AN ANALYSIS OF DATABASE DESIGN TRANSFORMATIONS AS APPLIED TO THE ADVANCED EDUCATION DATABASE SYSTEM

by

April Michelle Quarles

Project/Report submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

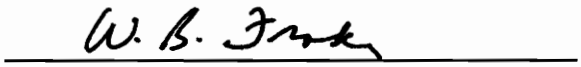
MASTER OF SCIENCE

in

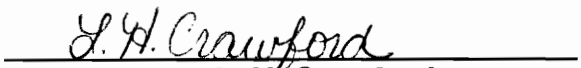
Computer Science

APPROVED:

C. J. Egyhazy, Chairman



W. B. Frakes



L. H. Crawford

May 1993

Blacksburg, Virginia

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

This paper presents an analysis of a database design using three different database methodologies. The database system used as a case study for analysis is the Advanced Education Program Database System. This system is designed using the Formed-Based, Relational, and Object-Oriented approaches to database design. Once these different logical structural designs have been completed, the report presents a series of transformations that can occur between the different design methodologies. Several examples of the transformation techniques are provided to show the variances that exist in the converted designs. A summarization is then made as to the feasibility of using the various transformation techniques that exist.

OBJECTIVE

A database can be defined as a data object that requires a direct access storage medium for processing (Maciaszek, 1990, p. 1). A system based on this definition has the capability of handling large integrated data structures in a shared multi-user fashion. The design of the database is important in order to ensure that system reliability is not hindered due to the magnitude of data that the system supports. There are several methods that can be used to design logical structures for a database system. The following report shows the procedures involved in designing a sample database system using three distinct methodologies -- The Form-Based Approach, The Relational Approach, and The Object-Oriented Approach. The purpose of exploring these different

methodologies is to illustrate the differences and determine the transformations necessary to move one into another.

The following report begins by defining the problem that creates the need for a database system. The System Requirements for the Advanced Education database system are provided along with a procedural description of all three of the above mentioned database design methodologies. The report then proposes and discusses the transformations that can occur between the different methodologies.

In this report, transformations are defined as translations between database designs. These transformations do not provide systematic procedures that convert one design into another. Instead, these transformations show methods that can be used to convert one design into another. These transformations are not standard for every possible transformation. The examples described in this report provide a means of converting the designs created for the Advanced Education Program database. These transformation methods can be used as model to create other more elaborate transformation methods.

ANALYSIS

System Requirements

The intent of this database is to serve as an automated means of accessing data concerning participants in the Advanced Education Program at GE Aerospace. The database will contain all of the information concerning the employee, including his/her personal information, professional status at the

company and current educational status. Table 1 provides a detailed listing of the information that will be included in the database along with the abbreviations that will be used in the discussion of this database design. The relationships that exist among these database fields are depicted in the Entity-Relationship Diagram shown in Figure 1. The data fields listed in Table 1 will be used to create different hard copy or soft copy reports that are based on ad hoc user queries. Currently, all of the data must be manually processed and reports are created on a case by case basis. Data access for the Advanced Education Program Database will be relatively simple, because all data will reside on a stand-alone IBM PC platform.

Database Field	Description
SSN	Employee Social Security Number
ENAME	Employee Name
ADDRESS	Employee Address
PHONE#	Employee Phone Number
TITLE	Job Title
MANAGER	Current Manager
LOCATION	Current Job Location
PAST_POSITION	Titles of Previous Positions
SKILLS	Technical Skills of the Employee
DEGREE	Pursuing Degree
MAJOR	Major Program of Study
SCHOOL	University Attending
SDATE	Degree Start Date
GDATE	Expected Graduation Date

Table 1. Fields of the Advanced Course Database

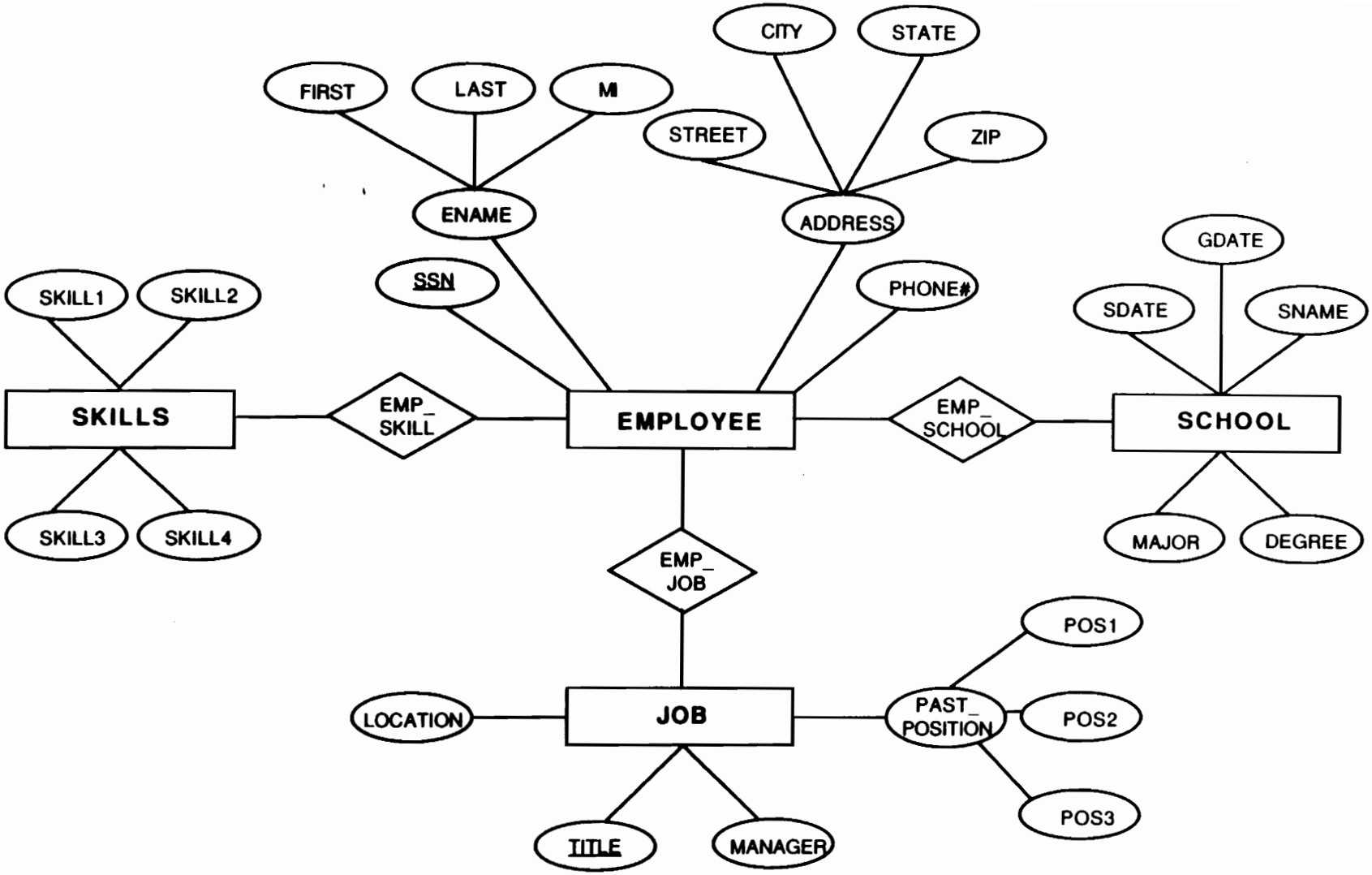


Figure 1. Entity-Relationship Model for Advanced Course Database

The following list provides a summary of some of the report queries that may be requested of the database.

1.) All educational information of an employee (DEGREE, MAJOR, SCHOOL, SDATE, GDATE)

2.) The current job of one or many employees

3.) All working status information of an employee (TITLE, MANAGER, LOCATION)

4.) All previous job positions held by a particular employee

5.) All skills of a particular employees

6.) All employees who posses a particular skill

7.) All employees that are enrolled in a particular major

8.) All employees that attend a particular school

9.) All employees working at a particular job location

10.) Start dates and graduation dates of particular employees

11.) All employees anticipated to graduate on a particular date

This list is not ranked in order of importance. The reason for this is that the queries have different levels of importance, depending on who is using the system. For example, consider a program participant who is trying to find information on a job position that involves a particular skill set such as networking. Also, consider a program administrator who is trying to determine how many students are enrolled at a particular university. Neither of these queries is more important than the other; however, the participant will probably not be interested in the information being queried by the administrator. Therefore, the importance of the queries is totally user-dependent. Although the

system location will be more convenient to the program administrators, the intent of the system is to support both administrators and participants. Thus, all users will have equal priority for system use.

These queries are to be processed on an as needed basis. Some of the queries such as those stated in items 8 and 9 may be processed daily to complete reports. However, other reports such as a request for educational information on students may be accessed monthly in order to determine the specialty fields of particular individuals. Although these queries are processed at different intervals, the processing will not be continual as is the case in many larger scale database systems. An example of such a system is used in an airline company, where the business is totally dependent on stored data. The Advanced Course database system will be used on a day-to-day basis by the program administrators and on an as needed basis by program participants who are seeking information concerning professional opportunities. In either of these cases, the user will need to access the data as efficiently and quickly as possible.

The security of this database is an issue in its design due to the sensitive nature of the data that is stored. Access to this data will be limited to users that possess a required password. Only users with this password are allowed to access the database. The responsibility of ensuring that a user possesses the required access privileges outlined in the data dictionary is held by the database administrator (DBA). The DBA is also responsible for maintaining the integrity of the database. In the case of the Advanced Education Program Database System, the administrative assistant of the program will serve as the DBA. This individual will be responsible for insuring that all stored information is kept current and accurate.

Form-Based Approach

The Form-Based approach to database design is based on the concept of database systems storing information once contained in business forms. A business form is defined as "a structured collection of variables that are appropriately formatted for data entry and display" (Choobineh, 1992, p.110). The Form-Based approach was created based on two major observations:

1.) Because of the familiarity , end users can effectively communicate many requirements through the forms they use.

2.) Usually the most widely used data are gathered or reported in a form. (Choobineh, 1992, p.109). There are two systems associated with the Form-Based approach to database design - the Form Definition System and the Expert Database Design System.

