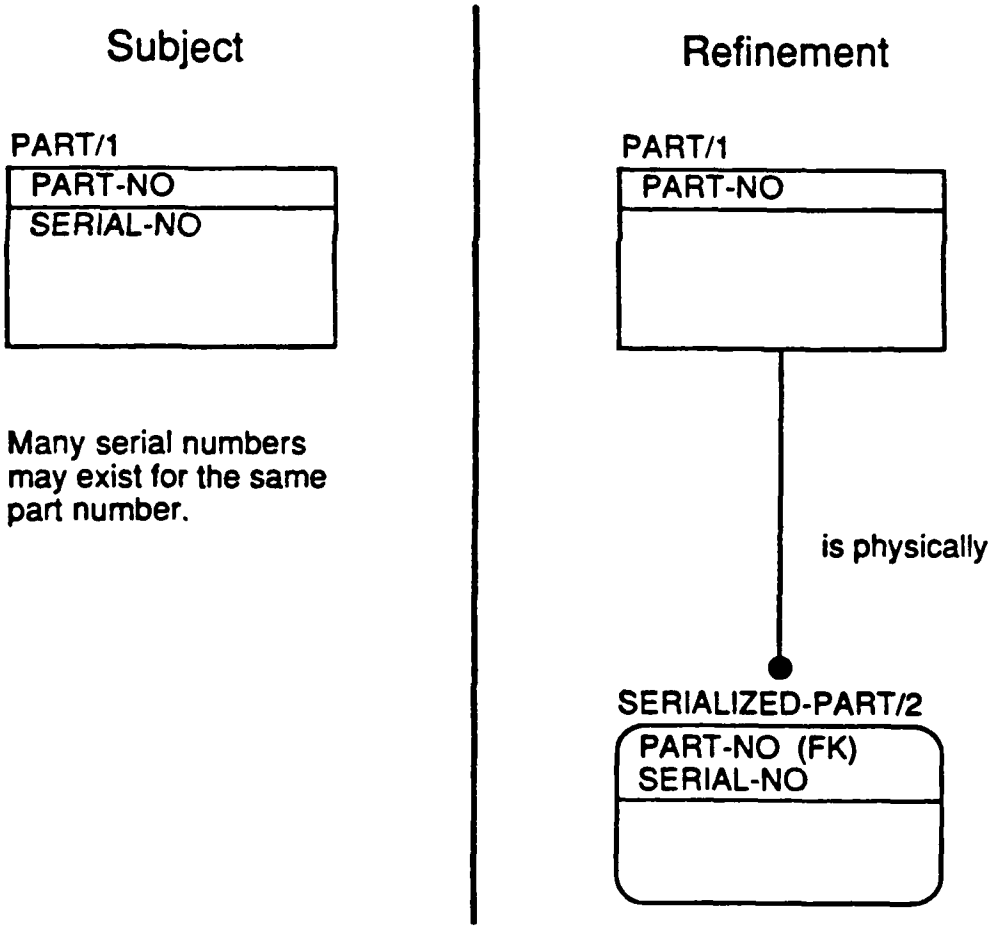


Figure 4-24. Phase IV - Applying the No-Repet Rule

As new entities emerge, they must be entered in the entity pool, defined, reflected in the relationship matrix, etc. In short, they must meet all of the documentation requirements of earlier phases in order to qualify for inclusion in Phase Four material.

The ownership of each attribute should also be evaluated for compliance with the Full-Functional-Dependency Rule. This rule states that no owned nonkey attribute value of an entity instance can be identified by less than the entire key value for the entity instance. This rule applies only to entities with compound keys and is equivalent to the second normal form in relational theory. For example, consider the diagram shown in Figure 4-19. If PROJECT-NAME was a nonkey attribute thought to be owned by the entity TASK, it would pass the no-null and no-repeat rules. However, since the PROJECT-NAME could be identified from only the PROJ-NO portion of the TASK key, it does not satisfy the Full-Functional-Dependency Rule. PROJECT-NAME would obviously be an attribute of the entity PROJECT.

All attributes in a Phase Four model must also satisfy the rule of No-Transitive-Dependency. This rule requires that no owned nonkey attribute value of an entity instance can be identified by the value of another owned or inherited, nonkey attribute of the entity instance. This rule is equivalent to the third normal form in the relational theory.

For example, consider the entity EMPLOYEE in Figure 4-19. If DEPT-NAME was to the entity EMPLOYEE as a nonkey attribute, it would satisfy the no-null and no-repeat rules. However, since DEPT-NAME could be determined from DEPT-NO which is an inherited nonkey attribute, it does not satisfy the No-Transitive-Dependency Rule and therefore, is not an owned attribute of EMPLOYEE. DEPT-NAME would obviously be a nonkey attribute of the entity DEPARTMENT.

A simple way to remember the rules of Full-Functional-Dependency and No-Transitive-Dependency is that "a nonkey attribute must be dependent upon the key, the whole key, and nothing but the key".

4.5.5 Depict Phase Four Results

As a result of attribute population, the Function View diagrams can now be updated to reflect a refinement of the model and expanded to show nonkey attributes. Nonkey attributes are listed below the line inside each entity box.

The size of the entity box may need to be expanded to provide room. An example of a Phase Four Function View is shown in Figure 4-25.

Supporting definitions and information for the model should be updated to reflect nonkey attribute definition and ownership assignment. This additional information may be reported by entity along the previously defined information. Each entity documentation set will now consist of:

- A definition of each entity
- A list of primary, alternate, and foreign key attributes
- A list of owned nonkey attributes
- A definition of each owned attribute (both key and nonkey)

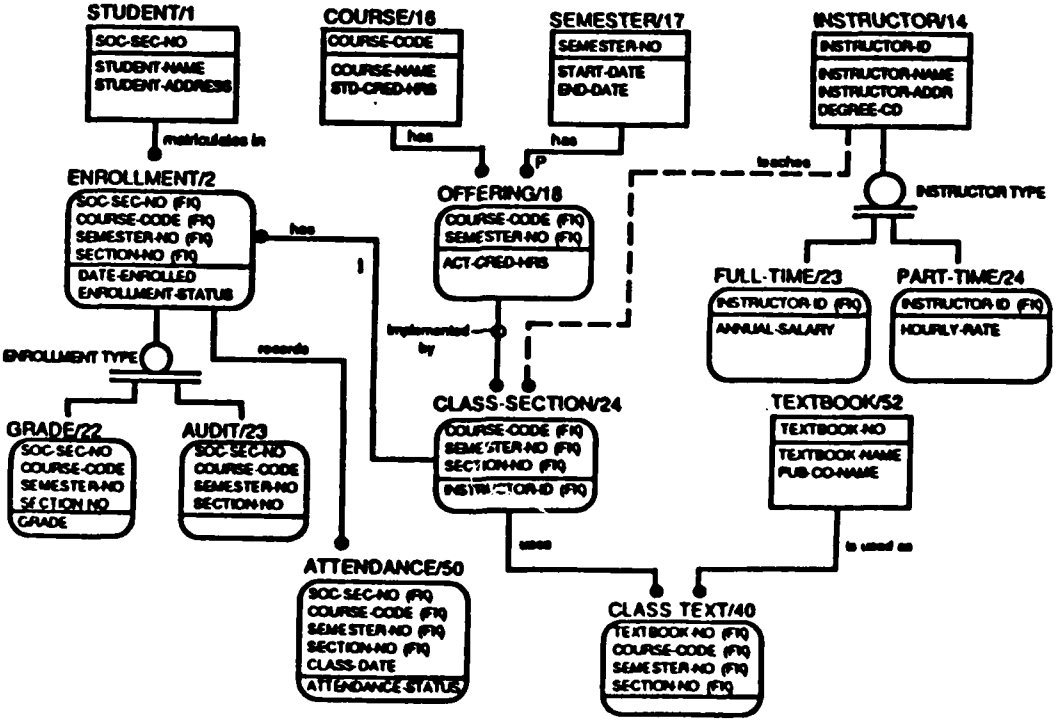


Figure 4-25. Example of Phase IV Function View Diagram

- A list of relationships in which the entity is the parent:

  - generic entity of a categorization
  - identifying parent relationships
  - non-identifying parent relationships

- A list of relationship(s) in which the entity is the child:

  - category entity of a categorization
  - identifying child relationships
  - non-identifying child relationships

- A definition of any dual path assertions

The optional individual entity diagrams may also be expanded to show nonkey attributes.

Relationship definitions may be repeated within the documentation set for each entity or listed separately with a cross-reference to the entity. Key and nonkey attributes should also be listed and cross-referenced to the entities.

SECTION 5

DOCUMENTATION AND VALIDATION

5.1 Introduction

The objective of IDEF1X is to provide a consistent integrated definition of the semantic characteristics of data which can be used to provide data administration and control for the design of shareable databases and integration of information systems. This means that the models must be well documented and thoroughly validated by both business professionals and systems professionals. Once an initial model has been built and validated, configuration management of data models may become an important consideration as new models are developed and integrated with existing models.

Much of the work of model documentation and configuration management can be eased through the use of software tools. At the simplest level of support a word processing system can be used to maintain the definition of entities, relationships and attributes. Standard interactive graphics packages may be used to create diagrams. These tools are limited in their benefit, however, because they do not take the model content into account. Most commercial data dictionary systems do no support the definition of semantic data models. However, some of the data dictionary systems have a user definable section which can be set up to store definitions and provide various reports. Another alternative is to construct a simple database to house the model description and to use the DBMS query facilities to generate various reports. The active three-schema dictionary of the U.S. Air Force Integrated Information Support System (IISS) itself is implemented with a relational database management system. Special modeling software has also recently become commercially available. Important features for a modeling software tool include:

- automated generation and layout of model diagrams,
- merging of data models,
- consistency-checking and automated refinement of models against the modeling rules,
- reporting capability, and

- configuration management support.

Although some level of automated support is highly desirable, it is not required for IDEF1X modeling. The following sections will discuss model documentation and validation issues assuming a minimum level of automated support.

5.2 IDEF1X Kits

A kit is a technical document which may contain diagrams, text, glossaries, decision summaries, background information, or anything packaged for review and comments. Each Phase of an IDEF1X modeling project requires the creation of one or more kits for review by subject matter experts and approvers of the model. Figure 5-1 summarizes the kit review cycle. If a kit is sent out for written comments, the author must always respond to the reviewer's comments. As an alternative to distributing kits for written comment, model walk-throughs may be used to gain reviewer concensus. Walk-throughs are discussed in Section 5.4.

