

DESIGN AND DEVELOPMENT OF A
THERMODYNAMIC PROPERTIES
DATABASE USING THE
RELATIONAL DATA
MODEL

By

PARIKSHIT SANGHAVI

Bachelor of Engineering

Birla Institute of Technology and Science,

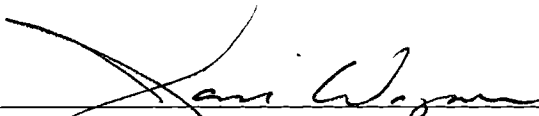
Pilani, India

1992

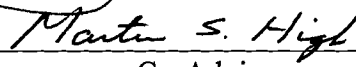
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1995

DESIGN AND DEVELOPMENT OF A
THERMODYNAMIC PROPERTIES
DATABASE USING THE
RELATIONAL DATA
MODEL

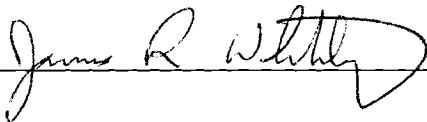
Thesis Approved:



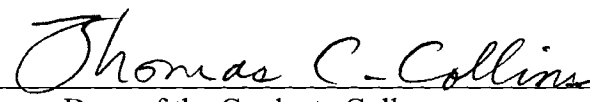
Thesis Adviser



Co-Adviser







Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to thank my adviser Dr. Jan Wagner, for his expert guidance and criticism towards the design and implementation of the GPA Database. I would like to express my appreciation to Dr. Martin S. High, for his encouragement and support on the GPA Database project.

I am grateful to Dr. Khaled Gasem, for helping me gain a better understanding of the aspects pertaining to thermodynamics in the GPA Database project. I would also like to thank Dr. James R. Whiteley for taking the time to provide constructive criticism for this thesis as a member of my thesis committee.

I would like to thank the Enthalpy and Phase Equilibria Steering Committees of the Gas Processors Association for their support. The GPA Database project would not have been possible without their confidence in the faculty and graduate research assistants at Oklahoma State University. I wish to acknowledge Abhishek Rastogi, Heather Collins, C. S. Krishnan, Srikant, Nhi Lu, and Eric Maase for working with me on the GPA Database.

I am grateful for the emotional support provided by my family. Lastly, I would like to thank my roommates and friends at Stillwater for their company, friendship, and support.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION	1
Motivation.....	2
Research Objectives.....	3
Organization of the thesis	4
II. BACKGROUND CONCEPTS.....	6
Types of Databases	6
Flat File	6
Hierarchical.....	8
Network.....	13
Relational	15
Data Structure	16
Attributes.....	16
Tuple	17
Relation.....	17
Candidate Key.....	18
Primary Key.....	18
Relationships.....	18
Entity-Relationship Diagram	20
Data Manipulation	20
Data Integrity	23
Object-Oriented.....	24
Objects	24
Encapsulation.....	25
Classes.....	25
Inheritance Hierarchies	26
Identity	26
Chemical Engineering Data	27
Current Chemical Engineering Database Technology	30
III. DATABASE MANAGEMENT SYSTEM SELECTION.....	32
Current Trends	32
Performance Factors	34
Data Types	34
Speed.....	35

Query Capabilities	35
Data Integrity	36
Domain Validation.....	36
Safety	37
Hardware.....	37
Data Sharing.....	38
Security	38
Interface	39
IV. DESIGN PROCEDURE	40
Data assessment	41
Data Analysis	42
Format Analysis.....	42
Error Checking.....	43
Data Transfer.....	44
Entity-Relationship model	45
Preliminary E-R Model.....	47
Pseudo-normalized E-R model	49
Logical Database Design	55
Transformation.....	56
Entities	56
Relationships.....	57
Normalization	60
First Normal Form	61
Second Normal Form.....	67
Third Normal Form.....	76
Physical Database Design	83
Data Types	84
Domains	86
Indexes	87
Relationships.....	88
Application Development	88
Data Retrieval	89
Data Maintenance	90
Data Manipulation	91
Graphs.....	91
Spreadsheets.....	92
Text Files	93
Database independence	94
Summary	95
V. SUMMARY.....	98
VI. CONCLUSIONS AND RECOMMENDATIONS	100