The Form-Definition System portion of the Form-Based Model supports both form and view definitions. The form layout component provides a full-screen editor for entering form field captions and example values (Choobineh, 1992, 110). The Form Definition System also consists of three components - the interface component, the command component, and the inference component. The interface component shows the type of environment the system will have. The sample that is given by Choobineh is a Macintosh-like environment with pull-down menus, a pointing device, and graphics that display the form. This description is also appropriate for this case study. The command component provides functions for input/output, form property definition, and explanatory feedback. The inference component, which is the most complex, supports the invoice made through a collection of rules and heuristics. A unique characteristic

of the Form Definition system is a component that makes inferences from examples "using a small collection of rules and heuristics and a powerful dialogue" (Choobineh, 1992, p. 109). These inferences are generally based on the hierarchical structure of a form as well as the functional dependencies among the form fields.

The Expert Database Design System uses the view definitions to incrementally build an entity relationship diagram. In this system, a collection of rules are "used for grouping the fields on a form into aggregate objects and for ordering a collection of forms for incremental view integration" (Choobineh, 1992, p.109). A summary of the steps involved in using these systems is given next.

In order to design a Form Definition System, the following steps must be followed.

1.) Define Form Model
2.) Define Form Hierarchy Inference
3.) Define Functional Dependencies
4.) Define Informative Feedback

These steps are described in detail below as they are applied to the Advanced Education Database System.

The first step in this system design is to define a form model. A form is defined as a structured collection of variables that are appropriately formatted for data entry and display. The form type defines the structure, constraints, and presentation of the form field. The form type for the Advanced Education database is character-oriented templates. A form instance is a particular collection of values for the form fields. Figures 2 and 3 represent a sample of the

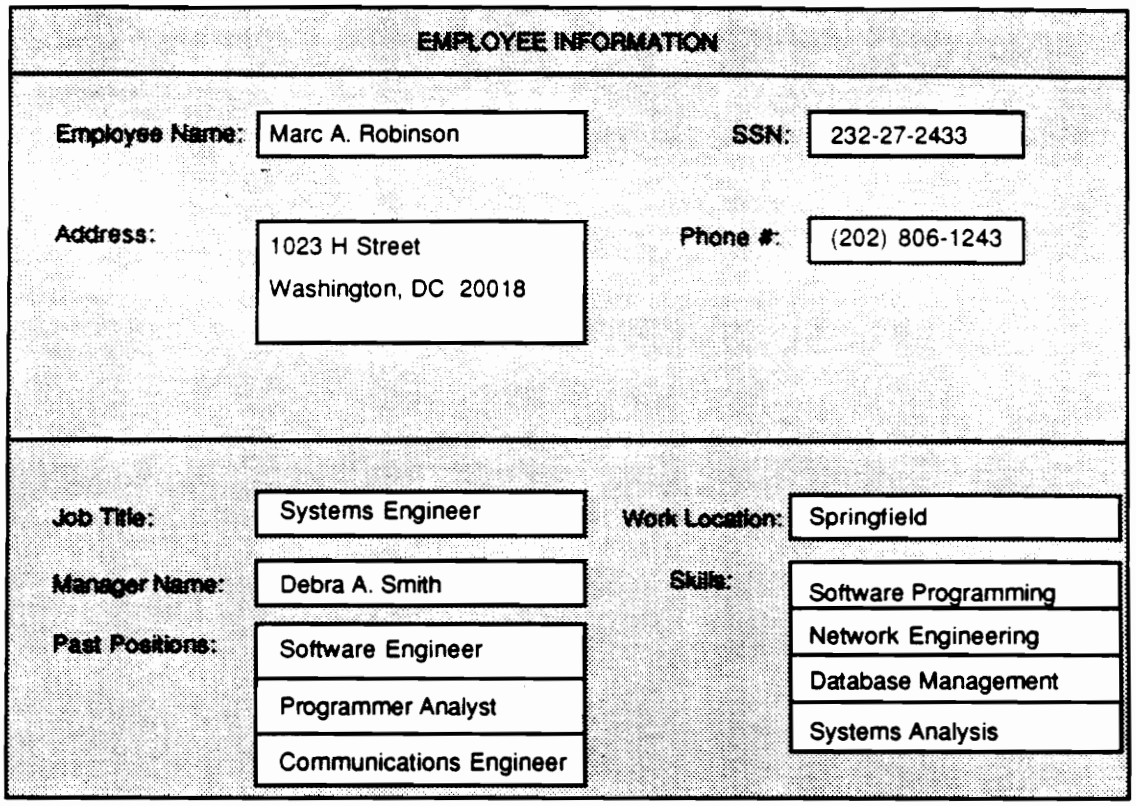


Figure 2. An Instance of the Employee Information Form

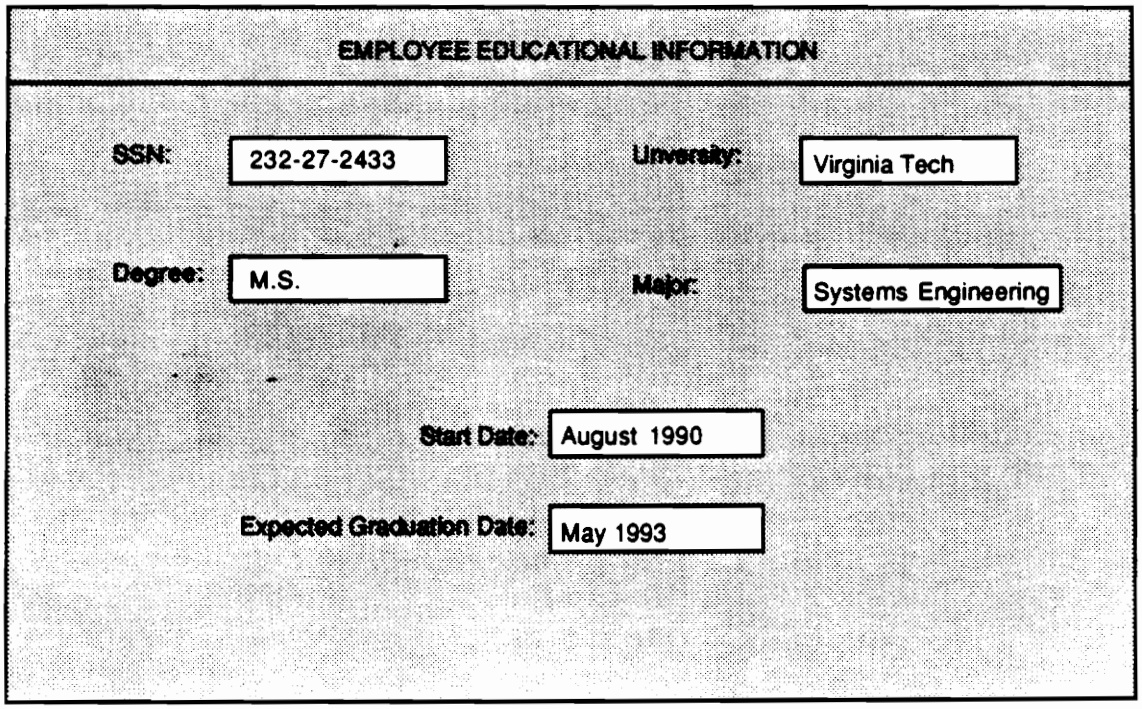


Figure 3. An Instance of the Educational Information Form

form templates for the Advanced Education database. These figures also represent an instance of the database and show the information that is required of each employee participating in the program. Another aspect of these form fields is that of their static properties These properties include their type, presentation, structure, origin, and constraints. These properties are discussed in detail below.

The form type is defined by the type of data stored in the fields. For purposes of this database the forms will consist of Character Strings and User-Defined Types. The SSN and PHONE# fields of the database will be user-defined enumerated types and all other forms will be represented by character strings.

The presentation of a form is the mapping of a form field to a template by organizing them into a hierarchical structure (Choobineh, 1992, p. 110). Figure 4 presents a hierarchical structure of the Employee Information and Educational Information forms. The Educational Information form has two levels and the Employee Information form has three levels.

The origin of a form field indicates the source of its values. Table 2 provides a listing of the possible origin values. For example, all fields of the Employee Information form are user-triggered, because all of the information in this form will initially be entered by the user. In the Educational Information form, the SSN field is form-triggered from the Employee Information form and all other fields are user-triggered.

The constraints of a form include the designation of node keys for the form hierarchy, null value permission, default values, and value ranges, if numeric, and enumerated values if nominal. The node key for both of the forms

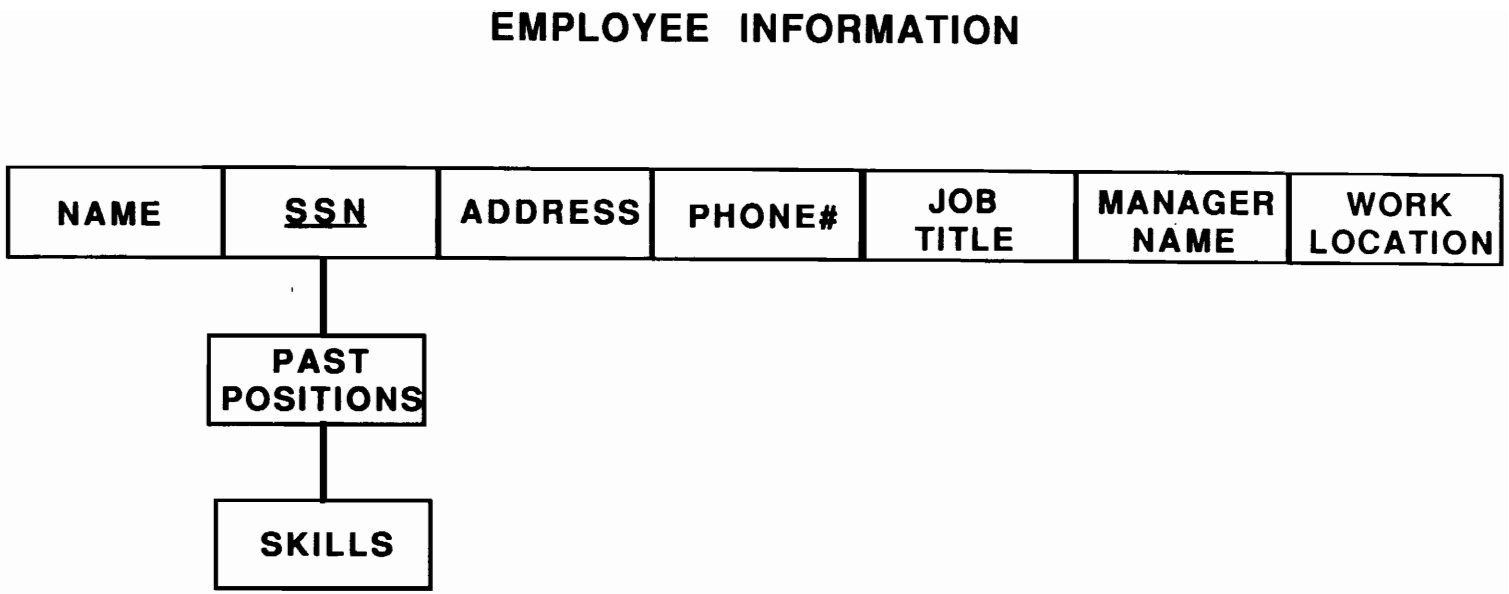


Figure 4. Hierarchical Representation of the Employee Information and Educational Information Form Fields