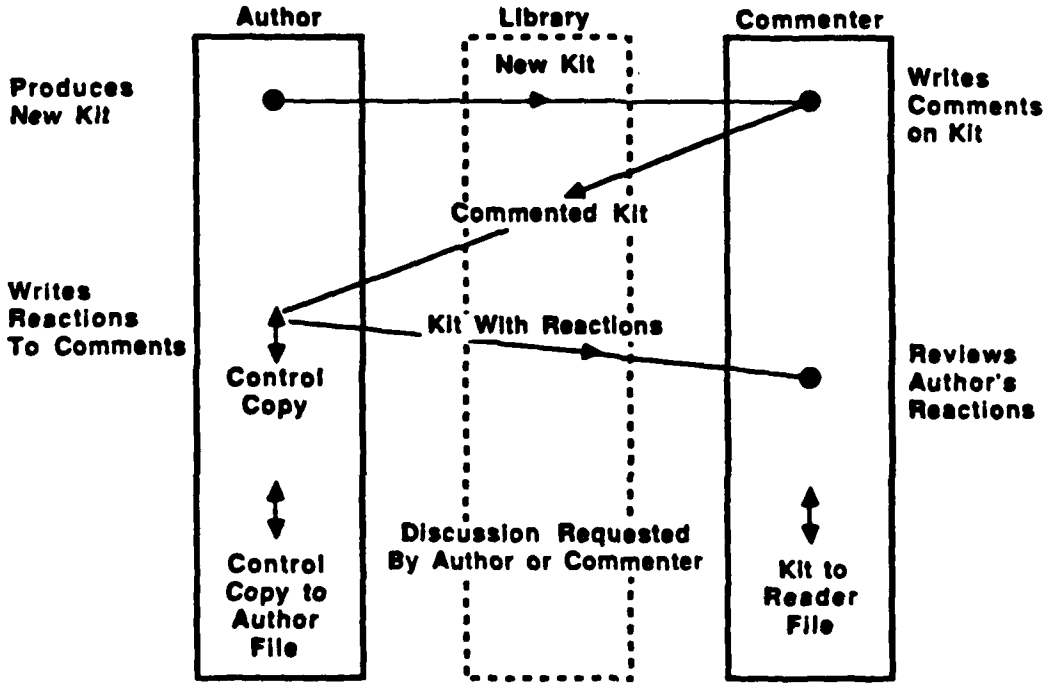


Figure 5-1. Kit Cycle

Each person participating in a project may wish to maintain a file of documentation received. A library function, however, should be established to maintain the master and reference files for each kit. The library function also serves as a distribution mechanism for kit review. A complete explanation of library files is given in the "ICAM Program Library Maintenance Procedures".

Although more than one kit may be used for each phase of modeling, the following is a summary of the overall kit contents which should be generated:

- Phase Zero Kit
  - Kit cover sheet
  - Statement of purpose and viewpoint
  - Model development and review schedule
  - Team membership and roles
  - Source materials (optional)
  - Author conventions (optional)

- Phase One Kit
  - Kit cover sheet
  - Entity pool
  - Entity definitions

- Phase Two Kit
  - Kit cover sheet
  - Relationship matrix (optional)
  - Phase Two (entity-level) diagrams
  - Entity reports (definition and relationships)
  - Relationship definitions
  - Relationship/entity cross-reference

- Phase Three Kit
  - Kit cover sheet
  - Phase Three (key-level) diagrams
  - Entity reports (definition, relationships, assertions, and keys)
  - Relationship definitions
  - Key attribute list and definitions
  - Relationship (entity cross-reference)
  - Key attribute/entity cross-reference

- Phase Four Kit
  - Kit cover sheet
  - Phase Four (attribute-level) diagrams

- Entity reports (definition, relationships, assertions, keys and attributes)
- Relationship definitions
- Attribute list and definitions (key and nonkey)
- Relationship/entity cross-reference
- Attribute/entity cross-reference (key and nonkey)

5.3 Standard Forms

An appropriate cover sheet distinguishes the material as a kit. The cover sheet has fields for author, date, project, document number, title, status, and notes. Complete one Cover Sheet for each kit submitted and fill in the following fields on the Cover Sheet (See Figure 5-2).

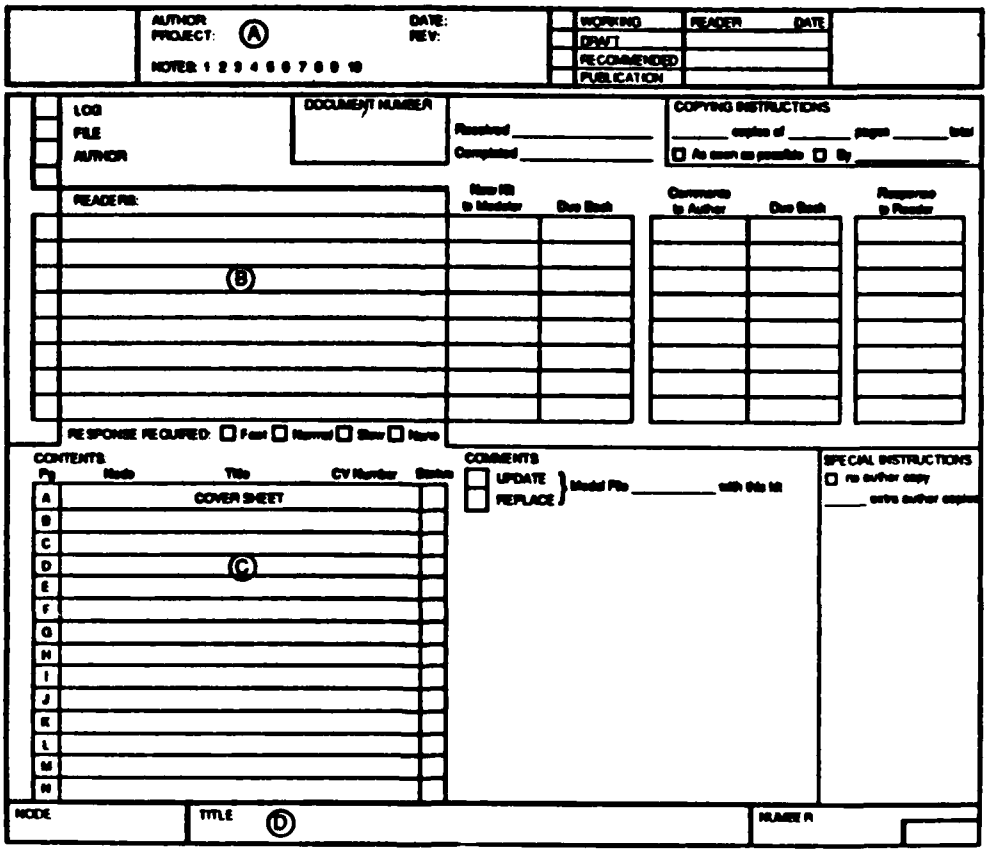


Figure 5-2. Kit Cover Sheet

- Working Information (Figure 5-2 note A)
  - Author or team generating the model
  - Project name and task number
  - Date of original submission to library
  - Dates of all published revisions
  - Status of the model, either working, draft, recommended for acceptance, or publication as final model.
  - Reader signature and date after his/her review
- Reviewer Information (Figure 5-2 note B)
  - Filing and copying information
  - List of kit reviewers
  - Schedule date for various stages of kit cycle
- Content Information (Figure 5-2 note C)
  - Table of contents for the kit
  - Status of each kit section
  - Comments or special instructions to librarian
- Identification Information (Figure 5-2 note D)
  - Model name ("Node") e.g. MFG-1
  - Title of the model
  - Page number

Standard Diagram Form

The Standard Diagram Form (Figure 5-3) has minimum structure and constraints. The sheet supports only the functions important to the discipline of structured analysis:

- Establishment of context
- Cross-referencing between diagrams and support pages
- Notes about the content of each sheet

The diagram form is a single standard size for ease of filing and copying. The form is divided into three major sections:

- Working information (Figure 5-3 note A)
- Message field (Figure 5-3 note B)
- Identification fields (Figure 5-3 note C)

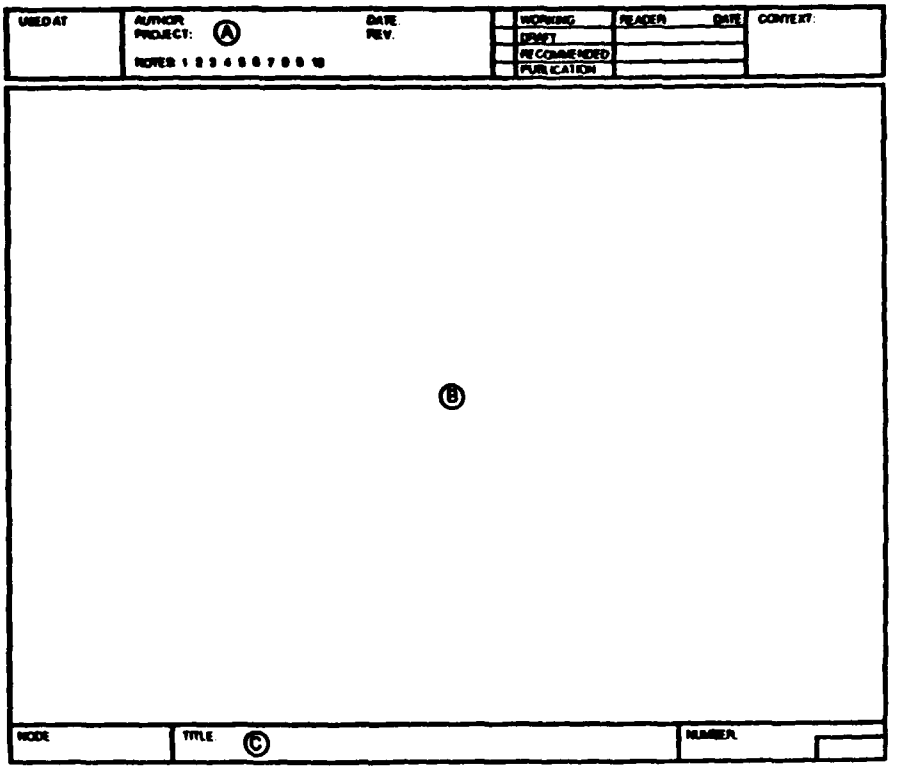


Figure 5-3. Standard Diagram Form

The form is designed so that the working information at the top of the form may be cut off when a final approved-for-publication version is completed. The Standard Diagram Form should be used for everything created during the modeling efforts including preliminary notes.

- The Author/Date/Project Fields

  This tells who originally created the diagram, the date it was first drawn, and the project title under which it was created. The Date Field may contain additional dates, written below the original date. These dates represent revisions to the original sheet. If a sheet is released without any change, no revision date is added.

- The Notes Field

  This provides a check-off for notes written on the diagram

sheet. As comments are made on page, the notes are successively crossed out. This provides a quick check for the number of comments.

- The Status Field

  The status classifications provide a ranking of approval.

  Working: The diagram is a major change, regardless of the previous status. New diagrams are working copy.

  Draft: The diagram is a minor change from the previous diagram, and has reached some agreed-upon level of acceptance by a set of readers. Draft diagrams are those proposed by a task leader, but not yet accepted by a review meeting of the technical committee or coalition.

  Recommended Both this diagram and its supporting text have been reviewed and approved by a meeting of the technical committee or coalition, and this diagram is not expected to change.

  Publication: This page may be forwarded as is for final printing and publication.

- The Reader/Date Field

  This is where a commenter initials and dates each form.

- The Context Field

  This field is not used when developing IDEF1X models.

- The Used at Field

  This is a list of diagrams that use this sheet in some way.

- The Message Filed

  The Message Field contains the primary message to be conveyed. In IDEF1X, this field may contain diagrams, function views, definitions, matrices, indexes, etc. The au-

thor should use no paper other than diagram forms. A standard matrix diagram as shown in Figure 5-4 can be used for a variety of purposes.

- The Title Field

  The Title Field contains the name of the material presented on the Standard Diagram Form. If the Message Field contains an entity diagram, the contents of the Title Field must precisely match the title of the subject entity.

- The Number Field

  This field contains all numbers by which this sheet may be referenced. Which includes the following:

  - C-Number

    The C-number is composed of the author's initials followed by a number sequentially assigned by the author. This C-number is placed in the lower left corner of the Number Field and is the primary means of reference to a sheet. Every diagram form used by an author receives a unique C-number. When a model is published, the C-number may be replaced by a standard sequential page number (e.g., pg. 17).

  - Page Number

    A kit page number is written by the librarian at the right-hand side of the Number Field. This is composed of the document number followed by a number identifying the sheet within the document.

5.4 The IDEF Model Walk-Through Procedure

In addition to the kit cycle, a walk-through procedure has been developed. This procedure may be used when the participants in building a model can be assembled for commenting:

1. Present the model to be analyzed by using its entity pool. This is the model's table of contents and gives the reviewers a quick overview of what is to come.