BIBLIOGRAPHY	102
GLOSSARY OF TERMS	104
GLOSSARY OF ACRONYMS.....	110
APPENDICES	112
APPENDIX A -- SAMPLE DATA FILE CONTAINING VLE DATA	113
APPENDIX B -- FORTRAN CODE FOR FORMAT CHANGE.....	115
APPENDIX C -- FORMATTED TEXT FILE FOR MS ACCESS.....	117
APPENDIX D -- VLE DATA FORMAT.....	119
APPENDIX E -- DATABASE RELATIONSHIPS	121
APPENDIX F -- DATABASE DICTIONARY	123

LIST OF FIGURES

Figure	Page
1. Hierarchical Data Model.....	10
2. Multiple Parents for the Hierarchical Data Model.....	10
3. Multiple Parent Implementation using Separate Database Structures.....	11
4. One-to-One and One-to-Many Relationship Cardinalities.....	12
5. Hierarchical Data Model for a Pure Properties Database.....	13
6. Network Data Model for a Pure Properties Database.....	15
7. Classification of the Properties of MatériaIs.....	29
8. E-R Diagram Nomenclature.....	46
9. Preliminary E-R Model for the GPA Database.....	48
10. A One-to-Many Relationship in the GPA Database.....	51
11. ISA Relationships in the GPA Database.....	51
12. Representation of Multi-Component VLE Data.....	53
13. Pseudo-Normalized E-R Model for the GPA Database.....	54
14. Steps in Normalization.....	62
15. Normalization of Table 10 (VLE Data) from 1NF to 2NF.....	74
16. Relationships between VLE Data Tables in 2NF.....	74
17. E-R Diagram of VLE Data in 3NF.....	78
18. Relationships between VLE Data Tables in 3NF.....	78
19. E-R Diagram of GPA Database in 3NF.....	82
20. Logical Data Model for VLE Data.....	83

LIST OF TABLES

Table	Page
1. Chemical Engineering Databases.....	31
2. Component -- Tabular Representation of the entity "Component"	57
3. Data Point -- Data Point Records.....	57
4. VLE -- Data Point Attributes Specific to VLE Data.....	58
5. Data Set -- Information Pertaining to Data Sets	58
6. Revised Data Point -- Data Point Records with a Foreign Key Include	59
7. Data Set-Component -- Representation of a M:N Relationship	60
8. VLE Data Flat-file Import Table with Multivalued Attributes.....	63
9. VLE Data with Repeating Groups	64
10. VLE Data in 1NF	65
11. VLE Data without DSN partial dependency	69
12. DS_VLE -- Data Set Level VLE Data in 2NF.....	69
13. VLE Data without DSN and DPN partial dependency	71
14. DP_VLE -- Data Point Level VLE Data in 2NF	72
15. DP_VLE_XY -- Data Point Level VLE Data for Mole Fractions in 2NF.....	73
16. DS_COMP -- Data Set Level Data for Component Information in 2NF.....	73
17. REFERENCE -- Reference Information such as Title and Source.....	75
18. COMPONENT -- Component Names	75
19. DS_VLE (3NF) -- Data Set Level VLE Data in 3NF	77
20. REFERENCE (3NF) -- Modified Reference Information Table in 3NF.....	77
21. DATA_SET -- Common Data Set Level Information for all Data Types.....	82

CHAPTER I

INTRODUCTION

Thermodynamic data pertain to the state of a pure compound or mixture measured in terms of intrinsic variables such as temperature and pressure. Thermodynamic data can take many forms depending on the quantities being recorded. These forms of data (phase boundary data, entropy data, etc.) can be stored as records and attributes. When multiple records are stored in a particular format, the resulting collection is called a database. The relational data model is one such format for storing data.

The relational data model was created by E. F. Codd at IBM (Codd, 1969). The key elements underlying this model are that all data are stored in the form of tables where each row is a record and each column is a data attribute. The model allows for relationships between tables, thus leading to a comprehensive database formulated from a collection of tables.

Data stored using the relational data model do not make much sense to a casual user, unless they are presented in a form similar to the original form in which they were assembled before being converted to a relational database. The tools used to achieve this presentation, along with the relational database engine, comprise what is known as a DataBase Management System (DBMS).

One of the most powerful features of a DBMS is its ability to allow the user to query a database for specific information. By means of a query, an end-user can obtain

data that match specified criteria. Thus, data from a large number of sources can be collated into a relational database consisting of several tables without significantly increasing the time required to obtain specific data from a single table.

Motivation

Thermodynamic data are a primary requirement for process development in the chemical industry. One of the many uses is to check the accuracy of models by comparing calculated values with the experimental data stored in the database. Thermodynamic properties are also used in equipment design for calculating operating parameters such as heat capacity, flash point, etc.