Origin Type	Symbol	Meaning
User-Triggered	U	User enters value
System-Triggered	S	System enters values without referencing any value on the form
Computation-Triggered	C	Value is computed from one or more form fields
Form-Triggered	F	Value is identically transferred from one form to another
Value-Triggered	V	Value displayed because of a value of a field in current form
Form-Value-Triggered	FV	Value displayed because of a value of a field in another form

Table 2. Origin Values for Screen Fields (Choobineh, 1992, p.110)

mentioned above is the SSN field. The null value permission is only allowed in selected fields - SKILLS and PAST_POSITIONS. These fields have multiple values and at any given time a portion of them can have a null value. Although it seems conceivable for the other fields in the form to have nulls values, a constraint of this system is to make sure that this does not happen. This constraint ensures that a form is complete. All of the fields on the two forms, excluding the two mentioned above, are necessary in order to maintain data on the participants of the system. Once an employee is enrolled in the Advanced Education Program, all fields in the form must be completed and this will create a form instance.

The next step in developing a Form Definition system is to determine the Form Hierarchy Inference. The Inference component uses a collection of rules

and heuristics, which suggest structures and assertions based on a given example set. A database system infers a form hierarchy in four major steps:

1.) Cluster the fields into nodes,

2.) Identify paths and determine the hierarchical structure of each path,

3.) Identify the parent nodes of multi-path structures, and

4.) Validate the conclusion through additional examples and generalize to the hierarchy which covers all instances (Choobineh, 1992, p.112).

The rules that have been defined for the Advanced Education System are Field Clustering, Parent Child, Path Detection and Hierarchy preference. These rules are generally common to most form-based systems. The purpose of the field clustering category is to group form fields into nodes if those fields have the equivalent field cardinality.

As stated earlier, the inference component of the Form Definition System supports the invoice made through a collection of rules and heuristics. These rules are based on certain assumptions that must be made. These assumptions are as follows.

1.) Users enter data in a fully non-normalized format.

2.) Except for the root node, fields of the same node are positioned together.

3.) Nodes on the same path are positioned adjacently from left to right where the nodes are positioned together.

Once these assumptions are made, the heuristics then suggest certain structures and assertions. These rules are designed to infer the simplest plausible hierarchy.

The hierarchy inference for the Advanced Education database is shown in Figure 4. The form hierarchy was inferred by the system using the four major steps listed above. A summary of the hierarchy inference rules that are used for this system is listed in Table 3. The field clustering rules group fields into nodes based on their cardinality, positioning, and mapping. Figure 4 shows that all fields that have a cardinality of one (1) are clustered together. This same rule applies for SSN, UNIVERSITY, DEGREE, MAJOR, START DATE, GRADUATION DATE, because their cardinalities are all identical.

After identifying the nodes, paths of nodes, and the hierarchical position, those nodes with a path are determined. The parent-child heuristics use

Category	Purpose	Conditions
Field Clustering	Group Form Fields into nodes	Equivalent field cardinality positioned adjacently 1:1 mapping among fields
Parent Child	Determine hierarchical position within a path	Smaller cardinality; Missing values; Node containment
Path Detection	Detect a Path Change	Larger cardinality; No containment

Table 3. Hierarchy Inference Rules

information about node cardinality and missing values to determine the parent-child relationship. In Figure 4, the node containing PAST POSITIONS is selected as the parent of the node containing SKILLS, because the cardinality of PAST POSITIONS is smaller than that of SKILLS. Also, there is a missing value for PAST POSITIONS where there is a value for SKILLS.

Once the parent-child relationships are determined, the path detection rules are used to determine the variety of paths. These rules use information about missing values and node containment to decide whether the system has multi-paths. These rules do not apply in the Advanced Education Database System because the Employee Information and Educational Information forms have single paths.

After a hierarchy is inferred, the node and local keys are determined by identifying, ranking, and testing potential keys. In the cases of the Employee Information and Educational Information forms, SSN is identified as the key because it has unique values across all examples on the form. The other potential keys in the root node of the Employee Information form, such as EMPLOYEE NAME, ADDRESS, JOB TITLE were eliminated because their values can be duplicated.

The functional dependencies for the Advanced Education Database System are shown in Figure 4 through the node keys. For example, there exists a functional dependency (FD) SSN --> EMPLOYEE NAME. This means that for every occurrence of a Social Security Number, there is a name of an employee that is associated with that number. Likewise, there is a functional dependency between SSN and every other field that is clustered in the same node.

"Informative feedback is provided in two categories: the background knowledge for making inferences and the inferences made on a given form" (Choobineh, p.114). The background provides definitions for terms such as field, field cardinality, node, root node and dependent nodes. Figure 5 shows the lowest level of explanation generated by the Form Definition System for the field clustering heuristics applied to both the Employee Information and Educational

Form Name:	Employee Information
Instance Number 1:	
Field Clustering (Cardinality = 1):	NAME SSN ADDRESS PHONE# JOB TITLE MANAGER NAME WORK LOCATION
Field Clustering (Cardinality = 3):	PAST_POSITIONS
Field Clustering (Cardinality = 4):	SKILLS

Form Name:	Educational Information
Instance Number 1:	
Field Clustering (Cardinality = 1):	SSN UNIVERSITY_NAME DEGREE MAJOR START_DATE GRADUATION_DATE

Figure 5. Hierarchy Explanation for Employee Information and Educational Information Instances.

Information instances shown in Figures 2 and 3. The hierarchy explanation shows that fields with a cardinality of 1 are grouped into the root node. Fields with the same cardinality, are positioned adjacently, and have a 1:1 mapping for every data item are grouped into dependent nodes. In this case, the root node contains only fields that have the cardinality of 1. SKILLS and PAST_POSITIONS have cardinalities of 3 and 4 respectively as shown in Figure 5, and are therefore shown here as dependent nodes.

As stated earlier, the Expert Database Design System (EDDS) produces an Entity Relationship Diagram (ERD) based on the analysis of the forms contained in the Form Abstraction Base. The EDDS is used in Form-Based Database Design to build a schema diagram by analyzing one form at a time. A

collection of rules is used to determine the order in which the forms are analyzed and to identify the entities, attributes, and relationships that represent the forms.

These rules are generally determined by the designer in order to derive a consistent schema diagram that correctly represents the forms.

According the Joobin Choobineh, "the Architecture of the EDDS is made up of a knowledge base and three databases. The Data Design Knowledge Base (DDKB) contains data design rules and rules for mapping from the forms to ERDs. The Form Abstraction base contains the form definitions that have been discussed previously. The Design Database (DDB) contains the schema diagram and the Design Status Base (DSB) records the current status and past design decisions. These three databases assist in ensuring continuity of work over time and between different sessions of the same design" (p. 115).

The EDDS supports six phases in database design - form selection, entity identification, attribute attachment, relationship attachment, cardinality identification, and consistency. These phases are described in detail below in order to show how the Entity Relationship Diagram for the Advanced Education Database System was derived.

The Form Selection Phase determines the next form that should be analyzed. The first case is when no forms have been analyzed or when no forms are related to any other forms that have been previously analyzed. In this case the form with no destination fields is chosen. For the Advanced Education System, the Employee Information form is analyzed first. The second case considers the remaining forms that have fields originating from the previously analyzed forms. The next case to analyze is the Educational Information form. This form is chosen because of the destination field, SSN. A destination field is

defined as any form field whose origin is either form or form-value triggered. Therefore, since the SSN field of the Educational Information form is derived based on the Employee Information form, the form choice is verified.

The Entity Identification Phase determines the form fields that represent entities. This phase uses rules based on local keys, dependencies, origin, node, and, groupings of form fields to suggest the possible representation of an entity. The entities for the Advanced Education System were chosen based on these rules. Since the identification of entities is left up to the designer, the following entities were chosen for the system: EMPLOYEE, JOB, MANAGER, SKILLS, POSITIONS, and UNIVERSITY.

In the Attribute Attachment Phase, the attributes of entities and relationships are identified. The rules are based on the functional dependencies that have been defined. Figure 6 shows the attributes that are attached to the entities identified in the previous phase.

ENTITY	**ATTRIBUTES**
EMPLOYEE	NAME, ADDRESS, PHONE#, SSN
JOB	TITLE, LOCATION
MANAGER	NAME
PAST_POSITIONS	POSITION TITLE
SKILLS	SKILL NAME
UNIVERSITY	UNIVERSITY NAME, DEGREE, MAJOR, START_DATE, GRADUATION_DATE

Figure 6. Entity Attributes for the Advanced Education System

The Relationship and Cardinality Identification phases connect the previously identified entities with relationships and determine the minimum/maximum cardinalities of an entity. These relationships and cardinalities are determined based on the origin of form fields, the hierarchical structure of forms, and the functional dependencies. A relationship can be established between two entities where a field in one functionally determines a field in the other. The relationships and cardinalities are shown in Figure 7. The PERFORMS relationship between EMPLOYEE and JOB is a prime example because an Employee's SSN determines the TITLE of the Employee as well as the JOB_LOCATION. The minimum/maximum cardinalities are also shown in Figure 7 by separating the two by a colon and enclosing them in parentheses. In order to interpret one of these cardinalities, consider the relationship between JOB and EMPLOYEE. The (1:m) for JOB means that a particular job position can be held by more than one EMPLOYEE. The (1:1) for EMPLOYEE means that an EMPLOYEE can hold exactly one JOB.