2. Present a glossary of terms. This will allow each reviewer to replace personal meanings of words with

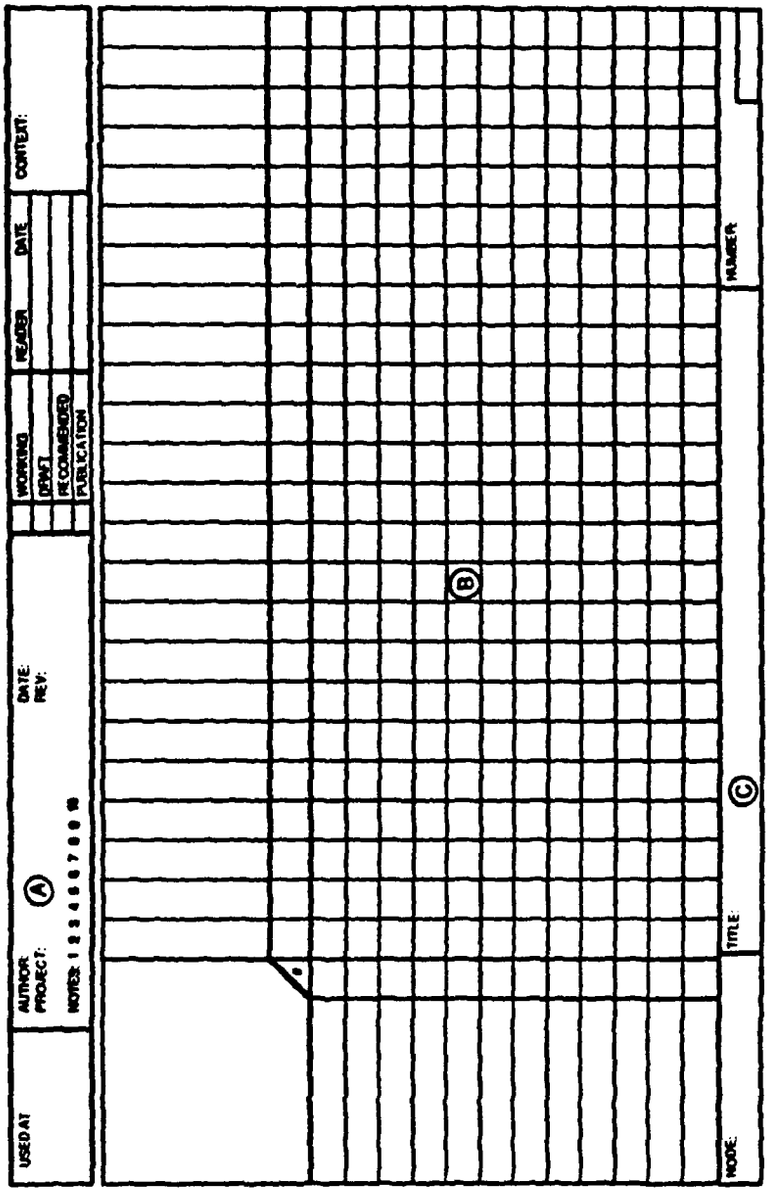


Figure 5-4. Matrix Form

those that the presenting team has chosen. The meanings should not be questioned at this point. A change in meaning would require many changes in the diagrams.

3. Present function view diagrams for review.

The function view walk-through process is an orderly, step-by-step process where questions can be asked that may identify potential weaknesses in the model. Six steps of a structured walk-through follow.

Model corrections may be proposed at any step. These corrections may be noted for execution at a later date or adopted immediately.

Step 1: SCAN THE ENTITY POOL

This step allows the reader to obtain general impressions about the content of the model. Since the entity pool also lists deleted entities, the reader gets a better feel for the evolution of the model to its current state. At this point, the reader should examine the definitions of the entities.

Criteria For Acceptance:

1. The chosen entities represent the types of information necessary to support the environment being modeled.

2. The chosen entities are, in the reviewer's opinion, relevant based on the purpose and scope of the model.

Unless a problem is very obvious, criticism should be delayed until Step 2 below. However, first impressions should not be lost. They might be put on a blackboard or flip chart pad until resolved.

Step 2: READ THE FUNCTION VIEW DIAGRAM

Once the reader understands the entities, the diagram is read to determine if the relationships are accurately represented.

Criteria For Acceptance:

1. The relationship cardinality conforms to the refinement rules defined in the IDEF1X Manual.

2. All required relationships are shown either directly or indirectly.

3. The diagram is structured so it is easy to read (minimal line crossing, related entities are located close to each other).

Step 3: EXAMINE THE KEY ATTRIBUTES

This step serves to verify that the specified key will in fact uniquely identify one instance of an entity. The reader verifies that all members/attributes of the primary key are necessary.

Criteria For Acceptance:

1. The values of the primary key attributes in combination uniquely identify each instance within the entity.

2. The primary key attributes are not in violation of the No-Null and No-Repeat rules.

Step 4: EXAMINE THE KEY ATTRIBUTE MIGRATION

This step examines the migration of primary keys from the parent to the child entities.

Criteria For Acceptance:

1. The primary key migration conforms to the modeling rules.

2. The owner entity of all foreign keys are present in the model.

3. Primary key migration is consistent with the relationship.

Step 5: EXAMINE NONKEY ATTRIBUTES

The attributes that are not members of the primary key are analyzed for each entity.

Criteria For Acceptance:

1. The attributes do not violate the No-Null and No- Re-

peat rules.

2. The attributes serve to capture information that is within the scope of the model.

3. Each attribute is unique within the model.

Step 6: SET THE STATUS OF THE DIAGRAM

1. Recommended as it stands.

2. Recommended as modified.

3. Draft: Too many changes made, a redraw is necessary, and future review is required.

4. Not Accepted: A complete re-analysis is required.

APPENDIX A

IDEF1X GLOSSARY

Acceptance Review Committee

One of the members of the functional organization whose responsibility is to provide guidance and arbitration over the modeling efforts and to pass final judgment over the completed product (i.e., model acceptance).

Assertion

A statement that specifies a condition that must be true.

Attribute

A characteristic or element of data describing something about an entity. An attribute is given a specific name denoting its meaning (e.g., hair color) and a value (e.g., brown).

Attribute, Inherited

An attribute that is the primary key (or part of the primary key) of another entity. It migrates from that entity because of a relationship between the entities. Also called a migrated attribute.

Attribute, Migrated

Same as Inherited Attribute.

Attribute, Owned

An attribute that is not inherited. Ownership is relative to an entity. An attribute can be owned by only one entity.

Attribute Population

That effort by which "ownership" of attribute classes is determined.

Attribute Role

Describes the function played by an attribute in

describing an entity, including inherited (= migrated), owned, primary key, alternate key.

Attribute Value

The exact data value given to an attribute (e.g., attribute: hair color; attribute value: brown).

Author Conventions

The special practices and standards developed by the modeler to enhance the presentation or utilization of the model. Author conventions are not allowed to violate any methodology rules.

Constraint

An assertion whose purpose is to explicitly specify data meanings.

Constraint, Boolean

A condition that restricts instances of child entities in multiple relationships with the same parent entity. The operator "AND" means the parent must have child entity instances in both relationships. The operator "OR" means the parent may have child entity instances in either or both relationships. The operator "XOR" means the parent may have child entity instances in at most one of the relationships.

Constraint, Cardinality

A limit on the number of occurrences of a child entity that may exist in a relationship to a parent entity.

Constraint, Existence

A condition that an instance of one entity cannot exist unless an instance of another related entity also exists.

Data Collection Plan

The plan which identifies the targets e.g., the functions, the departments, or the personnel, which are the sources of the material used for the development of the model.

Domain

A set of allowable values. A domain may be specified by a datatype (e.g., integer, date, money) and may include constraints on the range of values (e.g., greater than zero; between 2 and 12; 17 characters; from the list 2,5,10,16). A domain may be assigned to one or more attributes.

Entity

A collection of like instances (persons, places, things, or events) that is named by a generic noun, has a key (which will uniquely identify each instance), and has one or more attributes (which will describe each instance).

Entity Diagram

A diagram which depicts a "subject" entity and all entities directly related to the subject entity.

Entity Instance

An occurrence of a named entity. It can be specifically identified by the value of its key. Once the instance is determined, the values of all of the other attributes of that instance are also known.

Entity, Category

An entity whose instances are subclassifications of instances of another entity which represents the same real-world thing. All attributes of the generic entity also pertain to the category entity. For example, "salaried employee" is a category entity of the generic "employee".

Existence Dependency

A constraint between two entities indicating that instances of the dependent one cannot exist without being related to an instance of the other. Existence dependency is referential integrity plus the constraint that the foreign key cannot have a null value.

Expert Reviewer (Commenter)

One of the members of the modeling team whose expertise is focused on some particular activity within the manufacturing

enterprise, and whose responsibility it is to provide critical comments on the evolving model.

FEO

An acronym meaning For Exhibition Only; it is one vehicle by which supportive or explanatory information is provided for the model, via some combination of drawings, text, etc.

Functional Dependency

A constraint between two attributes indicating that the value of one is determined by the value of the other.

IDEF Kit Cycle

The regular interchange of portions of the model in development between the modeler and the readers/expert reviewers, the purpose of which is the isolation and detection of errors, omissions, and misrepresentations.

IDEF1X Model

A graphic representation of data meanings in an environment. It displays the basic structure and relationships of data. The product of using the extended ICAM Definition Language for information/data modeling (IDEF1X).

Identifier Dependency

A constraint between two entities that requires the foreign key in the dependent entity to be (part of) its primary key. Identifier dependency is a stronger form of existence dependency.

Key

An attribute, or combination of attributes, of an entity whose values uniquely identify each entity instance.

Key, Alternate

A key other than the primary key of an entity.

Key, Composite

A key comprising two or more attributes.

Key, Compound

Same as Key, Composite.

Key, Foreign

Attributes that appear in a dependent entity and also as the primary key in another entity.

Key, Member

An attribute that is part of a composite key.

Key, Migrated

Same as Foreign Key.

Key Migration

The process of placing the primary key of a parent/or generic entity in the child or category entity in a relationship.

Key, Primary

The key selected for migration for all relationships in which the entity participates as a parent or generic entity.

Modeler (Author)

One of the members of the modeling team whose responsibilities include the data collection, education and training, model recording, and model control during the development of the model; the modeler is the expert on the IDEF1X modeling methodology.

Normal Forms

Conditions reflecting the extent of the refinement in the identification of entities and the placement of attributes into entities in a data model. Each normal form reflects successively tighter control over the relationships between the attributes of an entity.

- First Normal Form (1NF) - there is no more than one value for any attribute in an instance of the entity.

- Second Formal Form (2NF) - 1NF, plus non-key attribute's value is determined by the entity instance's entire key, not by just part of it. An entity in 1NF with a key that is not compound is automatically in 2NF.

- Third Normal Form (3NF) - 2NF, plus no non-key attribute's value is determined by another non-key attribute's value. An entity in 2NF with only one non-key attribute is automatically in 3NF.

- Fourth Normal Form (4NF) - 3NF, plus non attribute of a compound key of three or more attributes is more closely related to one of the other two attributes of the key than to any other. An entity in 3NF whose key contains fewer than three attributes is automatically in 4NF.

- Fifth Normal Form (5NF) - 4NF, plus no attributes can be split off into another entity without introducing new meaning. An entity in 4NF whose key contains fewer than three attributes is automatically in 5NF.

Normalization

The process of refining and regrouping attributes in entities according to the normal forms, making the data meanings more explicit.

Phase Zero

The initial efforts of the modeling activity in which the Context Definition is established i.e., project definition, data collection plan, author conventions, standards, etc.

Phase One

The second in the orderly progression of modeling efforts during which the entities are identified and defined.

Phase Two

The third in the set of orderly progression of modeling efforts during which the entities are identified and defined.