Over the years, thermodynamic data collected by scientists and engineers have traditionally been disseminated by means of published papers in technical journals, research reports, and books. Thus, the data are not concentrated at one convenient source. If a process or model developer needs thermodynamic data for a particular set of operating conditions, he may need to search numerous technical publications before he can locate the required data.

The disadvantages inherent in this method of reporting thermodynamic data have been partially overcome by means of collecting and storing the data in a large data repository. The prevalent approach used for arranging the data has been the flat-file approach. The drawback of this approach is that it may take an unacceptably long time to sift through the large collection while searching for specific data, since the indexing

method used in the flat-file approach is limited to file names. The flat-file approach is not sufficient for identifying a set of thermodynamic data uniquely.

Taking into consideration the advantages of a formal database structure, such as the relational data model, over the more primitive flat-file approach, it becomes apparent that thermodynamic databases are prime candidates for relational database implementation. Therefore, it is surprising that the flat-file system is still the prevalent method of creating databases for chemical engineering applications. This could be attributed to a general lack of awareness among the chemical engineering community regarding the concepts pertaining to a relational database. Another factor influencing the ineffective use of database technology in the chemical industry is the dearth of literature on the topic of implementing a chemical engineering database using the relational data model.

Research Objectives

Chemical engineering databases differ from conventional databases in the kind of information that they store. One cannot assume that the data models used for business oriented databases will be suitable for storing other kinds of data, such as chemical engineering data. The objective of this thesis is to evaluate the suitability of the relational data model for chemical engineering data and to provide a generalized technique for implementing chemical engineering databases using the relational data model. This technique is presented as a formal methodology using a thermodynamic properties

database as an example. This thermodynamic properties database belongs to the Gas Processors Association (GPA), and was converted from a flat-file format to a relational database using Microsoft Access[®].

The methodology presented in this thesis is aimed at developers and managers of technical databases -- in particular chemical engineering databases. It is hoped that this methodology will explain some of the complexities involved in the implementation of a technical database using conventional tools. It is also hoped that this methodology will facilitate some of the core techniques employed for more conventional databases in becoming common tools used by technical database developers in the chemical industry. A detailed discussion of the implementation of the GPA Database is used to illustrate the effectiveness of relational database technology. The improvements achieved by this implementation are highlighted to demonstrate the value added to the data by the transition from the flat-file model to the relational data model.

Organization of the thesis

Types of database models are discussed in Chapter II. These include the flat-file, hierarchical, network, relational and object-oriented data models. This chapter also lists and briefly describes examples of chemical engineering data and their uses. Also included in this chapter are the results of a literature review of the status of database technology in chemical engineering.

Three database management systems Microsoft(MS) Access, Microsoft FoxPro[®], and Paradox[®] are compared in Chapter III. This comparison is used as a basis to illustrate the various factors pertinent to the selection of a relational DBMS for a chemical engineering database.

Chapter IV starts with a description of data assessment techniques pertinent to chemical engineering databases. The formal design procedure is then presented, as the design is taken through the stages of (a) entity-relationship modeling, (b) logical design, and (c) physical design for the vapor liquid equilibrium data in the GPA database. The reader is exposed to the theory behind each of these design steps and the advantages to be gained from the relational implementation of a flat-file database are discussed. The various steps involved in creating a stand alone application for accessing the underlying database are discussed at the end of the design procedure. The chapter ends with a summary of the design procedure in the form of an algorithm.

The results of the relational implementation of the GPA Database are discussed in Chapter V. The advantages associated with the relational data model are summarized in a point-wise fashion in this chapter.

The final chapter draws conclusions based on the results of the relational implementation of the GPA Database. Also given in this chapter are recommendations for the future of the GPA Database.

CHAPTER II

BACKGROUND CONCEPTS

Types of Databases

Flat-file

A flat-file is the basic unit for storing records in a database. The flat-file is a text based file with the text arranged as a table. As in a table, a flat-file contains column headings to denote the different data attributes. Each row of the file represents a record. The individual data attributes for each record are stored along the same row. Each data attribute is stored within the space allocated for it (exactly below its column heading). This helps to differentiate between data attributes by means of their position within the file.

The flat-file approach to data modeling entails the creation of a separate file for each record type. Thus, a flat-file database consists purely of text based files. The name of each file reflects the type of data that is stored in that file. The advantages to the flat-file approach are that it is extremely simple to implement and can be transferred across different platforms with ease. It is also a natural way to represent records, and it does not require special database skills from the end-user.