Object-Oriented Approach

"An Object-Oriented Database System is one that supports the classical features of database systems (Delobel, 1991, p.264). These features support large amounts of persistent, reliable, and shared data. One approach to designing an Object-Oriented Database uses the Structured Object Model (SOM). The SOM is a structured methodology that represents data semantics using objects, attributes, and two types of relationships: aspects and specializations. The SOM can be used in both logical and physical database design. Also, the SOM uses attributes to identify and describe objects by

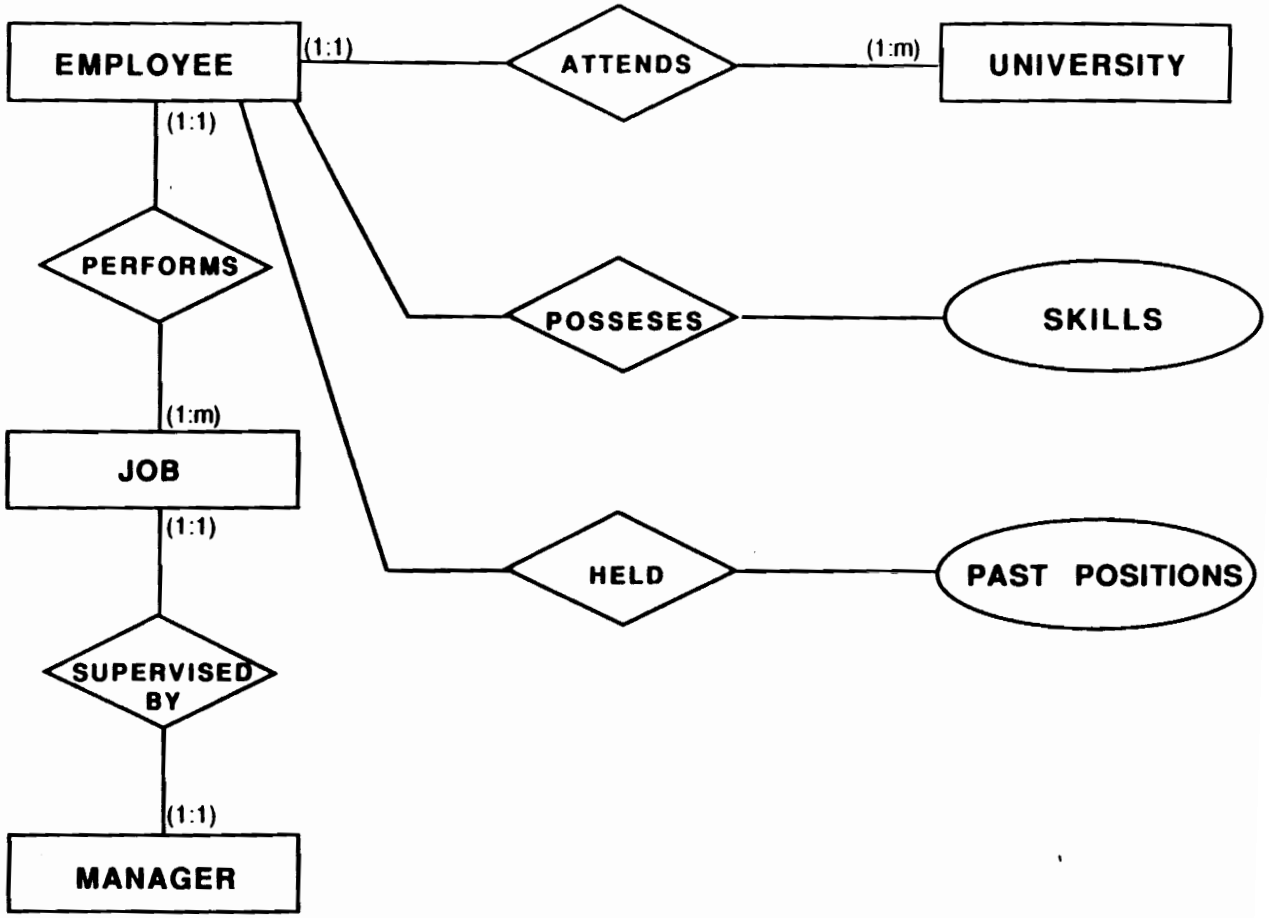


Figure 7. The Entity Relationship Diagram Based on the Expert Database Design System for the Form-Based Design Methodology.

providing properties such as name, shape, and color (Higa, 1992, p.101). Object-Oriented methods are much like the conventional procedure class, except that they are more powerful and more versatile.

An Object-Oriented database design centers around the definition of an object. From a data modeling perspective, an object is defined as a person, place, thing, organization, concept, or event about which users wish to collect and store information. These objects are grouped into classes according to those that have common behaviors. These classes are defined by tangible object types with multiple instances. Representations of classes and objects are formed by constants and variables that either have common class values or individual instance values.

The SOM database development process consist of the following four steps:

1.) Define the most general object
2.) Define relationships between classes
3.) Check completeness of relationships
4.) Simplify the SOM diagram

These steps are outlined below for the Advanced Education System, and are based upon a theory from "An Object-Oriented Methodology for Knowledge Base/Database Coupling" by Kunihiko Higa.

The first step in the SOM database development process is to define the most general object and identify all relevant classes and attributes. The general object is Advanced Education Program. The following list shows the relevant classes and attributes.

1.) EMPLOYEE describes all members of the Advanced Education Program. Attributes include SSN, NAME, ADDRESS, and PHONE#.

2.) JOB describes the position that is currently held by the employee. Attributes include TITLE and WORK LOCATION.

3.) MANAGER describes the direct functional manager of the employee. The attribute of MANAGER is name.

4.) PAST POSITIONS describes the jobs that have been previously held by the employee. Specifications include the actual titles of these positions - POS1, POS2, and POS3. Attributes include the Title of the position.

5.) SKILLS describes the technical talents that are possessed by the employee. Specifications include the types of skills - SKILL1, SKILL2, SKILL3, and SKILL4. Attributes include the name of the skill.

6.) UNIVERSITY describes the school that the employee is currently attending to earn a graduate degree. Attributes include NAME, MAJOR, DEGREE, START DATE, and GRADUATION DATE.

The second step in the design process defines the relationships between classes. "Experience with the SOM indicates that steps 1 and 2 should be done iteratively using SOM and another conceptual database" (Higa, 1992, p.103) such as the Entity Relationship Diagram shown in Figure 1. The result of steps 1 and 2 is the Logical SOM shown in Figure 8. The SOM shows the entity class EMPLOYEE. The relationships illustrated on the Logical SOM are discussed based on this class. EMPLOYEE has the aspects Job, Skills, Past Positions, and University. The entities Skills and Past Positions both possess specializations defining the individual skills and positions that are associated with those objects. The legend shown in Figure 8 provides a summary of the types of relationships that exist between entities. Figure 8 provides a very basic Logical SOM as compared to other systems; however, the diagram does represent the simple system that is being designed in this case.

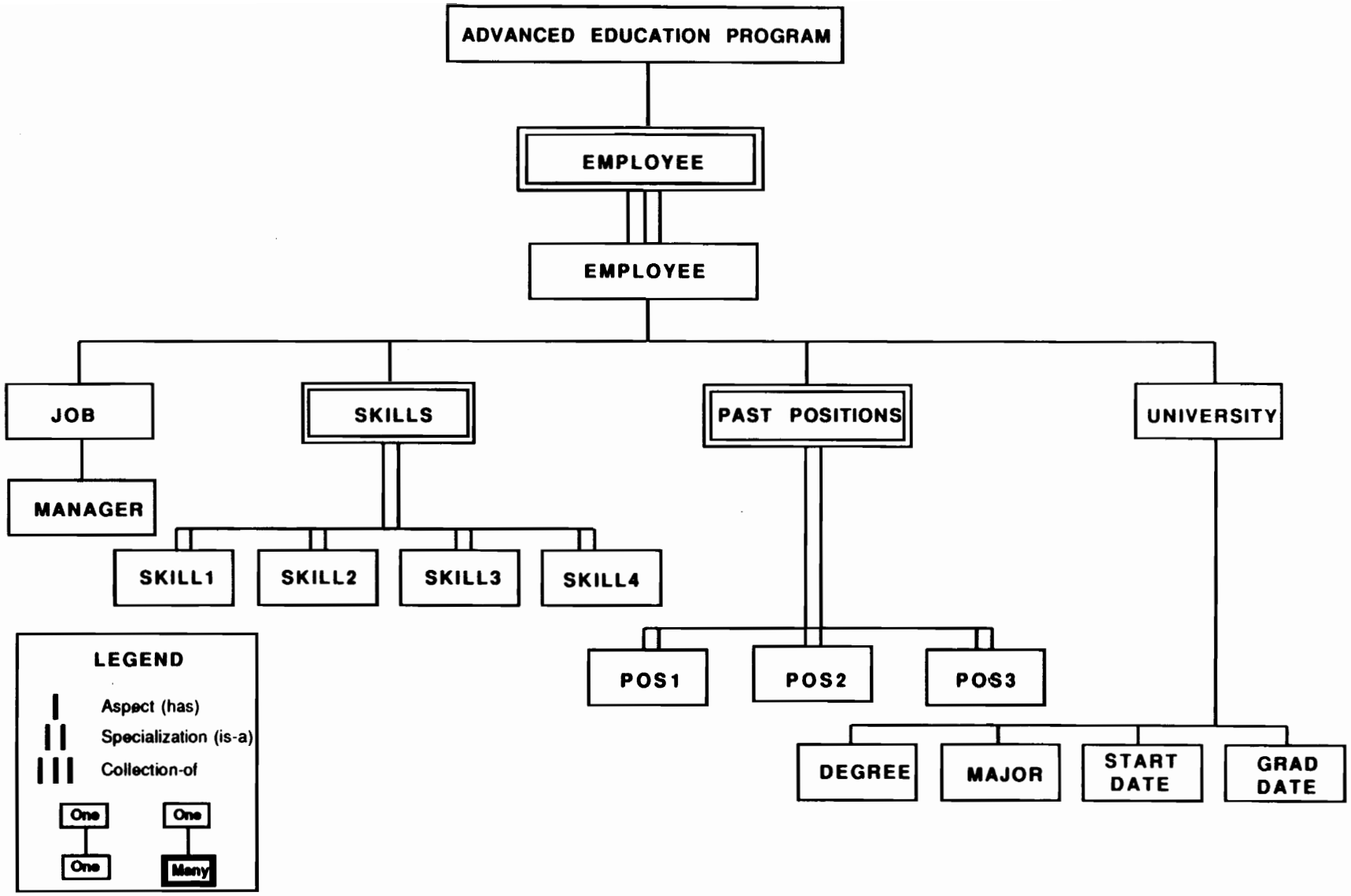


Figure 8. Logical SOM Model for the Advanced Education Database System

Step 3 of the Object-Oriented Database Development Process is to check the completeness of the involved relationships. This step ensures that there are no incomplete binary or ternary relationships. Figure 8 provides a complete depiction of the relationships included in this system.