Phase Three

The fourth set in the orderly progress of model development, during which keys are identified and defined.

Phase Four

The fifth effort in the progression of orderly model development during which the "non-key" attributes are identified and defined.

Project Manager

One of the members of the modeling team whose responsibilities include the administrative control over the modeling effort. The duties include: staff the team, set the scope and objectives, chair the Acceptance Review Committee, etc.

Relationship

A logical association between entities.

Relationship Cardinality

The number of entity instances that can be associated with each other in a relationship. See Constraint, Cardinality.

Relationship Name

A phrase-like definition which reflects the meaning of the relationship expressed between the two entities shown on the diagram on which the name appears.

Relationship, Nonspecific

A relationship in which neither entity can be said to be independent of or existence dependent on the other. Examples are many-to-many relationships and zero-or-one-to-many relationships.

Relationship, Specific

A relationship in which one entity is existence-dependent on the other.

Role Name

A name assigned to a foreign key that appears more than once in an entity.

Schema

A definition of data structure

- Conceptual Schema: A neutral definition of the integrated, shared data within an enterprise. It is represented by a semantic data model which conforms to the rules of refinement, and is in fifth normal form.
- External Schema: Describes an application's or end-user's perspective of shared data.
- Internal Schema: Describes a DBMS's physical representation of shared data.

Semantics

The meanings of words and sentences in a language, or of constructs in a model. Contrast with Syntax.

Source(s)

One of the members of the modeling team whose responsibility it is to provide the elements of information (documents, forms, procedures, knowledge, etc.) on which the development of the model will commence and continue.

Syntax

Grammar. A set of rules for forming meaningful phrases and sentences from words in a vocabulary. Contrast with Semantics.

Validation

An effort which results in the informed consensus of the experts who are knowledgeable about the model; the model is considered "valid" if the majority of experts agree that it appropriately and completely represents the area of concern.

APPENDIX B

COMPARISON OF IDEF1X WITH IDEF1

1. Terminology

The recommended IDEF1-Extended terminology is slightly different from the IDEF1 terminology. The new terminology is more consistent with the terminology used by the data modeling community at-large. It is also more consistent with the way that IDEF1 users actually refer to the model constructs.

The following table shows the correspondence in terms for concepts that occur in both IDEF1-Extended and IDEF1.

IDEF1-Extended	IDEF1
entity	entity class
attribute	attribute class
relationship	relation class
candidate key	alternate key class
primary key	key class
primary key with alternate key(s)	alternate key classes
foreign key	migrated key class
entity instance	entity
attribute value	attribute
relationship instance	relationship

The rest of this appendix uses the recommended IDEF1-Extended terminology, even when referring to constructs of an IDEF1 model.

The IDEF1-Extended terminology is more consistent with the evolving industry-standard vocabulary of the data resource

management field. The relational model (developed by IBM and others) and the entity-relationship model (developed by UCLA and others) use the IDEF1-Extended terms.

IDEF1-Extended gives the simpler terms to the concepts that are used more often. For example, modelers commonly deal with entities, but only rarely with entity instances.

2. Entity Syntax

The entity-related semantic constructs that IDEF1-Extended supports are:

2.1 Entities

2.2 Identifier-independent entities vs. identifier-dependent entities

2.3 Entity names

2.4 Entity numbers

The following paragraphs first indicate how IDEF1 supports these constructs, then discusses the similarities or differences in the IDEF1-Extended approach.

2.1 Entities

IDEF1 represents entities by rectangular boxes, as does IDEF1-Extended.

2.2 Identifier-Independent vs. Identifier-Dependent Entities

IDEF1-Extended distinguishes between identifier-independent entities, which depend on no other entities for their identification, and identifier-dependent entities, which do depend on other entities for their identification (and existence).

By contrast, IDEF1 uses the same symbol (a rectangular box with square corners) for both constructs. IDEF1-Extended's syntax allows the identifier-independent entities to be more prominent in a data model diagram.

Identifier-independent entities are drawn in IDEF1-Extended as rectangular boxes with square corners. Identifier-

dependent entities are drawn as rectangular boxes with rounded corners.

Note that in IDEF1-Extended, entity identifier independence/dependence is relative to the entire model, not just relative to a particular relationship.

2.3 Entity Names

IDEF1's entity label is placed inside the entity box. The entity name is provided as part of the definition.

By contrast, IDEF1-Extended's entity name is placed above the entity box, as it applies to everything in the box. This convention allows modelers to quickly sketch model diagrams. Entity labels are not used in IDEF1X since entity names may use abbreviations.

2.4 Entity Numbers

IDEF1's entity-numbering scheme blocks out a significant portion of an entity box. The entity number is placed in the upper left corner of the box, with a diagonal line setting off the area.

By contrast, IDEF1-Extended's entity number is placed such that it occupies no space in the entity box. It follows the entity name, and is separated from the name by a "/".

3. Attribute Syntax

The attribute-related semantic constructs that IDEF1-Extended supports are:

3.1 Attributes

3.2 Candidate-key attributes

3.3 Primary-key attributes

3.4 Foreign-key attributes

3.5 Role names

3.1 Attributes

IDEF1 places attributes within entity boxes, as does

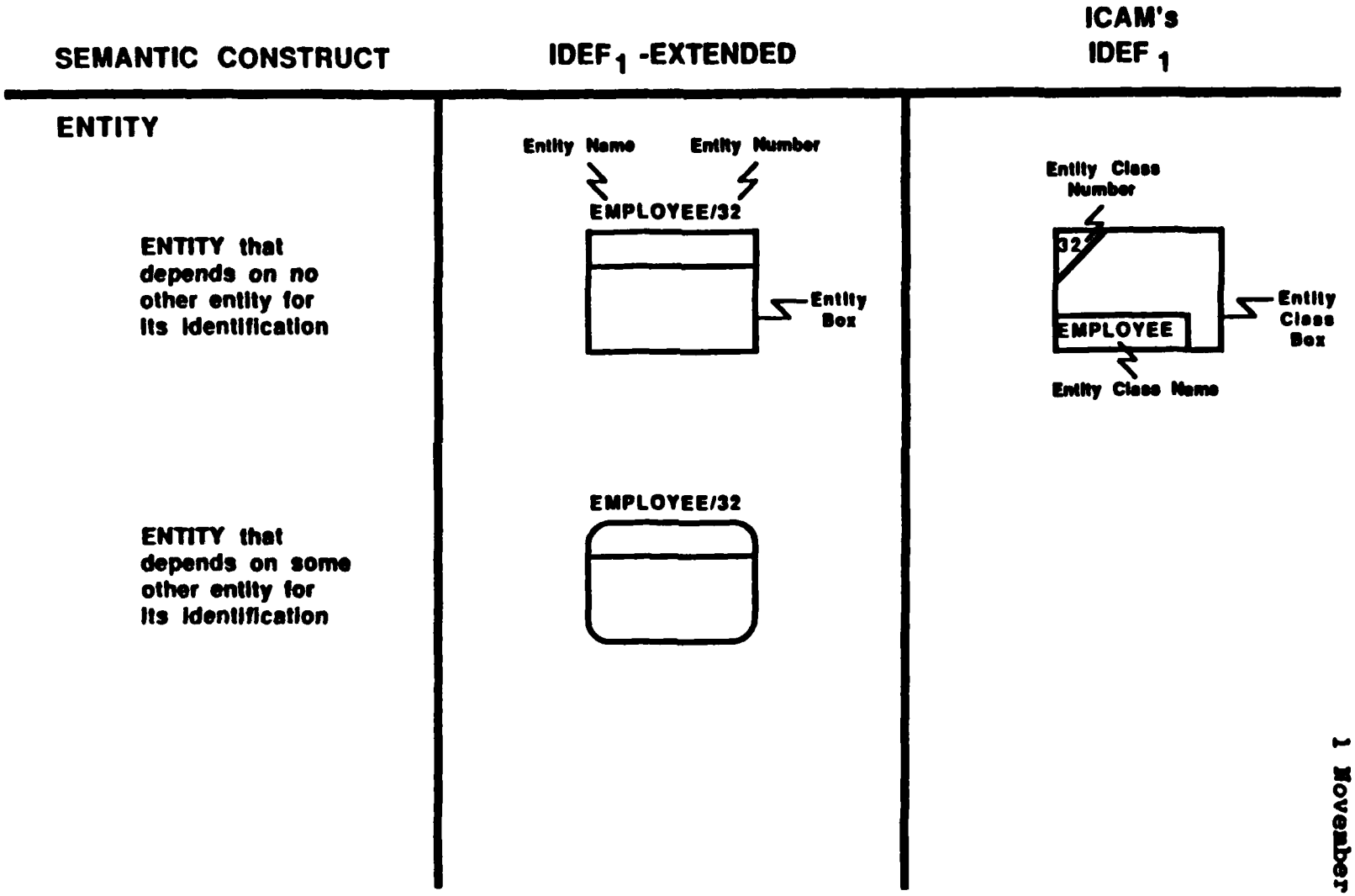


Figure B-1. IDEF1X vs. IDEF1 Entities

IDEF1-Extended. Attribute names are used in IDEF1-Extended in the place of attribute labels used in IDEF1.

3.2 Candidate-Key Attributes

A candidate key is one or more attributes, whose values uniquely identify entity instances. IDEF1 underscores candidate-key attributes. If there is more than one candidate key, then each is enclosed in parentheses. If the candidate keys are compound and overlap, then the attribute that appears in multiple keys appears multiple times in the entity.

In contrast to IDEF1, IDEF1-Extended requires that one of the candidate keys be designated the primary key. This key is the one that is migrated through relationships to other entities. The other candidate keys are called alternate keys. Designation of a primary key is necessary for automated normalization and consistency checking.

IDEF1-Extended marks each attribute of an alternate key with an alternate key number following the attribute name: (AKn). An attribute may be a component of multiple alternate keys, and therefore have multiple alternate key numbers. For example, consider the following three attributes:

DRAWING-# (AK1)
REVISION-# (AK1, AK2)
PART-# (AK2)

One of the alternate keys is DRAWING-#, REVISION-#. This key is designated as AK1. The other alternate key is PART-#, REVISION-#. This key is designated as AK2. REVISION-# is a components of both alternate keys. As an attribute, REVISION-# appears only once in the entity. By contrast, IDEF1 would show REVISION-# twice in the entity, once for each of the alternate keys.

3.3 Primary-Key Attributes

IDEF1 does not distinguish a primary key from any other candidate key. All candidate-key attributes are underscored.

By contrast, IDEF1-Extended places primary-key attributes above a line dividing the entity box. IDEF1-Extended's main advantage here is a visual one. The most important attributes are easily distinguished, because they are obviously separated from the non-key attributes. Automated graphing support also

becomes less device-dependent if underscores are not used in the graphic syntax.

3.4 Foreign-Key Attributes

A foreign-key attribute is an attribute that is part of the primary key of another entity.

IDEF1 designates a foreign-key attribute (called a migrated key class) by its having the same name as where it appears in a candidate key. IDEF1 attributes must have identical names to be detected as foreign-key - primary-key pairs.

By contrast, IDEF1-Extended marks foreign-key attributes by following the name with (FK). This makes it clear which attributes have migrated in from other entities. A foreign-key attribute may be a primary-key attribute, an alternate-key attribute, or a non-key attribute. A foreign-key attribute is always part of the primary key in the entity from which it migrates.

3.5 Role Names

IDEF1 does not support role names. If an attribute migrates in through two relationships, then it appears twice with the same name in each entity.

IDEF1-Extended supports role names for attributes where "second-names" better indicate their meaning. These attributes are always foreign-key attributes. They commonly are attribute that migrate in through multiple relationships, appear twice in the entity and need to be distinguished. Automated normalization requires that they be given different names to indicate that they have different meanings.