The flat-file approach has a number of inherent disadvantages in its structure that far outweigh any advantages it offers in the implementation of a large database. Some of these disadvantages are illustrated below using an example of a database to store employee information (McFadden, 1993). Two basic record types needed for this database are:

1. Employee: (Employee Name, Employee Address, Department Name)
2. Department: (Department Name, Department Head, Department Telephone No.)

The Employee File in this example contains the name and address of the employee and the department that he or she works for. The Department File contains the name and telephone number of the department and the name of the head of the department. Now, if the company decides to change the name of one of its departments, the database administrator would have to update the Department File. But it is not so obvious that he or she also would have to change all the corresponding entries for this department name in the Employee File. This is called a form of update anomaly.

Flat-file databases can also exhibit another form of data integrity error called a deletion anomaly. This occurs when a record is deleted in one file, while it is still being referenced from another file. Consider a database containing water solubility data for inorganic salts. This database contains two record types "DATA_POINT" and "DATA_SOURCE." The structures of these record types are:

1. DATA_POINT (Compound Name, Solubility, Data Source Number)
2. DATA_SOURCE (Data Source Number, Publication, Author, Date)

In the above record structure, "Compound Name" refers to the name of the inorganic salt for which the solubility data are being recorded; "Data Source Number" refers to the source from which the solubility data were obtained. Now, if a data source is to be deleted from the database due to the presence of unreliable data, it would seem natural to simply remove the corresponding record from the file containing the record type "DATA_SOURCE." This simple deletion can cause a deletion anomaly, because records in the "DATA_POINT" file that reference the deleted data source might still exist.

Another drawback associated with the flat-file approach is the overhead associated with any change in record type structure, for example, the addition of an extra data attribute to store temperature in the solubility database. This would involve reformatting the complete file, row by row, to maintain the associativity of the column heading with the data values.

Finally the flat-file system does not implement any formal methodology to order the data within the file itself. This makes data retrieval from the file highly inefficient.

Hierarchical

The hierarchical data model was developed during the same time as the file based approach described above. In fact, the earliest database management systems used commercially were based on the hierarchical data model (McFadden, 1993). The most popular of the hierarchical database management systems, IMS, was jointly developed by

IBM and North American Aviation to address the data processing needs of the aerospace industry.

The hierarchical data model has been used for at least twenty years, and it is likely to be used for many more years because of its status as a legacy database. Its basis lies in the assumption made by its inventors at IBM that all views of life are hierarchical in nature. This fundamental concept underlying the hierarchical data model is the basis for many of the current DBMS's. In fact many of the databases in use today are managed by a hierarchical database management system.

In the hierarchical data model, various record types are arranged as an inverted tree as shown in Figure 1. This structure logically follows a top down approach. Each node in the tree is a record type. In this model a higher record type is called a parent, and a lower record type is called a child. In the example shown in Figure 1 DEPARTMENT is the parent of STUDENT; GRADUATE is the child of STUDENT. The two basic constructs of the hierarchical data model are:

1. A parent record type can own any number of child record types.
2. A child record type can belong to only one parent.

It would seem that TEMPORARY is a natural child for STAFF and FACULTY as shown in Figure 2. But due to the limitation of a child belonging to only one parent, it is not possible to implement this natural order using the hierarchical data model. This construct is the limitation inherent in the hierarchical data model.