The fourth step of the process is to simplify the SOM diagram and copy each object structure to a record. A set of rules, developed by Higa and others, has been developed to simplify the Logical SOM into a set of relations representing the physical database design. The rules that are applicable to this system design are One-to-One and Many-to-Many. The result of applying these rules to the Logical SOM is shown in Figure 9. This figure presents a depiction of the objects that exist in the system and defines the attributes that are associated with these objects. In many cases the Physical SOM is considered a skeletal depiction of the SOM, because it is more focused on the objects and their attributes rather than the relationships that exist between the objects.

Relational Approach

A relational database is defined as a collection of relations that contain all of the information that is to be stored in the database (Jackson, 1992, p. 5). Each relation in such a database is stored as an individual file. However, in some larger systems, relations are stored as indexed files, where the index is an attribute or set of attributes. The most reasonable comparison for a relation is that of a table that has several rows and columns. According to Jackson, the goals of a relation include the following:

1.) Have the capability of storing all pertinent data in the database.

2.) Eliminate all redundant data.

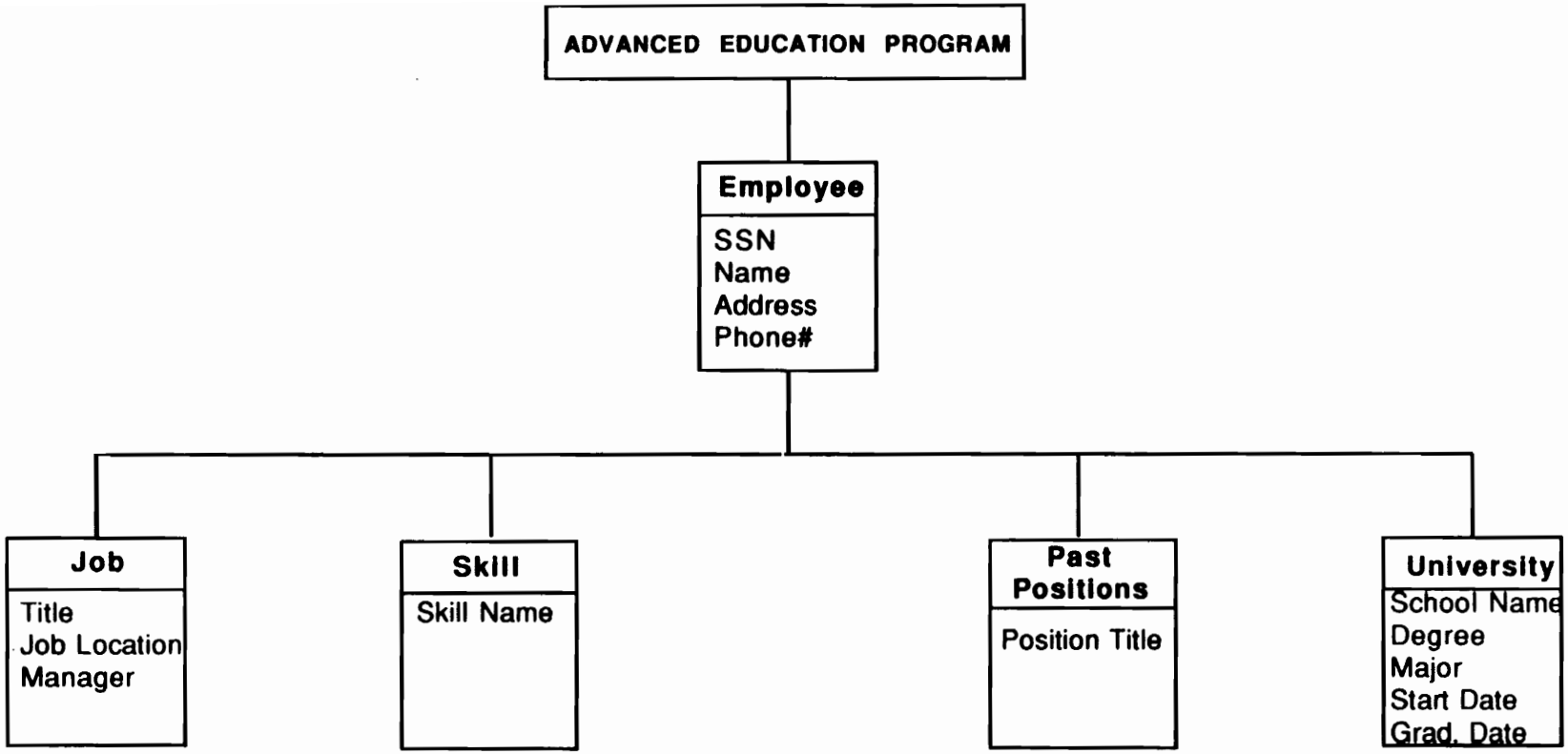


Figure 9. Physical SOM Model for the Advanced Education Database System

3.) Keep the number of relations in the database to a minimum.

4.) Have the relations normalized, so as to minimize update and deletion problems (p. 9).

The realization of these goals leads to the design of an accurate Relational Database.

The first step in the design process using the relational approach is to determine all of the attributes that a user wishes to have in the database. For purposes of the Advanced Education System, the attributes have not changed. The user for this system requires the same attributes that are listed in Table 1. These attributes are listed again in Table 4 along with a short two character form that will be used in the functional dependency (FD) diagrams. The two character versions are used in the relational approach to prevent the FD diagrams from becoming too cumbersome. As in both of the afore mentioned database designs the fields listed in Table 4 represent all of the fields that will be needed in the system, according to the potential users.

Once the attributes of the system have been determined, the next phase in the Relational Database design process is to determine the universal relation of the system being designed. A relation is most simply defined as a table with data values stored in table rows. The clearest way to understand the meaning of a relation is through visual representation. Figure 10 represents relations that can exist in the Advanced Education system based on the requirements stated earlier. This figure shows the relation POSITION made up of four columns, named SSN, Employee Name, Job Title, and Job Location. This relation stores

Database Field	Attribute Name
SSN	Sn
ENAME	En
ADDRESS	Ad
PHONE#	Ph
TITLE	Ti
MANAGER	Mg
LOCATION	Lc
PAST_POSITION	Pp
SKILLS	Sk
DEGREE	Dg
MAJOR	Mj
SCHOOL	Sl
SDATE	Sd
GDATE	Gd

Table 4. Database Field Names and Associated Attribute Names

the job information of employees in the Advanced Education Program. Each row in the relation shows the job title and location of one person in the program. According to Glenn Jackson, the universal relation is formed by grouping all of the attributes that will be used in the database system. Therefore, the universal relation for the Advanced Education System is defined as

R(Sn,En,Ad,Ph,Ti,Mg,Lc,Pp,Sk,Dg,Mj,Sl,Sd,Gd)

The third step of the design process is the determination of all of the FDs relating the attributes in the universal relation. Figure 11 gives the functional

POSITION

SSN	Employee Name	Title	Work Location
227-32-1278	Duane Brown	Software Specialist	Springfield
322-54-3987	Andrea Jones	Systems Engineer	Reston
122-34-5628	Franklin Smith	Project Engineer	Reston
044-89-3476	Michelle Smith	Programmer Analyst	Hanover
345-12-9267	Lesia Williams	Systems Engineer	Springfield

EDUCATION

SSN	University	Major
227-32-1278	Virginia Tech	Computer Science
322-54-3987	George Mason	Electrical Engineering
122-34-5628	Virginia Tech	Systems Engineering
044-89-3476	Virginia Tech	Systems Engineering
345-12-9267	George Washington	Computer Science

Figure 10. Relations POSITION and EDUCATION

dependency diagram for the universal relation shown above. This figure shows that there are two composite keys - <Sn,Ti> and <Sn,Sl> - that represent an employee's job and educational information, respectively. All other attributes relating to the personal data of an employee are dependent upon the key, Sn, because the Social Security Number of an individual makes each instance unique. For this reason, Sn is also included as a part of each of the composite keys.

The next step in this process is to remove all of the redundant FDs from the set of FDs defined for the universal relation. For purposes of simplicity in this

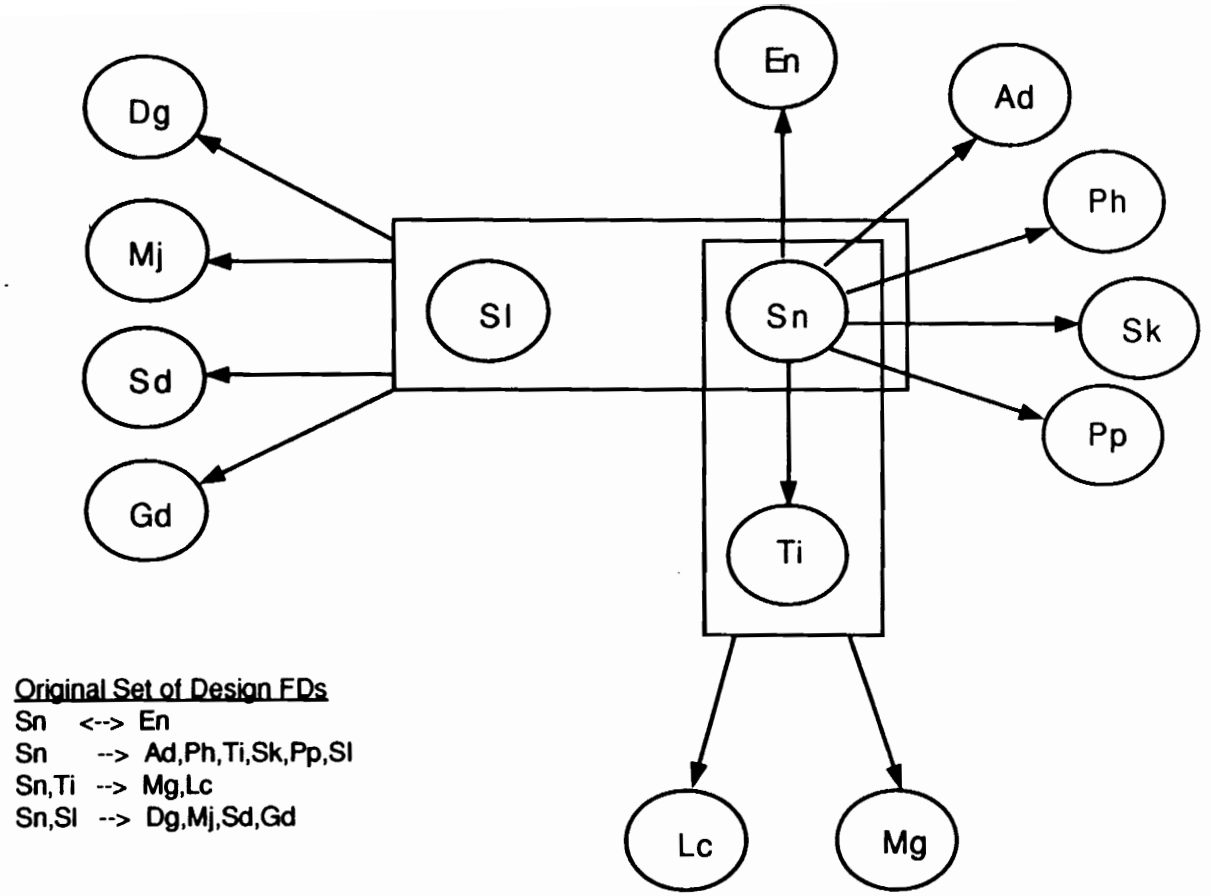


Figure 11. The Original FD Diagram for the Advanced Education Database

design all of the redundant FDs have already been eliminated and were never presented in the original FD; therefore, Figure 11 reflects a valid FD diagram.