For example, consider a bill-of-materials structure between entities PART and COMPONENT. PART's primary-key attribute PART# migrates into COMPONENT twice, once through the relationship IS-IN and once through the relationship HAS. In an IDEF1 model PART# would appear twice in COMPONENT. By contrast, IDEF1-Extended supports giving rolenames to these appearances of PART#: one appearance could be named COMPONENT-#.PART#; the other appearance could be named ASSEMBLY-#.PART#. Software support then is able to detect the two roles for the foreign-key and not interpret them as being redundant.

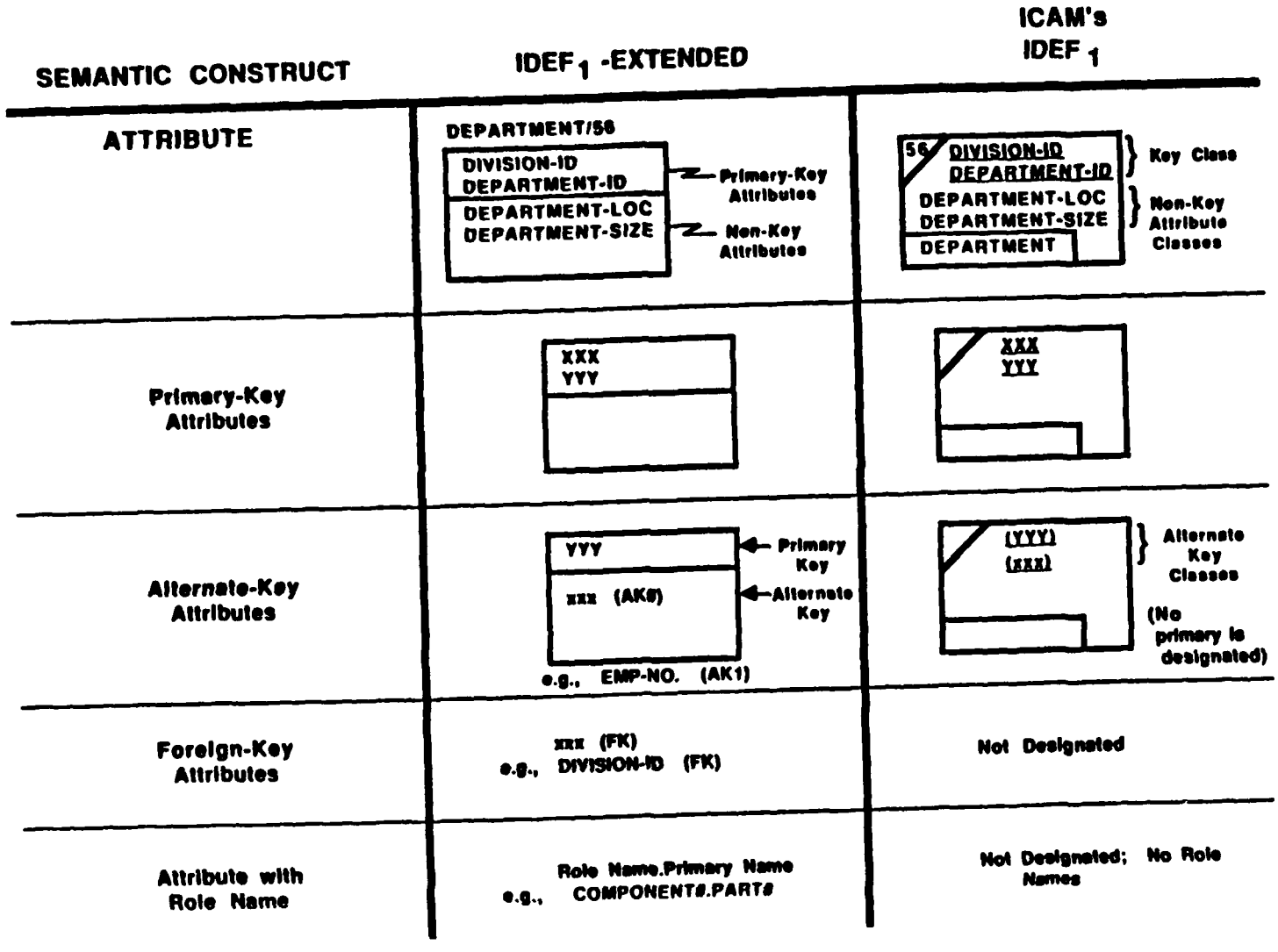


Figure B-2. IDEF1X vs. IDEF1 Attributes

4. Relationship Syntax

The relationship-related semantic constructs that IDEF1-Extended supports are:

4.1 Connection Relationships

4.2 Categorization Relationships

4.3 Identifier Dependency

4.1 Connection Relationships

An connection relationship is an association between two unlike entities. For example, DEPARTMENT and EMPLOYEE are related by a connection relationship named EMPLOYS. DRAWING and PART are related by a connection relationship named SPECIFIES.

IDEF1 connects the two entities that participate in a connection relationship by a line. The symbols at the ends of the line indicate how many instances of each of the entities can be related to how many instances of the other entity. The symbols are called cardinality symbols. IDEF1-Extended uses basically the same approach.

However, IDEF1 uses two fundamentally different symbols to represent relationship cardinality. Cardinality of zero-or-more is indicated by an open diamond; cardinality of zero-or-one is indicated by a half-diamond on the line. Because it looks very much like an arrow, the half-diamond can be misinterpreted as showing data flow.

By contrast, IDEF1-Extended always uses a "big-dot" symbol to represent the cardinality of relationships. The big-dot is annotated to indicate exact cardinality. Unadorned, the big-dot means "zero, one or many". A "p" indicates positive (i.e., one-or-more); a "z" indicates zero-or-one; an "n" indicates a specific number (i.e., = n). IDEF1-Extended uses the big-dot symbol because it is the largest single-character symbol and is not distorted by photo-reduction. It also is easy to draw freehand.

4.2 Categorization Relationships

IDEF1 does not represent categorization relationships in a satisfactory way. It uses the same syntax for a categorization relationship that it uses for a connection relationship with cardinality of zero-or-one. IDEF1 relies on the modeler to assign a relationship name of "can be" to distinguish the categorization from a connection. A category entity typically is discovered through application of the IDEF1 "No-Null Rule".

Additionally, IDEF1 is not able to bundle together the category entities for a generic entity. An IDEF1 model can indicate that an EMPLOYEE "can be" zero-or-one HOURLY-EMPLOYEE, and an EMPLOYEE "can be" zero-or-one SALARIED-EMPLOYEE, and an EMPLOYEE "can be" zero-or-one UNCLASSIFIED-EMPLOYEE. It cannot indicate that an EMPLOYEE instance "can be" only one of these categories.

By contrast, IDEF1-Extended uses a line with an open circle to represent a categorization relationship. The graphics of a categorization relationship are obviously different from those of a connection relationship.

The discriminator attribute of the generic entity appears across the circle. The discriminator's value determines which of the category entities exists for this particular generic entity instance.

If the complete set of categories is represented, then the categorization circle has a double baseline. This means that every possible value of the discriminator is represented by a category entity. For example, if there were only three possible EMPLOYEE subtypes, and all three were modeled as category entities, then a double baseline would be used.

However, if an incomplete set of categories is represented, then the categorization circle has a single baseline. This means that the discriminator hay have a value that is not represented by a category entity. For example, if only two of the three possible EMPLOYEE categories were modeled, then a single baseline would be used. This is common when a category entity has no attributes of its own, separate from the generic entity.

A categorization relationship does not have a name shown on the diagram. When the relationship is read, the name "can be" is used.

4.3 Identifier Dependency

An identifier dependency occurs when the migrated key attributes become (part of) the primary key in the child entity. The identification of the child depends on the key of the parent entity. Thus, the child entity is identifier dependent on the parent. It is completely dependent on the parent and cannot exist without the parent.

IDEF1 implies identifier dependency when foreign-key attributes (i.e., the migrated-key attributes) are underscored.

By contrast, IDEF1-Extended makes identifier dependencies obvious in the model graphics. It represents a relationship with identifier dependency by a solid relationship line and a relationship without identifier dependency by a dashed relationship line. Identifier dependency is a characteristic of the relationship between two entities, therefore is represented by the graphics of the relationship.

Categorization relationships always have identifier dependency. Connection relationships may or may not have identifier dependency.

5. Examples

The following are examples of the similarities and differences between the graphic representations of IDEF1 and IDEF1-Extended. The numbers correspond to the diagrams on the next several pages.

1. Two entities ALPHA and BETA, with a non-identifying one-to-many relationship REL. ALPHA's primary key A becomes a non-key-foreign-key attribute in BETA.

2. Two entities ALPHA and BETA, with an identifying one-to-many relationship REL. ALPHA's primary key A becomes part of BETA's primary key.

3. Two entities ALPHA and BETA, with an identifying one-to-zero-or-one relationship REL. ALPHA's primary key A becomes BETA's primary key.

4. Two entities ALPHA and BETA, with a non-specific relationship REL. An ALPHA instance can be related to many BETA instances. A BETA Instance can be related to

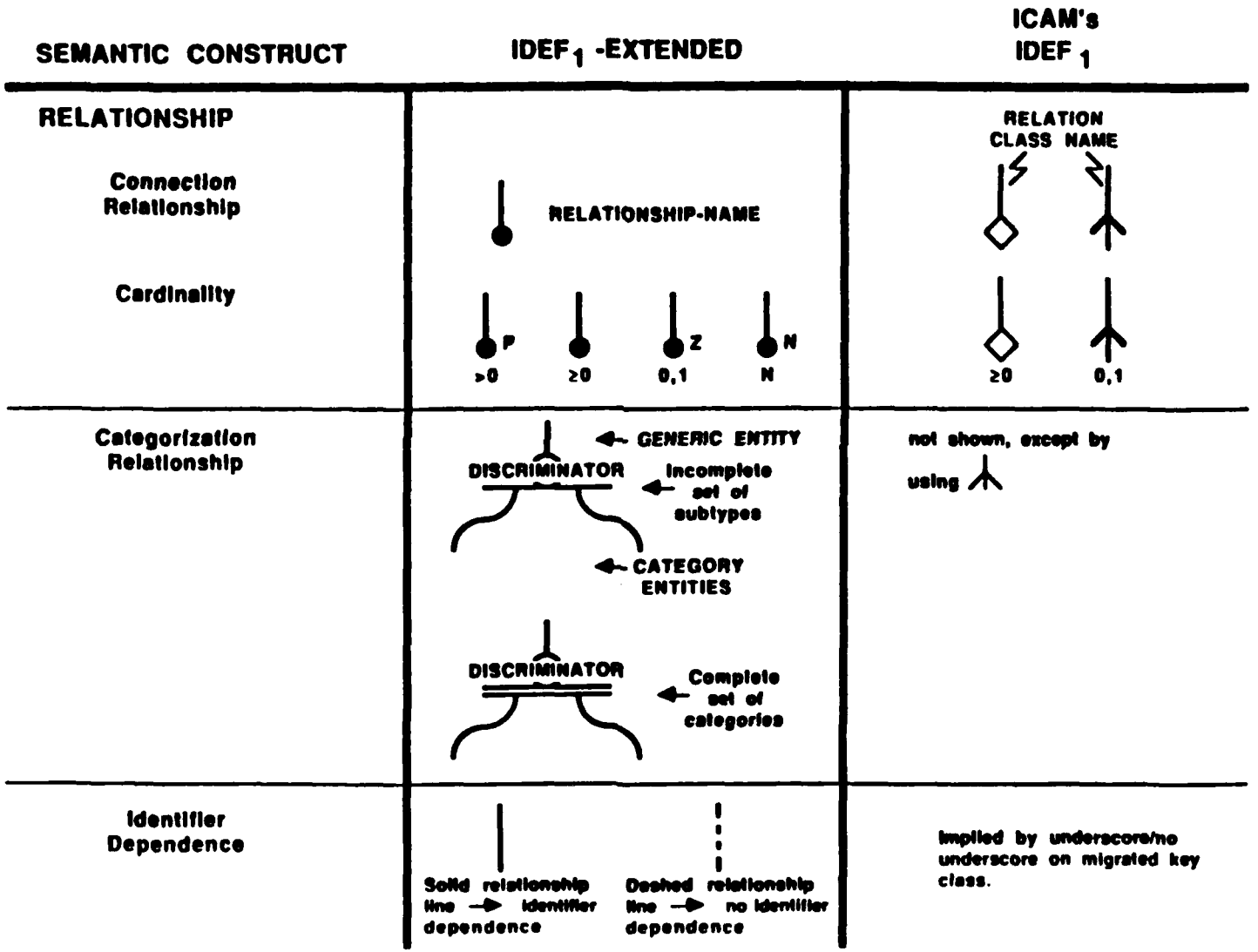


Figure B-3. IDEF1X vs. IDEF1 Relationships

zero-or-one ALPHA instance.