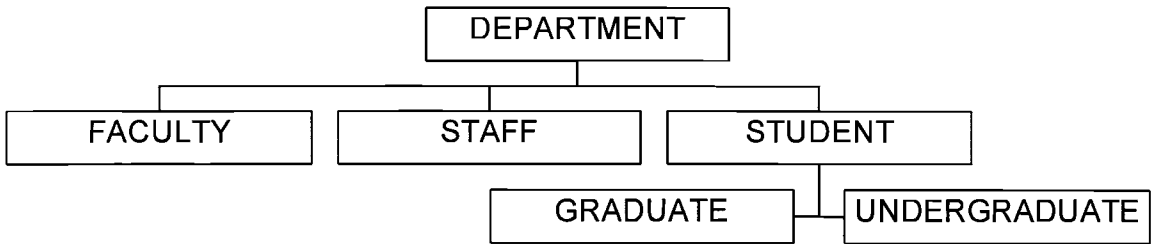


Figure 1. Hierarchical Data Model

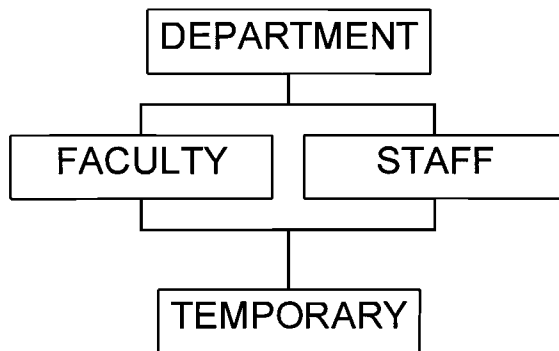


Figure 2. Multiple Parents for the Hierarchical Data Model

One way to work around this limitation is to create two instances of TEMPORARY and assign them as shown in Figure 3. Some hierarchical database management systems allow multiple parents as shown in Figure 2. This involves extra processing overhead, and is not an efficient solution.

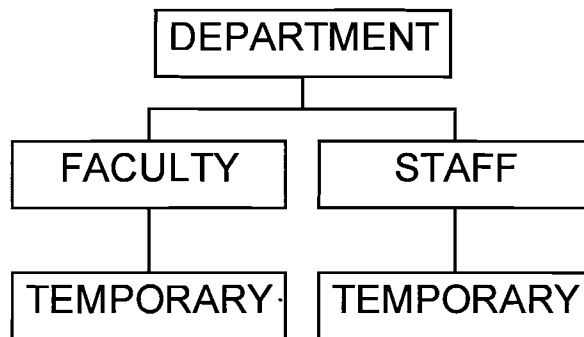


Figure 3. Multiple Parent Implementation using Separate Database Structures

The physical implementation of the hierarchical data model supports one-to-one (1:1) and one-to-many (1:N) relationships between parent and child. This is known as the cardinality of the relationships supported by the database. A 1:1 relationship is the case where each record in the parent cannot have more than one associated record in the child. In contrast, in a 1:N relationship each record in a parent record type can have more than one associated record in the child. An example of a 1:N relationship is given in Figure 4. Here an EMPLOYEE can have many SKILLS (1:N) but he can have only one SPOUSE (1:1).

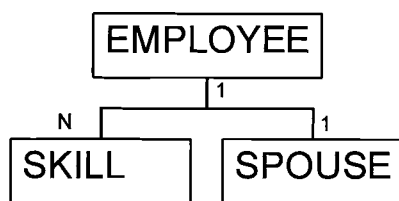


Figure 4. One-to-one and One-to-many Relationship Cardinalities

Figure 5 gives an example of the use of the hierarchical data model for a pure properties database. All compounds can be broadly classified into organic compounds, inorganic compounds, and elements, as shown in the figure. This segregation is required because each class of compounds has a different information storage requirement. Thus we might need to store extra information for organic compounds such as saturated/unsaturated, cyclic/acyclic, etc. These attributes are particular to organic compounds and need to be stored in a separate record type called “Organic compounds.” Elements on the other hand might have different information storage needs, such as valence number or electronegativity/electropositivity. Assuming that inorganic compounds do not have any special information storage requirements, one could argue that the creation of a separate record type called “Inorganic compounds” is unnecessary. This may not always be true. For example, if the entity in question (“Inorganic compound”) is conceptually different from its counterparts, then there is always a possibility of specialized information storage requirements arising for that entity at a later date. This specialized information would be impossible to store unless provisions had

been made to treat inorganic compounds as a separate record type. The information common to all classifications is stored in the parent record type “Compounds.”

Each of these classifications can be further divided into solid, liquid, and vapor. These record types are used for storing information specific to that record type. Thus the record type “solid” would have attributes such as form (amorphous, crystalline, etc.) and average particle size. The “Liquid” record type could store information on interfacial tension, vapor pressure, etc.