The fifth step in the design process involves the reduction of the universal relation into a set of Boyce Codd Normal Form (BCNF) relations. The transition or reduction to BCNF is based on a comparison between the candidate keys and the determinants. If these two lists are identical, then the relation is in BCNF. The universal relation R, defined above, is not in BCNF because when the candidate keys and determinants for R are compared, they are not identical (as shown below).

Candidate Keys in R	**Determinants in R**
<Sn,Ti>	<Sn>
<Sn,Sl>	<Sn,Ti>
	<Sn,Sl>

Therefore, R must be decomposed in order to reach the desired result. This relation can be decomposed by projecting out all FDs with <Sn> as the determinant. This set includes Sn --> En and Sn --> Ad,Ph,Ti,Sk,Pp,Sl. This projection creates the relations R1 and R2 as shown in Figure 12. Both of these relations can now be shown to be in BCNF; therefore, the final design relations are R1 and R2.

The set of BCNF relations that were generated from relation R (shown on page 26) is

R1(<u>Sn,Ti,</u>Lc,Mg,Sl,Dg,Mj,Sd,Gd)
R2(<u>Sn</u>,En,Ad,Ph,Sk,Pp,Ti)

where the primary key of each relation is underlined. (Note: The relation R1 has a candidate key that is not represented by underlining. The attribute, Sl, is also a

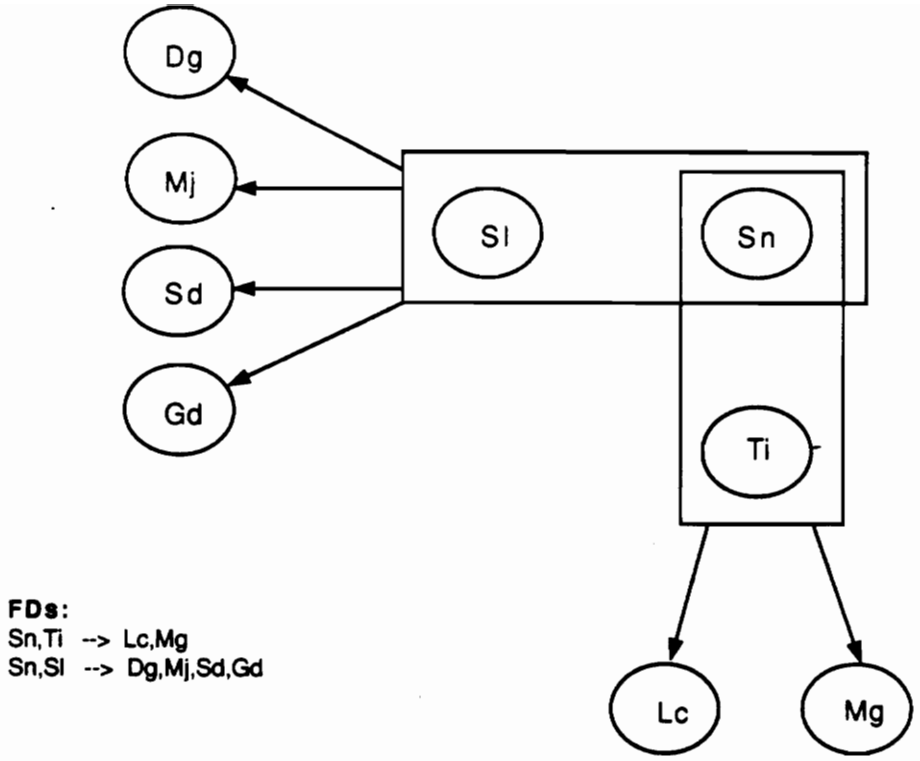


Figure 12(a). R1(Sn,Ti,Lc,Mg,Sl,Dg,Mj,Sd,Gd)

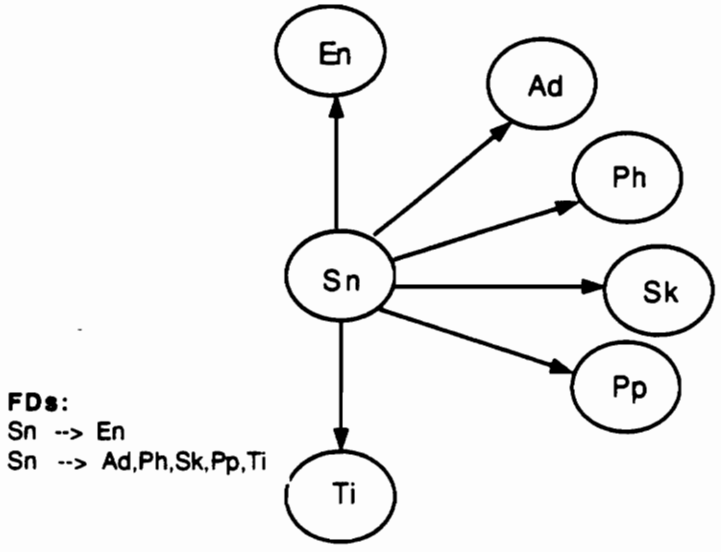


Figure 12(b). R1(Sn,En,Ad,Ph,Sk,Pp,Ti)

candidate key as shown in Figure 12a.) The final step in the design of a Relational Database is to evaluate the BCNF relations. The FDs should first be checked to ensure that (1) no FD appears more than one time and (2) the set of resulting FDs is the same as the original set of relations. This statement is proven true for the Advanced Education Database by listing the FDs of the new relations determined during decomposition. These FDs are

Relation	**Figure**	**FDs**
R1	12 (a)	Sn,Ti --> Lc,Mg Sn,Sl --> Dg,Mj,Sd,Gd
R2	12 (b)	Sn --> En Sn --> Ad,Ph,Sk,Pp,Ti

This information is identical to the FDs shown in Figure 11. Once this equivalence is determined, the relations should then be analyzed to make sure that no attributes of one relation appear as a subset of the attributes in another. This is not the case for the Advanced Education System. The valid relations for the Advanced Education Program Design are

R1 (SSN,TITLE,LOCATION,MANAGER,UNIVERSITY, DEGREE,MAJOR,START DATE,GRADUATION DATE)

R2 (SSN,EMPLOYEE NAME,ADDRESS,PHONE#,SKILLS,PAST POSITIONS,TITLE)

Summary

After completing the design of the Advanced Education Program Database System, the differences between the Form-Based, Object-Oriented, and Relational approaches became very apparent. By examining the resulting database design diagrams shown in Figures 7, 8, and 11, the variations between

these diagrams and the original Entity-Relationship (ER) diagram shown in Figure 1, were observed. Although these variations exist, the core design still produces a system that will be responsible for storing and retrieving the appropriate data for the Advanced Education Program. The three design processes discussed in this report vary in many ways, but the result is similar in each case.

The ER diagram was created based on an initial set of defined system requirements that were consistent for any design methodology. Consequently, this ER diagram can be used to unify different views of data: the Form Model, the Structured Object Model, and the Relational Model. This simply means that although the designs of a database system are created using different methodologies, the results in each can be converted to form the design of another methodology. These conversions are known as transformation methods and are used to show how one could go from one design to another. For example, if a transformation existed, one could go from a Relational design to an Object-Oriented design. The remaining text provides a summary of some of the transformations that can occur among the different designs created for the Advanced Education Program Database System. In order to provide a synopsis of the change from one method to another, the transformations are created using only a portion of the entire database design.

DATABASE DESIGN TRANSFORMATIONS

Transforming An Object-Oriented Design To A Relational Design

"It is generally agreed that object-oriented models provide a more accurate representation of the complexities of an application problem than do many other models" (Hansen, 1992, p. 192). This is definitely the case in the design of the system for the Advanced Education Program. After completing the system design using the different techniques, the resulting representation of the Structured Object Model (SOM) was the least complex of the three designs. The SOM also provides an accurate representation of the objects and their attributes. As discussed earlier, a SOM consists of objects, relationships, and attributes. This section will show the methods for transforming each of these constructs into relations.

The transformation of an object is best shown by reviewing a section of the Logical and Physical SOMs. Consider the object EMPLOYEE in Figure 9 with its four attributes, SSN, NAME, ADDRESS, and PHONE#. EMPLOYEE is considered to be an abstract object which means that it consists of instances that cannot be printed. However, SSN, NAME, ADDRESS, and PHONE# are lexical attributes, because their instances consist of values that can be printed. Since, attributes in the relational model must be lexical, all of these attributes can exist in a relation. Therefore, the EMPLOYEE object of Figure 9 can be transformed into a relation with attributes as follows:

EMPLOYEE (SSN,NAME,ADDRESS,PHONE#)

Assuming that SSN uniquely identifies an employee, the conclusion is that SSN is the key for this relation. Therefore,

EMPLOYEE (SSN,NAME,ADDRESS,PHONE#)

is the transformation of the EMPLOYEE object shown in Figure 9. Thus, this analysis indicates that a direct conversion exists between abstract objects and relations.

Another consideration is the transformation of specialization object sets. An example of this case is the SKILL1 object shown in Figure 8. This object is shown to be a specialization of the object SKILL; therefore, SKILL1 must inherit all of the attributes of SKILL. The attributes of SKILL are shown in Figure 9 as SKILL NAME. Thus, the resulting relation is

SKILL1 (SKILL NAME),

where SKILL NAME is the key for this particular relation.