5. Two independent entities ALPHA and BETA, with their dependent intersection entity AB-RES. ALPHA's primary key A becomes a key foreign-key in AB-RES, because relationship R1 is identifying. BETA's primary key B becomes a non-key foreign-key in AB-RES, because relationship R2 is non-identifying.

6. A bill-of-materials structure between independent entity PART and dependent entity COMPONENT. PART's primary key PART# has rolename COMP# where it migrates into COMPONENT through relationship HAS. Both are identifying relationships.

7. EMPLOYEE is a generic entity, with categories SALARIED-EMP and HOURLY-EMP. Every EMPLOYEE instance must have either a corresponding SALARIED-EMP or HOURLY-EMP instance, with the same primary key SOCNO value. (Note that the IDEF1 model does not represent the mutual exclusivity of the two relationships nor the requirement that one must exist.) DEPT# and JOBCODE in EMPLOYEE are both non-key foreign-key

8. PURCHASED-PART has three candidate keys: VENDOR-PART# is the primary key; DWG#,REV# is an alternate key; PART#,REV# is an alternate key.

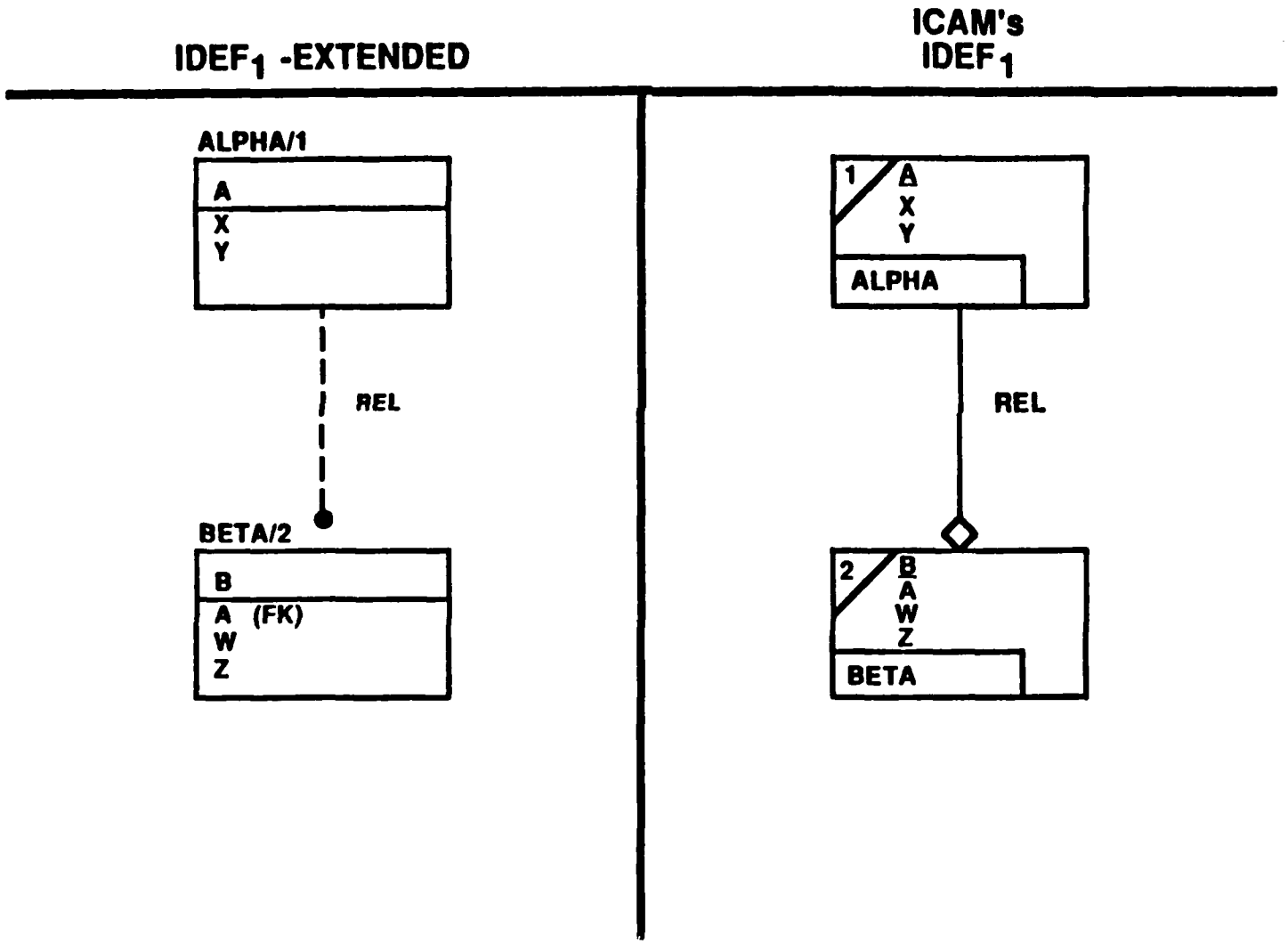