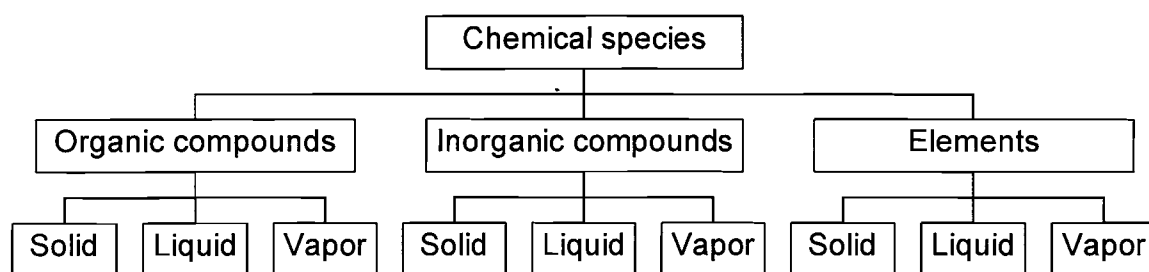


Figure 5 Hierarchical Data Model for a Pure Properties Database

Network

The network data model was created by the CODASYL (Conference on Data Systems Languages) committee (CODASYL, 1971). This committee was originally formed to discuss possible improvements for COBOL (Common Business Oriented Language). The committee discovered that the changes required for COBOL were too

drastic for a smooth transition and proposed a new data model called the network data model. The implementation to be used for this data model was named CODASYL.

The network data model is an extension of the hierarchical data model in that it does not distinguish between parent and child. This helps to remove the need to create redundant record types. Figure 2 is an example of the network data model. The network data model also allows the database designer to create any level of structure so that the flexibility of design is very high.

The simple network data model using the CODASYL implementation allows 1:1 and 1:N relationships. The other type of implementation is the complex data model. It is called complex because it allows M:N or many-to-many relationships. A many-to-many relationship is one where each of the parent records can be associated with many child records and each of the child records can also be associated with many parent records. The complex implementation is not very common, because it does not have a significant number of advantages over the simple network implementation.

The network data model equivalent of Figure 5 is given in Figure 6. Since all three record types “Organic compounds”, “Inorganic compounds”, and “Elements” can exist as solid, liquid, or vapor; the network data model allows them to be made common and connected. This networked data structure removes any redundant record types present in the earlier hierarchical data structure.

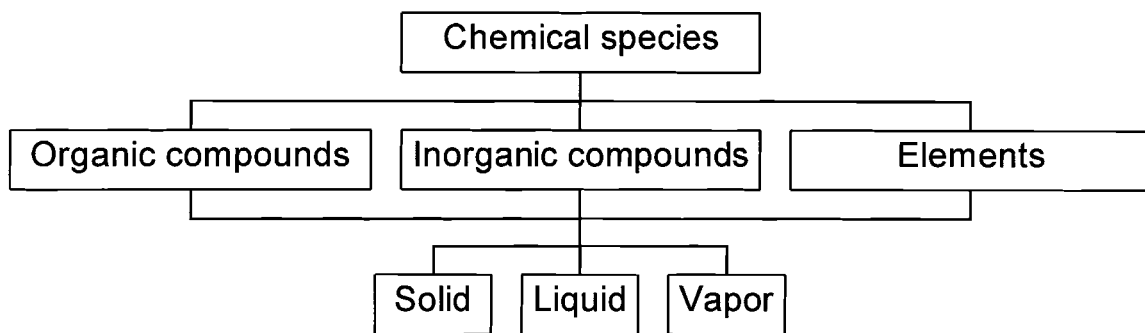


Figure 6 Network Data Model for a Pure Properties Database

Relational

The relational data model has seen widespread industry acceptance since its conception at IBM in 1969 by Codd (1969). It uses mathematical set theory as its basis. The relational data model owes its existence to the need for a formal database theory during the 1960's. Codd realized this need and went about setting up the relational data model with rigorous definition requirements. This helped elevate database design from an art to a science.

The relational data model consists of the following three components (Fleming, 1989):

1. Data structure -- data are organized in the form of tables.
2. Data manipulation -- operations are used to manipulate data in the database.
3. Data integrity -- facilities are provided to specify rules that maintain the integrity of data during manipulation.

The three components outlined above are what a DBMS tries to implement with the greatest efficiency and ease of use. To fully grasp the concept of a relational database, it is necessary to examine each of these concepts in more detail.

Data Structure

The data structure of a relational database is made up of smaller components put together in an orderly fashion. Described below are some of the key components of a relational data structure.

Attribute. Each item of data that needs to be stored for any given entity is called a data attribute for that entity. Thus, each entity will normally have many data attributes depending on the amount of information that needs to be stored. Consider, as an example, an entity called “Employee.” The various data attributes that would have to be stored for this entity would be name, address, phone number, etc. Thus, if we consider each line of a flat-file data base to represent an employee, then each data attribute corresponding to the given employee would be stored along that line. In addition to being stored on the same line, they would also have a vertical ordering (that is, each attribute would be stored within the column(s) allotted). This is akin to the concept of a field or cell in a spreadsheet.