The next major consideration in this transformation activity is the transformation of relationships. Relationships are transformed based on their cardinality. For example, consider the one-to-one relation of EMPLOYEE to JOB shown in Figure 8. This means that an employee can have at most one job. The resulting relations are shown as

EMPLOYEE (SSN,NAME,ADDRESS,PHONE#)
JOB (TITLE,LOCATION,MANAGER NAME)

To show the connections between the two relations, it is necessary to add the key of one relation to the other relation. This additional key is known as a "foreign key." In this case, SSN must be added to the JOB relation and TITLE must be added to the EMPLOYEE relation, resulting in

EMPLOYEE (SSN,TITLE,NAME,ADDRESS,PHONE#)
Foreign Key: TITLE references TITLE in JOB

JOB (TITLE,SSN,LOCATION,MANAGER NAME)
Foreign Key: SSN references SSN in EMPLOYEE

Because this solution duplicates data and it is only necessary to have one combination of SSN and TITLE, one of the combinations should be eliminated. The best decision is to eliminate TITLE from the EMPLOYEE relation because if TITLE is null in this relation, a SSN will still exist for each employee. This yields

EMPLOYEE (SSN,NAME,ADDRESS,PHONE#)

JOB (TITLE,SSN,LOCATION,MANAGER NAME)
Foreign Key: SSN references SSN in EMPLOYEE

In summary, "relationships of this sort are transformed by showing one of the object sets as an attribute of the other's relation" (Hansen, 1992, p. 196). The relation is chosen based on the needs of the particular application. In this case, the SSN and the Job Title are needed to determine the other attributes of that relation for an instance of EMPLOYEE.

The process of transforming an object-oriented design to a relational design involves creating relations for each object set in the model. The attributes of the object set are attributes of the relation (Hansen, 1992, p. 204). If the relations determined by the transformations are compared to those that were created in the Relational Design approach, very few differences are found. The only way to safeguard against the loss of data is through the careful execution of the transformation process. When time and effort are expended during this process, the probability of retrieving all of the original data is increased. The one thing to consider in any design transformation is that the results may not be

exact; however, with minimal modifications the database design will produce the desired system result. These modifications include the redefinition of relations to produce the desired form. The transformed objects produce relations normalized to Fourth Normal Form. Therefore, the relations must first be represented in Boyce-Codd Normal Form to produce the exact results that occur when designing a database system using the Relational Approach.

Transforming A Form-Based Design Into An Object-Oriented Design

The Form-Based design can be transformed into an Object-Oriented design by first examining the data structures that exist in each model. The Form-Based Model is made up of entities, attributes, and relationships as shown in Figure 7. The Structured Object Model (SOM) is made up primarily of objects and the relationships that exist between them as shown in Figure 8. In order to convert a Form-Based design to an Object-Oriented design, it is necessary to determine the similarities that exist between the two. In other words, one must determine a link between the structures that exist in the two designs.

First, consider the entity EMPLOYEE that is shown in Figure 7. The entities shown in this diagram represent the form fields identified as essential components of the Advanced Education System forms. EMPLOYEE consists of four attributes - SSN, NAME, ADDRESS, and PHONE# - as shown in Figure 6. This entity can be transformed into an object based on its representation in the Form-Based Model. As defined earlier, an object is a person, place, thing, or concept about which a user wishes to collect data. Based on this definition, an entity of the Form Model is equivalent to an object structure of the SOM. Thus, the entity EMPLOYEE shown in Figure 7 is representative of the object

EMPLOYEE that is shown in both Figures 8 and 9. This object also adopts the attributes of the entity that existed in the Form Model. Thus, the object EMPLOYEE shown in the first level of Figure 9, along with all of the appropriate attributes, is representative of the entity that has been described above.

Next, the relationships of the Form Model must be transformed into those that exist in SOM. The transformations of these relationships is dependent upon the cardinalities of the entities. The simplest way to show these transformations is through a graphical representation. Therefore, the following transformations make numerous references to relationship diagrams that have been designed previously. For example, a relationship ATTENDS exists between the entities EMPLOYEE and UNIVERSITY, as shown in Figure 13. The cardinality of EMPLOYEE in this case is one-to-one as denoted by the (1:1) representation. This means that an employee can attend at most one university at any given time. This relationship can be transformed into a SOM relationship by defining the type of relationship that exists. This relationship is represented in the SOM as an object, UNIVERSITY, which is shown as an aspect of the object EMPLOYEE. The aspect is the type of relationship that exists between the two objects. In this case, the ATTENDS relationship is denoted in Figure 14 by the aspect relationship and the one-to-one cardinality. This means that an EMPLOYEE has (or can attend) only one UNIVERSITY. For purposes of the Advanced Education Program Database, the one-to-one relationship is represented in the SOM diagram by a path consisting of a single line that extends from the object EMPLOYEE to the object UNIVERSITY, where the object is denoted by a single box.

Figure 13. The Form-Model ATTENDS Relationship Between the Entities EMPLOYEE and UNIVERSITY

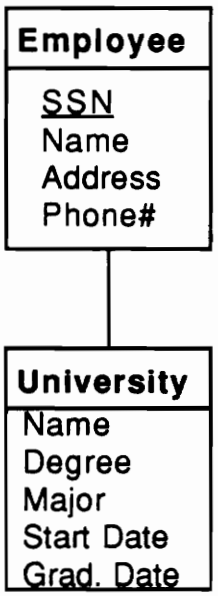


Figure 14. The Object-Oriented Transformation of the Form-Model ATTENDS Relationship

Similarly, other relationships with different cardinalities are transformed using the same method. For example, consider the POSSESSES relationship that exist between the entities EMPLOYEE and SKILLS. The object SKILLS is considered in this case to be an aspect of the of the object EMPLOYEE. This means that any employee is allowed to have a set of SKILLS. This relationship is considered to be a one-to-many relationship and is shown in the SOM by the double box that denotes the SKILLS object.

In summary, the process of transforming the Form-Based design into the Structured Object design is the simplest technique for the Advanced Education Program Database application. The reason for this is that there is a direct correspondence between the entities and relationships that exist in the Form-Based Model and the objects and relationships that exist in the SOM. Although this correlation exists, it is important to remember that once the transformation is made, some simple modifications must be made in order to produce the desired result. These modifications include providing a means of identifying one-to-many relationships as specializations or collections. Steps must then be taken to further divide these identified groups into more specific objects. For example, consider the transformation of the relationship POSSESSES that is shown above. Although the method of entity-to-object transformation will create the SKILLS object, some steps must be taken in order to ensure that the possibility of object class specializations and collections is considered when a one-to-many relationship exists. As shown in Figure 8, the object SKILLS is further represented as a specialization of four more objects - SKILL1, SKILL2, SKILL3, and SKILL4, meaning that these four objects are a subset of the object SKILLS.

Transforming A Relational Design Into A Form-Based Design

The Relational Model takes the approach of organizing and representing data in the form of tables, or relations. In order to transform this design into the Form-Based design, it is essential to first define the contents of these relations and the functional dependencies that exist between the attributes of these relations.

The BCNF Relational Model that was created earlier consists of the following relations:

R1 (SSN,TITLE,LOCATION,MANAGER,UNIVERSITY,
DEGREE,MAJOR,START DATE,GRADUATION DATE)

R2 (SSN,EMPLOYEE NAME,ADDRESS,PHONE#,SKILLS,PAST
POSITIONS,TITLE)

Based on these two relations, the following functional dependencies were determined as necessary for the Advanced Education Program Database System:

SSN --> EMPLOYEE NAME
SSN --> ADDRESS,PHONE#,SKILLS,PAST POSITIONS,TITLE
SSN,TITLE --> JOB LOCATION,MANAGER
SSN,UNIVERSITY --> DEGREE,MAJOR,START DATE,
GRADUATION DATE

The relations and functional dependencies listed above are represented in Figures 11 and 12. In order to transform these relations into form entities, the relation must be considered on the basis of its dependencies. However, before proceeding with the transformation, the dependencies must first be restructured to more clearly define the existing relationships. For example, the above listed dependencies can be restructured to create

SSN --> EMPLOYEE NAME,ADDRESS,PHONE#
SSN --> TITLE
SSN --> SKILLS
SSN --> PAST POSITIONS
SSN,TITLE --> MANAGER,JOB LOCATION
SSN,UNIVERSITY --> DEGREE,MAJOR,START DATE,
GRADUATION DATE

These dependencies are merely reorganized here in an effort to separate the dependencies according to their significance to each other. For example, there is no need to have TITLE and SKILLS in the same dependency with the personal information of the employee such as NAME, ADDRESS, and PHONE#.

Based on these rearranged dependencies, the relation R1 can be partitioned to create a relation called EMPLOYEE, which contains all of the appropriate attributes for that relation. This relation can be represented as

EMPLOYEE (SSN,NAME,ADDRESS,PHONE#)

and can now be transformed into an entity of the Form Model. The attributes of this relation are now considered to be attributes of this new form entity EMPLOYEE. The form entities are constructed based on the relations that exist in the Relational Model. Another example would be the partitioning of relation R2 to create a relation called JOB that consists of the attributes TITLE, MANAGER, and LOCATION. This relation is represented as

JOB (SSN,TITLE,MANAGER,LOCATION)

The SSN attribute is included in the relation as a part of the composite key, which allows users to access this information on a particular individual. This concept is discussed earlier in the Relational Model portion of this paper. Based on this

relation definition, the relation can now be transformed into the form entity JOB, with all of the associated attributes.

The transformation of these relations into entities for the Advanced Education Program System can only occur if the relations are partitioned into relations that consist of related data. For example, if the transformation of the relation R1, as defined above, is attempted, the resulting form entity would be legal according to the definition of an entity; however, the entity would contain the attributes associated with an individual's job and their educational status. A problem only arises when a user of the database is trying to obtain information pertaining to educational status only. This problem is solved by using the method of partitioning relations that is discussed above.

The functional dependencies of the Relational Model can be transformed into relationships of the Form Model by evaluating the entities and attributes included in the dependency itself. For example, consider the functional dependency

SSN --> PAST POSITIONS,

which states that PAST POSITIONS is dependent on SSN. First, it is important to note that SSN is not a relation, but an attribute. One of the rules of the Form Model is that relationships exist between entities, not attributes; therefore, the relation that contains the attribute SSN must be used in order to create the relationship, not the attribute itself. Based on the knowledge of this system's design (which shows that PAST POSITIONS is an item that contains attributes of its own) and the fact that a relation can be transformed into an entity, the relation EMPLOYEE which contains the attribute SSN, is used to create the relationship between the relations EMPLOYEE and PAST POSITIONS. The definition of the

relationships is often dependent upon the designer's knowledge of the user requirements. In this particular case, an employee is allowed to have held previous job assignments; therefore, the relationship that exists between EMPLOYEE and PAST POSITIONS could be entitled HELD. Thus the FD noted as

SSN --> PAST POSITIONS

can be represented as the Form Model structure shown in Figure 15.