Figure B-4. Example 1

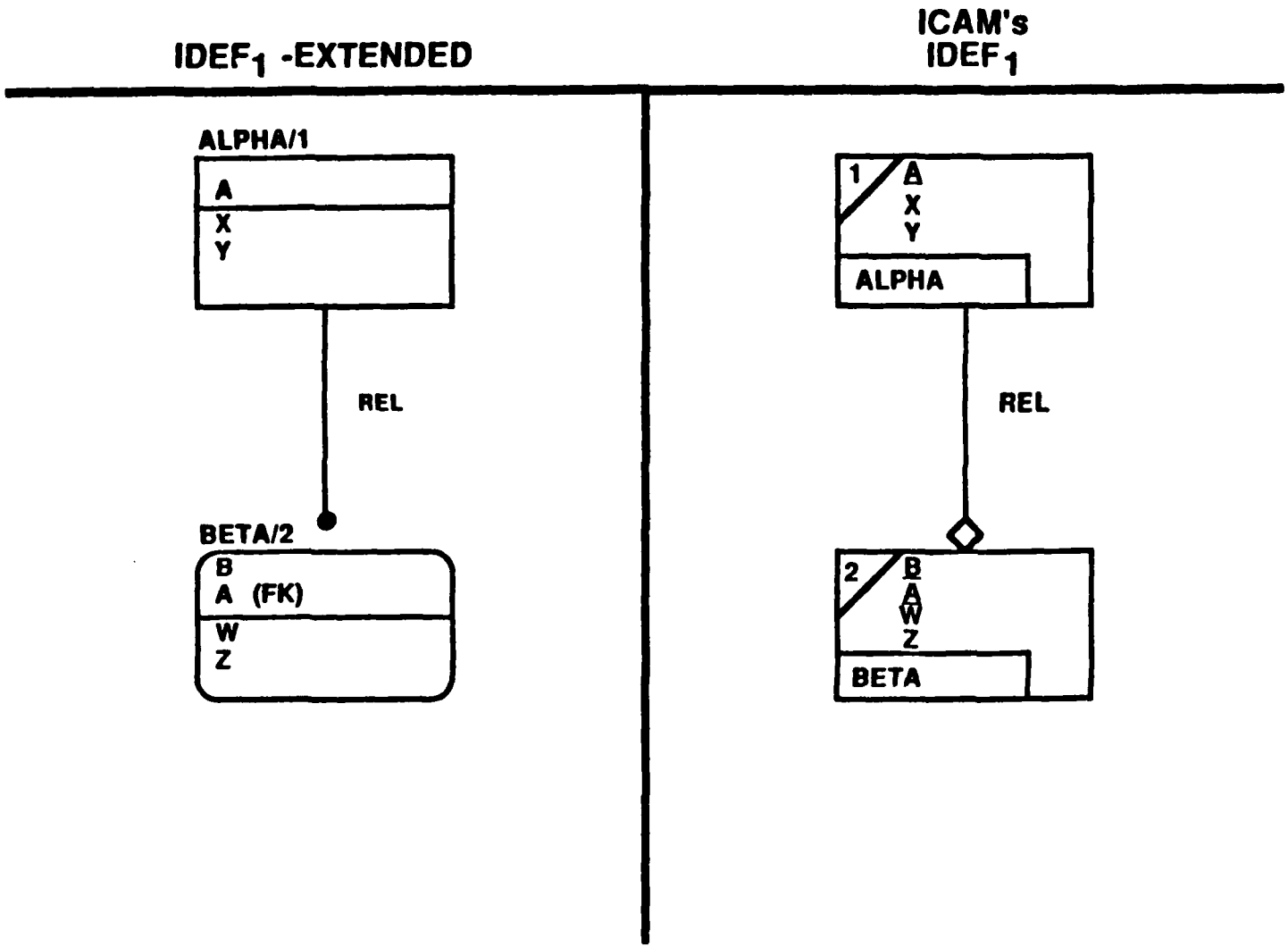


Figure B-5. Example 2

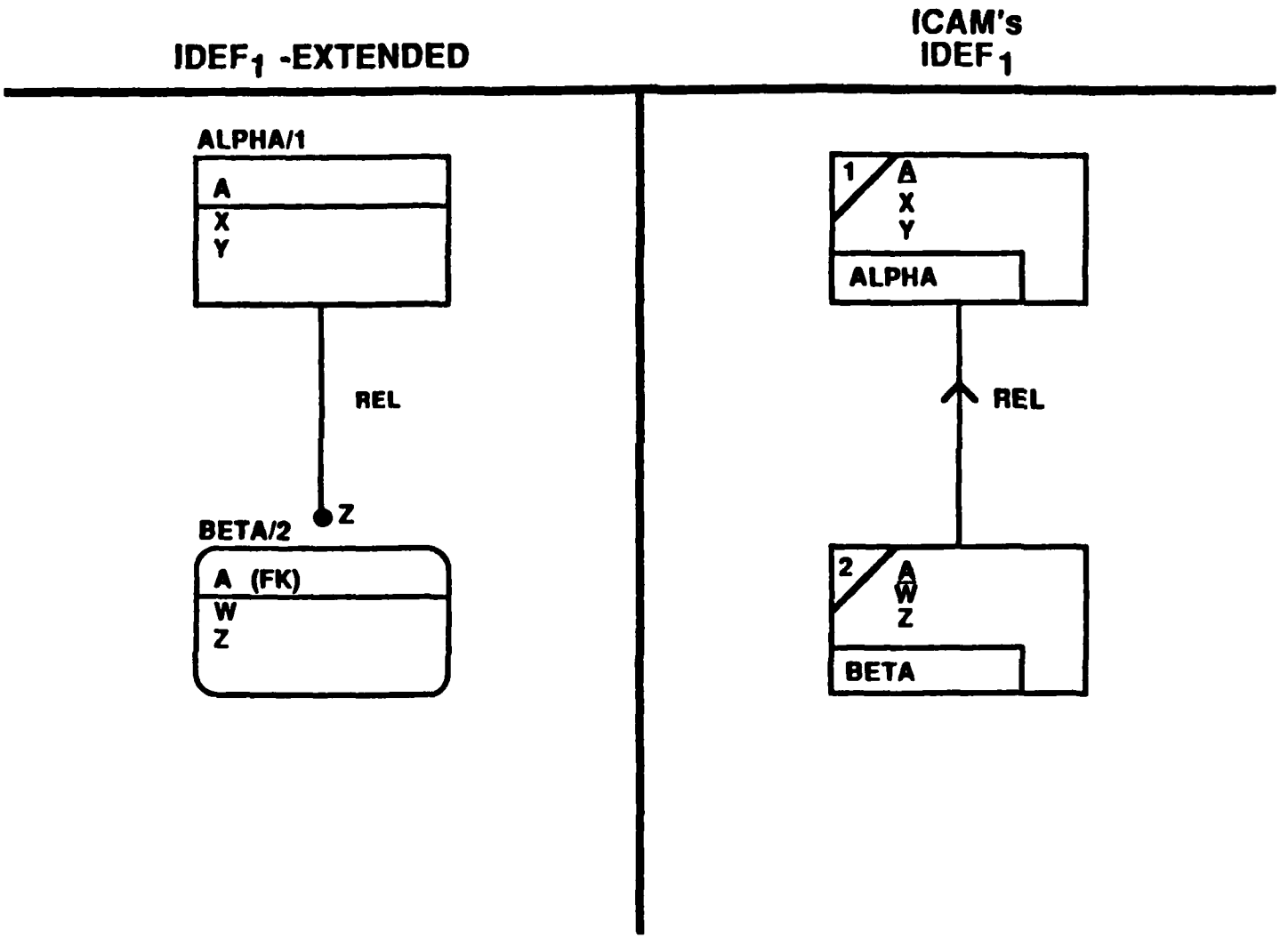


Figure B-6. Example 3

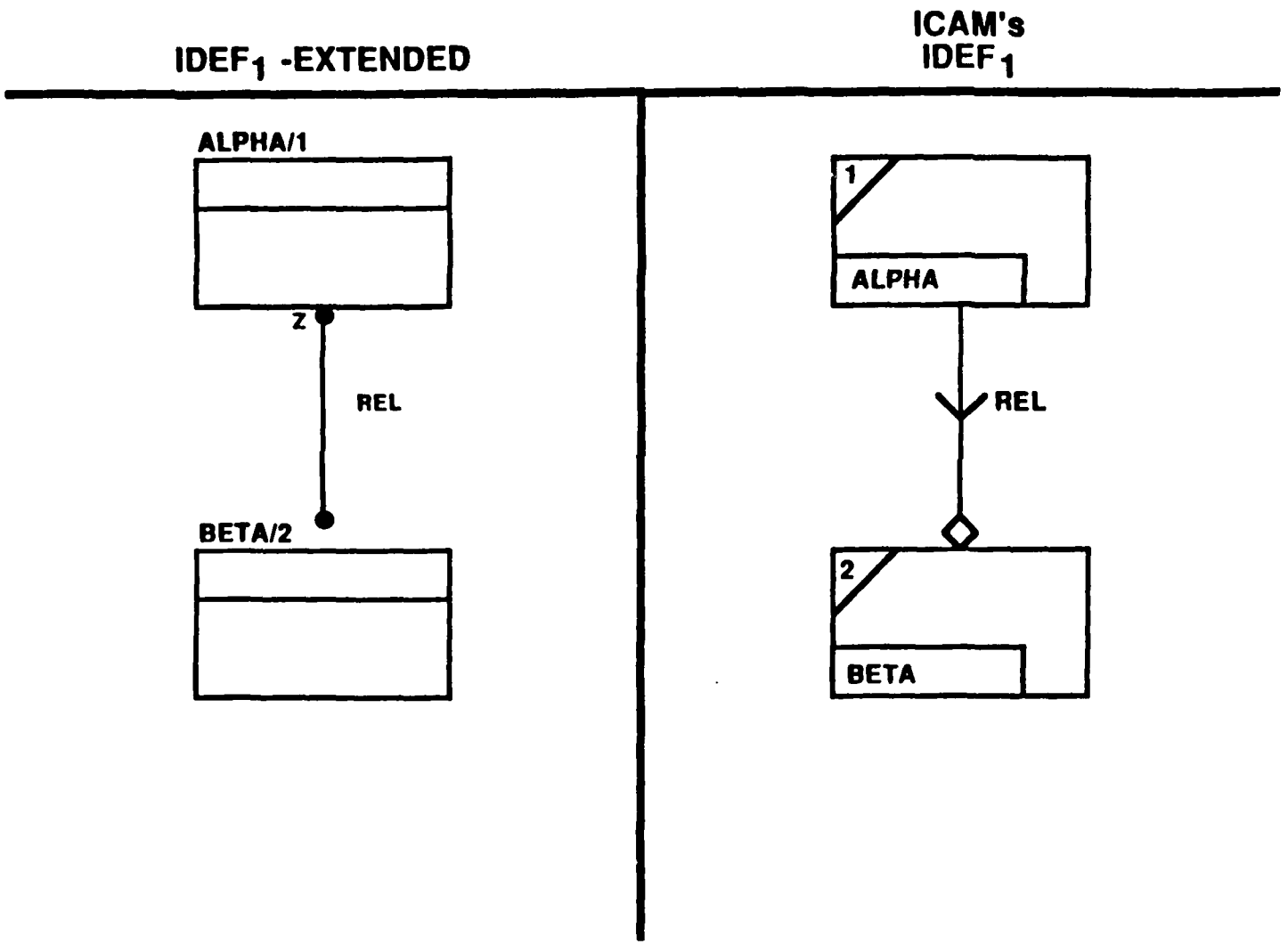


Figure B-7. Example 4

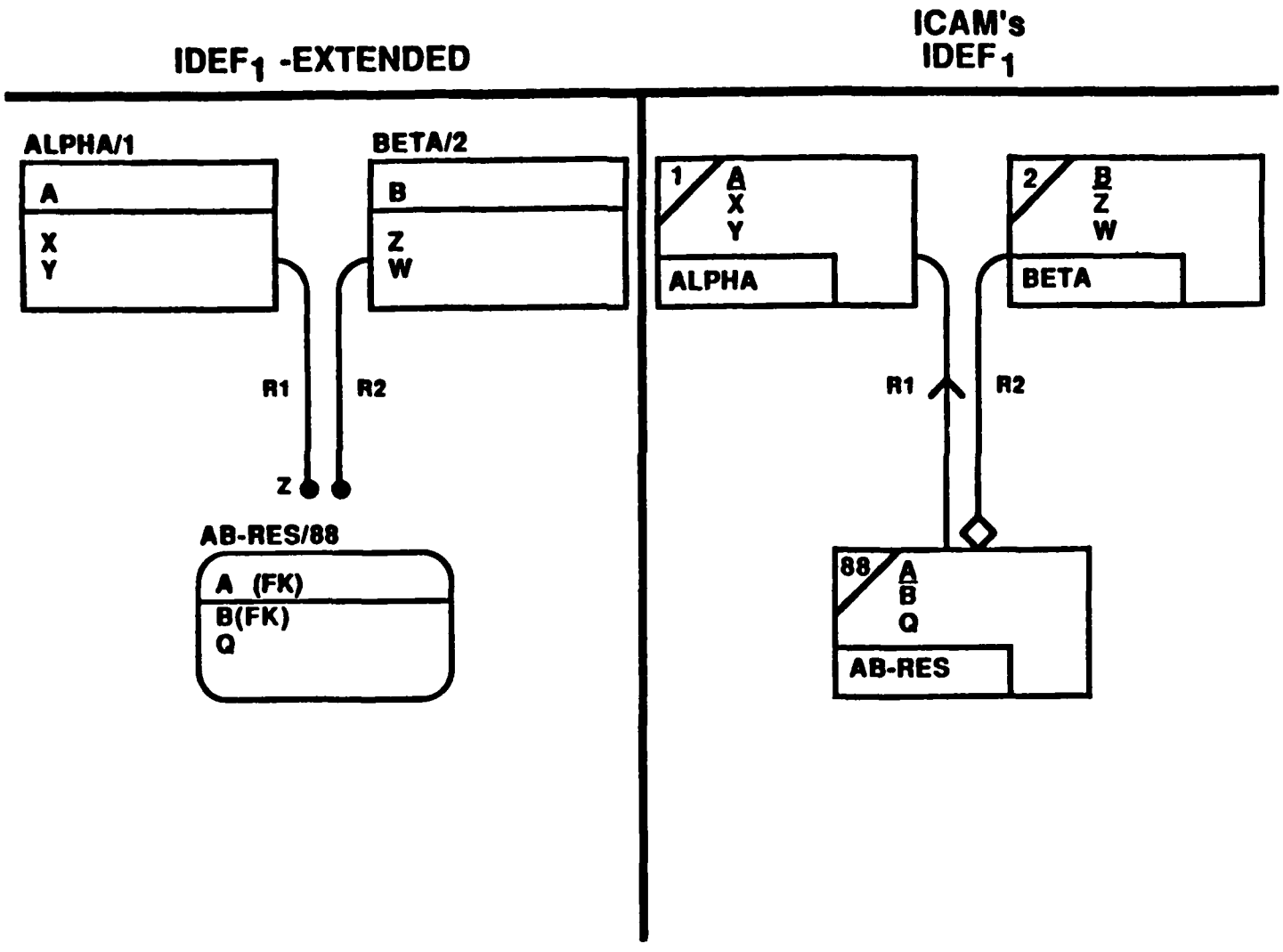


Figure B-6. Example 5

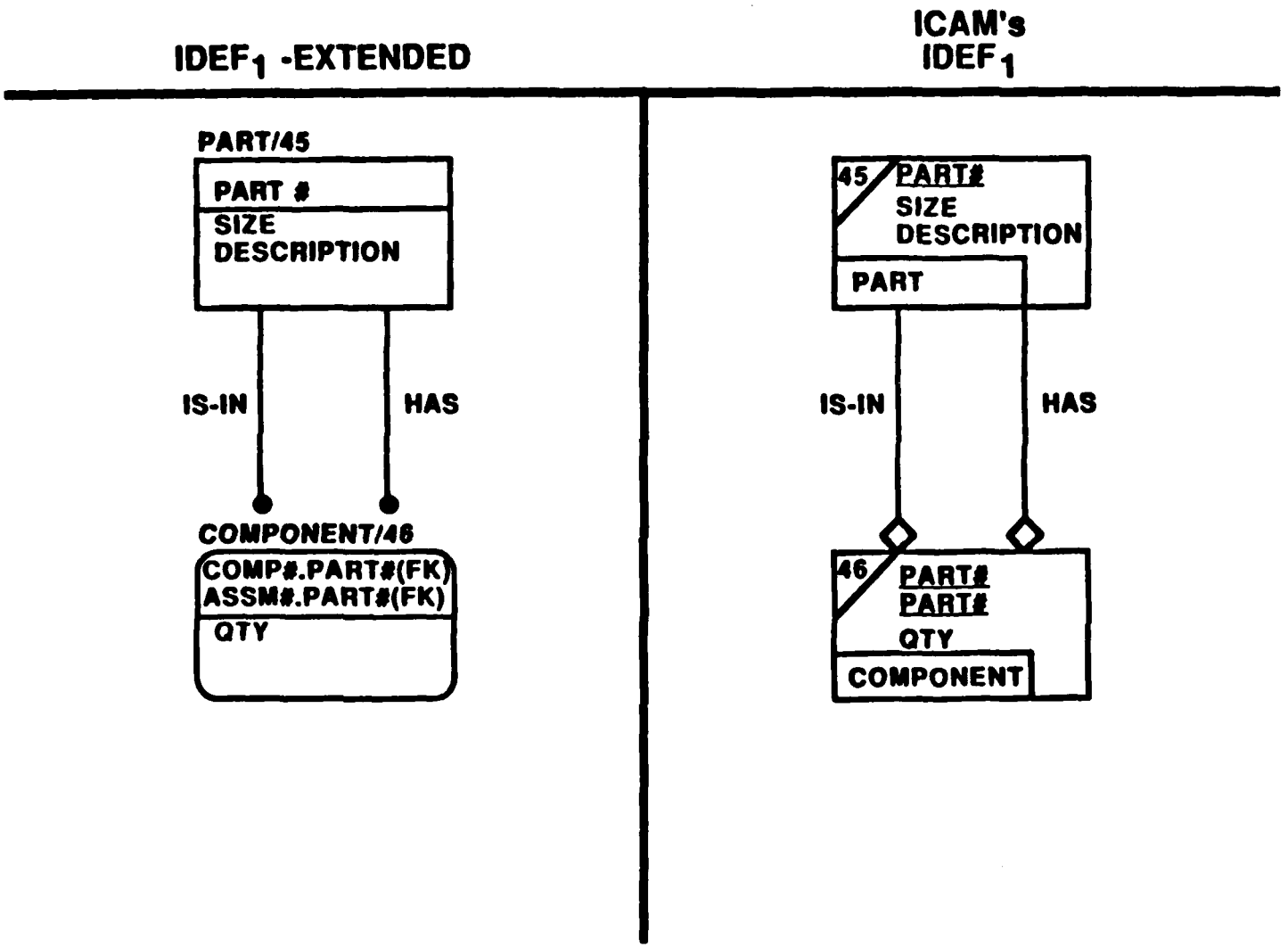


Figure B-9. Example 6

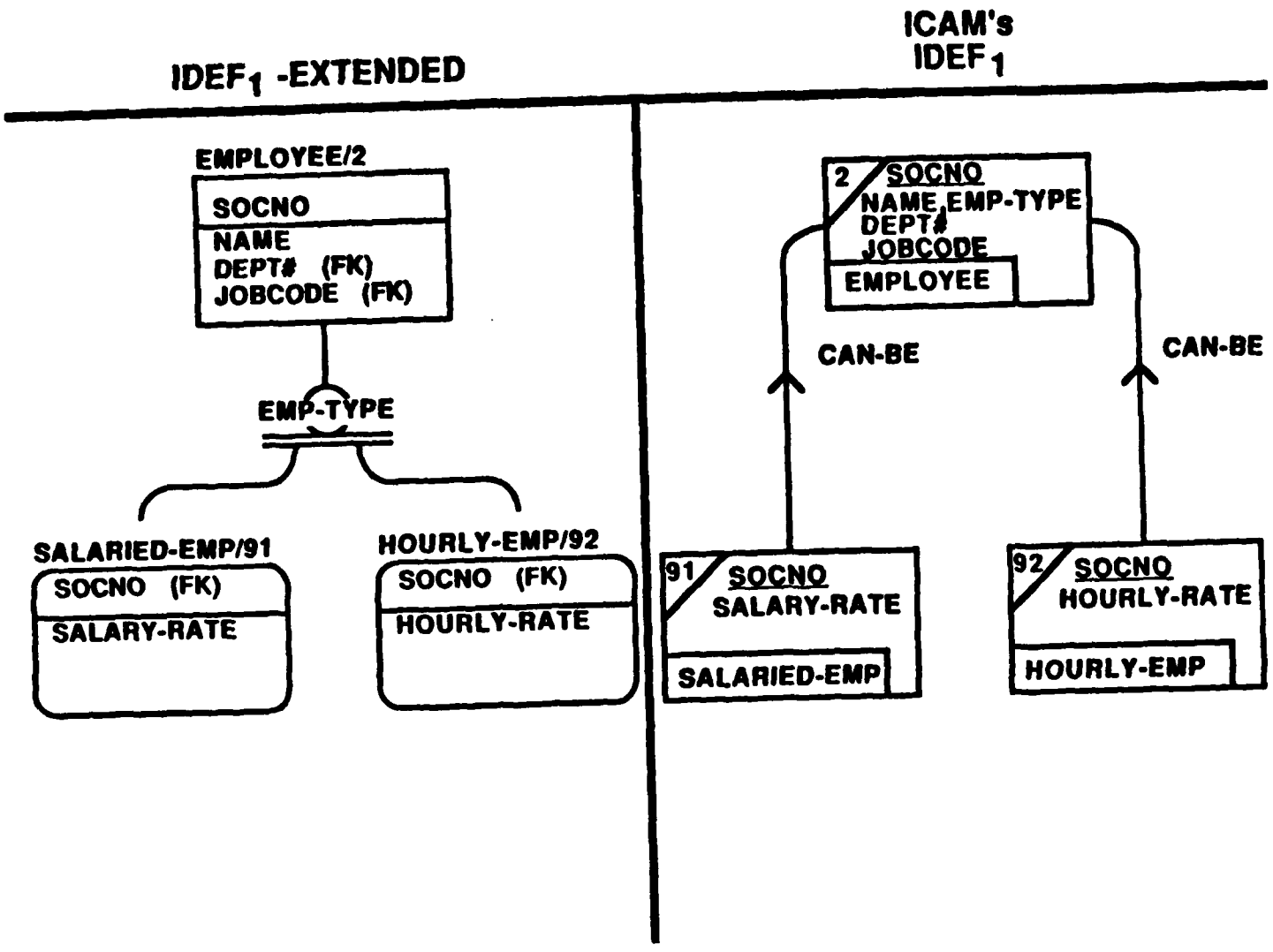


Figure B-10. Example 7

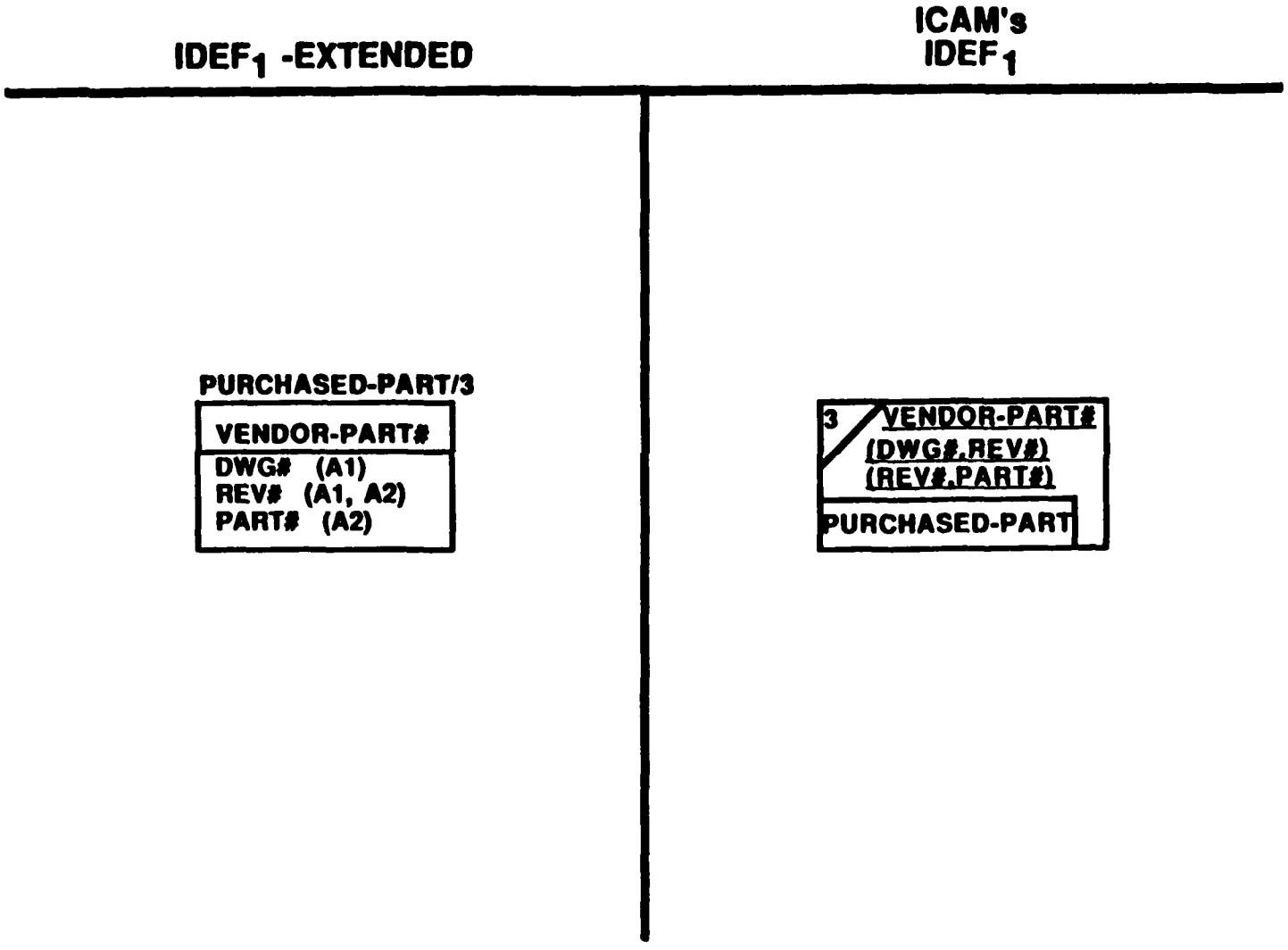


Figure B-11. Example 8

APPENDIX C

REFERENCES

Brown, R.G., Logical Database Design Techniques, The Database Design Group, Inc., Mountain View, CA, 1982.

Chen, P. P-S., "The Entity-relationship Model -- Toward a Unified View of Data," ACM Trans. on Database Systems, Vol. 1, No. 1, March 1976, pp. 9-36.

Codd, E.F., "A Relational Model of Data for Large Shared Data Banks," Communications ACM, Vol. 13, No. 6, June 1970, pp. 377-387.

Codd, E.F., "Extending the Database Relational Model to Capture More Meaning," ACM Trans. on Database Systems, Vol. 4, No. 4, December 1979, pp. 397-434.