For the pure properties database shown in Figure 5, name, molecular weight, formula, melting point, etc., are some of the data attributes associated with the entity “Chemical Species.”

Tuple. A tuple is a collection of data attributes pertaining to the same entity. It is the relational equivalent of a record. A record normally contains all the attributes that the template it is based on specifies. The creation of a template is based on the information storage requirements of a given entity. Thus a record or tuple helps to group the various data attributes that a database might contain. In a flat-file database a tuple is one line of information; it is equivalent to a row in a spreadsheet.

In the employee example outlined above, the attributes name, address, phone number, etc., when taken collectively form a tuple of the employee. Correspondingly, for the pure properties database of Figure 5, the tuple or record of a chemical species would consist of the data attributes name, molecular weight, formula and melting point taken together.

Relation. A relation is a collection of records having the same template. A relation or table stores information on the various entities belonging to a given entity type. It is the equivalent of a flat-file or a spreadsheet. A relation is so named because data attributes stored in the record type or tuple on which it is based are all related.

A collection of employee records, from the example above, stored one below the other in the form of a table constitute a relation in a relational database. Similarly a

collection of records for various chemical species such as methane and ethane would form a relation.

Candidate Key. Any attribute or collection of attributes that can uniquely identify a record in a relation is known as a candidate key for that relation. In the employee example above, Name and Social Security Number are candidate keys for the “Employee” relation. For the pure properties database, Name is the only candidate key for the relation “Chemical Species.”

Primary key. The key that is chosen from the available candidate keys to uniquely identify a record in a relation is known as the primary key. It is imperative for each relation to have a primary key to enable retrieval of individual records. This primary key can sometimes be a derived field created solely for the purpose of unique identification. The Social Security Number is an example of a derived primary key used in Federal databases. For the pure properties database it would be prudent to create a new attribute called “Chemical Species Number” purely for the purpose of serving as a primary key even though the attribute Name is a valid candidate key. This is because one of the requirements of a primary key is that it remain unchanged throughout the life of the database

Relationships. Relationships are used to interconnect different relations or tables. This helps a database designer to allocate the information storage requirements of the

database among tables which are then connected by means of relationships. The connection is more theoretical than physical.

Consider the example shown in Figure 4 above. EMPLOYEE and SKILL would have the following structures:

- EMPLOYEE(SSN, Name, Address, Phone)
- SKILL(SSN, Description, Year acquired)

Here SKILL is meant to store information about the various skills possessed by an employee in an organization. It stores a description of the skill and the year that the skill was acquired. The relationship existing between the two entity types is implemented by comparing the SSN in EMPLOYEE and SKILL. Thus, all records in the two tables that have the same SSN's are related. This is the basis of a relationship in a relational database. The relational data model supports all possible cardinalities of relationships. Thus two entities can have a 1:1, 1:N, or a M:N relationship. However during physical implementation of the database a M:N relationship is normally mapped onto two 1:N relationships to facilitate querying of the database.

The water solubility database example used earlier in the section on flat-file databases also has a 1:N relationship between the two relations "DATA_POINT" and "DATA_SOURCE" contained in it. This relationship would be implemented by comparing the field "Data Source Number." Thus each data source can be referenced by many data points and, hence, has a 1:N relationship with the entity "DATA_POINT."

Entity-Relationship Diagram. The data structure of a relational database is the key factor in the performance of a database. A good database structure helps maintain the integrity of the data and also allows for efficient queries and updates. There are various methodologies available today to design database structures, the most widely used being the entity-relationship diagram.

The basic aim of the entity relationship diagram is to structure the available information into an ordered schema -- a description of the overall logical structure of the database. This schema is then mapped onto the relational data model to create a relational database. Thus, the entity relationship diagram acts as an interface between the information being modeled and the relational data model.

The entity relationship diagram is effective because of its ability to separate the available information into distinguishable entities. There are various CASE (Computer-aided software engineering) tools currently available to create sound data structures using entity-relationship diagrams of various kinds.

Data Manipulation

Data stored in a database are not very useful, unless there exists a means to manipulate them into a form suitable for use by the end-user of the database. This is where the power of the relational data model comes into the picture.

Relational set theory states that when data are modeled using the relational data model, they can be manipulated into any required form for further processing. The tools

used to achieve this manipulation are the cornerstone of a relational DBMS. They are implemented in the form of SQL (structured query language -- pronounced as “sequel”).