This transformation process appears to be much more difficult than those that have been discussed previously, but this is due to the modifications that must be made in the designs before the transformation can occur. The relations must first be partitioned to include only related data items. Also, the FDs must be restructured in order to separate the dependencies according to their significance to each other. Although the possibility of these types of modifications have been discussed earlier, this particular process requires that these types of modifications be made initially in order to create a successful transformation.

Figure 15. Form Model Relationship Transformed from a Relational Model Functional Dependency

The transformation methods that are described above provide methods of showing that the three logical structural database designs could convert from one into the other. The transformations that have been discussed - Object-Oriented to Relational Approach, Relational to Form-Based Approach, and Form-Based to Object-Oriented - provides a full cycle conversion in the design of a database. A representation of the relationship that exists between these transformations is shown in Figure 16. This conversion process simply shows that if a database designer begins with an Object-Oriented design, he/she is able to convert this design to another form by using the transformations described above. The fact that these different types of transformations can occur shows that there is a strong association between the different design methodologies. The transformation relationships discussed here do not reflect all possible transformations. Other processes such as the transformation from Form-Based to Relational are also feasible. Transformation methods similar to those described above can be used for this purpose. Based on an analysis of the transformations that have been discussed in this report, the results for these new transformations should be just as successful.

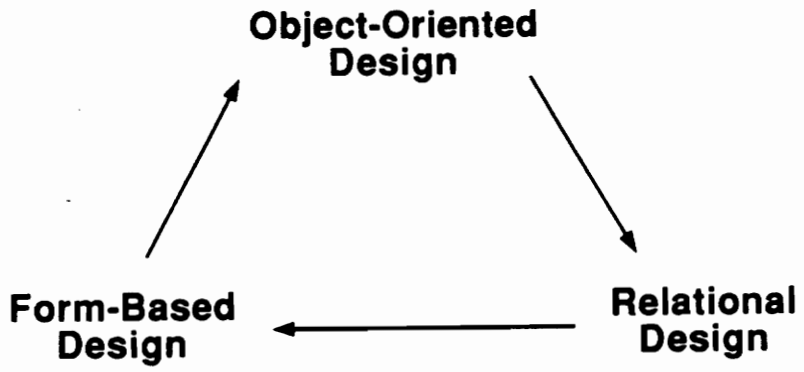


Figure 16. Full-Circle Transformation Relationship

CONCLUSIONS

During the research of the three database design methodologies discussed in this paper, several observations were made concerning these designs and the transformations between them. During the database design portion of this report, I was able to determine some key differences in the characteristics of the various design methodologies. These characteristics include the speed of design, simplicity of design, frequency of use, and understandability. Also, some of the benefits and shortcomings of the database design transformations were determined. An analysis of these observations is presented below.

The time required to design the database varied between the different approaches. The Form-Based design for this case study was the most time consuming approach. This time requirement is a result of the number of steps that must be followed in order to complete an accurate design. The Relational approach is also a time consuming method for several reasons. First, the functional dependencies must be carefully defined in order to correctly represent all possible relationships that exist in the system. Also, once these relations are determined the normalization process used to create a fully normalized system requires several iterations in order to ensure proper design.

Another difference that exists between the design methodologies is that of the level of simplicity. The Object-Oriented design approach proved to be the simplest approach to designing the Advanced Education Program Database System. This design required very few complicated steps to reach the desired goal of an effective system. The Structured Object Model (SOM), upon which

this design is based, is dependent upon an understanding of the relationship representations that exist between objects. For example, the concepts of aspect and specialization were initially somewhat difficult to understand, but were crucial in the creation of a SOM for the Advanced Education Program System. The Relational and Form-Based models were not as simple to design due to the intricacy and magnitude of these methods.

The three database methodologies discussed in this report can also be compared in terms of their frequency of use. This characteristic also affects the understandability of the designs. The Relational approach to database design is most frequently used in practice. This methodology was the easiest of the three to understand, because it is so frequently used to explain the database design process. The use of tables, or relations, allows the designer to visualize possible reports that may be retrieved from the system. By creating these structures during the design phase, it becomes very easy to match these relations, or tables, with the system query requirements.

The Form-Based methodology also presents an easily understood approach to design, because the designer is allowed to focus on a concept that is familiar to most users of the system. This design is centered around the use of forms that are to be automated in order to create the database system. One of the advantages of using this approach to design is that forms provide "an important input to the database design process that should be formalized into existing database design methodologies" (Choobineh, p. 119). In the case of the Advanced Education Program Database System, the forms are a very important aspect in the design of the system. The forms that were discussed earlier in this paper are of necessity when individuals enter into the program. Since the end-

users are accustomed to form-based work, this methodology allows the designer to use a familiar source of information in order to create a system that will provide a reliable and usable means of storing and processing data.

After analyzing the differences among the designs, based on the characteristics described above, the Relational approach was chosen as the best design for the Advanced Education Program Database. Although this approach did not produce the least time consuming design, it did provide the best means of conceptualizing the future system. Because this approach is frequently used as an instruction tool, the resulting design is the easiest to interpret and thus easiest to modify.

The database design transformations that have been discussed in this report provide a summary of possible conversions for the Advanced Education Program Database System designs. There are a few critical issues involved when considering the use of the various transformation methods. These issues include completeness, simplicity, formality, and verification and are discussed in detail below.

The completeness of design transformations is based on the issue of possible design modifications. After researching the steps involved in transforming the various database designs, it became apparent that no particular transformation was perfect. In order to completely convert one methodology into another, there are certain changes that may be required in order for the modification to be complete. An example of such a modification existed in the transformation from Object-Oriented to Relational. This particular transformation creates relations that are in Fourth Normal form; therefore, in order to create the

same relational design that was shown earlier, it was necessary to first represent these relations in Boyce-Codd Normal form.

Another consideration of database design transformations is simplicity. It is important to realize that many of the transformations are more detailed than others. For example, the transformation from the Form-Based approach to the Object-Oriented approach involved more specificity than the transformation from the Object-Oriented approach to the Relational approach. There were many more modifications that were made in order to ensure that the transformation was successful. The modifications made during database transformations are simple in most cases. However, it is necessary to conceive of their possibility so that steps can be taken in order to prevent interpretation errors once the transformation process is complete.

After creating the transformation methods for the Advanced Education Program Database System, the issue of formality surfaced. The transformation methods presented above were not based on any formal set of rules. These methods were created based on the similarities that exist between the components of the different methodologies. For example, the method of transforming a Form-Based design into an Object-Oriented design was created by examining the correlation between form entities and objects and their corresponding relationships. Once this analysis is complete, the appropriate steps can be taken to transform one component into another. These steps form a transformation method for a particular case. These methods will not be standard for all possible transformations. The transformation methods will depend on the relationships that exist between the components and relationships of various database designs.

The issue of verifying database design transformations is also important. As stated earlier, the transformation methods discussed in this report were created based on a particular case study. These transformations were verified based on the result of the design that had been created previously. For example, the method of transforming an Object-Oriented design into a Relational design was verified by ensuring that the resulting relations were representative of the relations created from the original database design. This verification method was successful for this particular case study; however, the verification of other transformations will be dependent upon the chosen system design. As the system design becomes more complicated, the verification procedures will have to be altered in order to ensure that proper results are obtained.

The transformation methods discussed in this report provide both database designers and implementors with the flexibility of adapting to an evolving work environment. As technology continues to change, the applications that support it must possess qualities that allow fast and easy modification. The continued use of these transformation processes and the creation of new transformation methods allow database systems to adapt to this changing technological environment.

REFERENCES

Attwood, Thomas M. "The Case for Object-Oriented Databases." *IEEE Spectrum.* (February 1991), 44-47.

Blaha, Michael R. et al. "Relational Database Design Using An Object-Oriented Methodology." *Communications of the ACM.* 31, 4 (April 1988), 414-427.

Cardenas, Alfonso F. Database Management Systems. Boston: Allyn and Bacon, Inc., 1985

Cattell, Roderic G.G. Object Data Management: Object-Oriented and Extended Relational Database Systems. Massachusetts: Addison-Wesley Publishing Company, 1991.

Chen, Peter P. "The Entity-Relationship Model - Toward A Unified View of Data." *ACM Transactions on Database Systems.* 1, 1 (March 1976), 9-30.

Choobineh, J., Mannino, M.V., and Tseng, V.P. "A Form-Based Approach for Database Design." *Communications of the ACM.* 35, 2 (February 1992), 108-119.

Date, C.J. An Introduction to Database Systems. Massachusetts: Addison-Wesley Publishing Company, 1990.

Delobel, Claude, et al. Deductive and Object-Oriented Databases. New York: Springer-Verlag, 1991

Hansen, Gary W. and Hansen, James V. Database Management and Design. New Jersey: Prentice-Hall, Inc., 1992

Hawryszkiewycc, Igor T. Relational Database Design: An Introduction. New York: Prentice-Hall, Inc., 1990.

Henderson-Sellers, Brian and Edward, Julian M. "The Object-Oriented Systems Life Cycle." *Communications of the ACM.* 33, 9 (September 1990), 143-159.

Higa, Kunihiko, et al. "Object-Oriented Methodology for Knowledge Base / Database Coupling." *Communications of the ACM.* 35, 6 (June 1992), 99-112.

Inmon, W.H. and Friedman, L.J. Design Review Methodology For A Database Environment. New Jersey: Prentice-Hall, Inc., 1982.

Jackson, Glenn A. Relational Database Design with Microcomputer Applications. New Jersey: Prentice-Hall, Inc., 1988.

Jacobs, Barry E. Applied Database Logic I: Fundamental Database Issues. New Jersey: Prentice-Hall, Inc., 1985.

Lewis, Ted G. Microbook: Database management for the IBM Personal Computer. Oregon: Dilithium Press, 1983.

Maciaszek, Leszek A. Database Design and Implementation. New York: Prentice-Hall, Inc., 1990.

Nijssen, G.M. and Haplin, T.A. Conceptual Schema and Relational Database Design. New York: Prentice-Hall, Inc., 1989.

Teory, Toby J. Database Modeling and Design: The Entity-Relationship Approach. California: Morgan Kaufmann Publishers, Inc., 1990.

Tsichritzis, Dionysios C. and Lachorsky, Frederick H. Data Models. New Jersey: Prentice-Hall, Inc., 1982.

Yao, S.B. and Kunii, T.L. Database Design Techniques. New York: Springer-Verlag, 1982.