D. Appleton Company, Inc., INFO Model-ER Reference Manual, Manhattan Beach, CA 1982.

D. Appleton Company, Inc., Data Modeling Technique, Product Functional Specification, PFS-DMT-3.0, Manhattan Beach, CA 1985.

Hammer, M. and D. McLeod, "The Semantic Data Model: a Modelling Mechanism for Data Base Applications," Proc. ACM SIGMOD Int'l. Conf on Management of Data, Austin, TX, May 31-June 2, 1978, pp. 26-36.

Shipman, D.W., "The Functional Data Model and the Data Language DAPLEX," ACM Trans. on Database Systems, Vol. 6, No. 1, March 1981, pp. 140-173.

Smith, J.M. and D.C.P. Smith, "Database Abstractions: Aggregation," Communications ACM, Vol. 20, No. 6, June 1977, pp. 405-413.

Smith, J.M. and D.C.P. Smith, "Database Abstractions; Aggregation and Generalization," ACM Trans. on Database Systems, Vol. 2, No. 2, June 1977, pp. 105-133.

SofTech., ICAM Architecture Part II-Volume V - Information Modeling Manual (IDEF1), AFWAL-TR-81-4023, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air

Force Systems Command, Wright-Patterson Air Force Base, Ohio 45433, June 1981.

U.S.Government Printing Office: 1987 - 748-061/60873

END

8-87

DTIC