SQL has been accepted as the standard 4GL (fourth generation language) for data creation and manipulation by the American National Standards Institute (ANSI). The first standard was laid down in 1986 (X3H2, 1986) and newer standards are created as the need arises. The last standard was adopted in 1992. Currently the SQL3 standard is under development by the ANSI SQL committee which includes, among others, representatives from the various DBMS vendors. Despite SQL having been laid down as a standard, there exist as many dialects of SQL as there are DBMS's.

SQL, like any other 4GL, consists of a basic set of keywords or commands which, when used with the right syntax and parameters, will achieve the desired result. SELECT, FROM, and WHERE are the three most commonly used keywords in SQL. SELECT is used to specify the fields or attributes that are to be returned or manipulated, FROM specifies the table or relation where the required attributes can be found, and WHERE is a tool for imposing restrictions on the records or tuples that are turned up by the search.

The simple example given below helps illustrate the syntax to be used while using these three keywords. The EMPLOYEE entity type discussed in an earlier example is used as the table upon which the query illustrated below is going to act.

SELECT Name, Address

FROM EMPLOYEE

WHERE Phone = “405*”

This query will result in the names and addresses of all the employees, from within the EMPLOYEE table, who have their telephone number starting with 405.

The three basic functions of SQL are (a) database definition, (b) data manipulation, and (c) data retrieval. Database definition is the process of creating the various tables that a database might contain. For instance, this could involve the specification of the number and types of fields that are to be contained in a table. Data manipulation is the process of modifying data to reflect current values. Thus, if the telephone number of an employee was to change, we would have to use the data manipulation functions provided by SQL to make the necessary changes to the record in the EMPLOYEE table. Data retrieval functions are used while querying the database for specific information. The SQL syntax example given above is an example of a data retrieval function.

Data retrieval using SQL can also be implemented for querying multiple tables simultaneously. An example of how this is implemented is given below for the water solubility database, acting on the two tables "DATA_POINT" and "DATA_SOURCE."

```
SELECT Compound Name, Solubility  
FROM DATA_POINT, DATA_SOURCE  
WHERE (DATA_POINT.Data Source Number = DATA_SOURCE.Data Source  
Number) AND (Author = "*Perry*")
```

This SQL statement will result in a listing of the compound names and solubilities for all the compounds that have a data source with Perry as one of the names of the author. This SQL statement queries the two tables "DATA_POINT" and

“DATA_SOURCE” as is evident from the **FROM** statement. The first part of the **WHERE** statement is the implementation of the one-to-many relationship using “Data Source Number” as the comparison field.

Data Integrity

This part of the relational data model is the means by which the data contained within the database are kept error free. Integrity issues concerning data in a relational database include conformity to data domains specified during database definition and referential integrity.

This conformity issue is usually database specific. This is because each area of application of the relational data model may have its own set of business rules. For example a banking database may have as one of its data integrity rules the condition that account withdrawal amount may not exceed account balance. Whereas if one were to consider the application of the relational data model to a thermodynamic database used for storing vapor-liquid equilibrium data, a commonly used data integrity rule would be that the sum of vapor or liquid mole fractions should equal one.

Referential integrity is a by-product of the relational implementation of any database. As an example, if one were to have a record on the many side of a 1:N relationship but omit the corresponding record on the one side, a basic referential integrity rule would be violated.

Object-Oriented

The object-oriented data model (OODM) derives its basic concepts from two different fields: data modeling and object-oriented programming (Coad, 1990). It has been predicted that the OODM is the future in the database industry. In fact, there are a number of DBMS's that are based on a marriage of the relational and object-oriented data models. However, due to the lack of standards, this technology has yet to become a major player in the database market.

Objects, encapsulation, classes, inheritance hierarchies, and identities are some of the core concepts of the OODM (McFadden, 1993). These concepts are briefly discussed below.

Objects

Real world entities are modeled for an OODM using the abstraction mechanisms provided for creating objects. These objects store information about the state and behavior of the real world entities they represent. The state of the object is represented by means of data attributes pertaining to the real world object. Similarly, the behavior of the object is captured in the form of methods (or stored procedures) that the object executes. This represents a break from the other data models, because none of them store procedures or operations along with the data.

Consider C1 as an abstraction of the real-world object methane. Common data attributes for C1 would be molecular weight, melting point and boiling point. An example of a method associated with C1 would be to determine the state of the object (vapor, liquid, or solid) given T (temperature) and P (pressure).

Encapsulation

Encapsulation is the means of hiding all unnecessary information about the object from the outside world. Thus the only method of interaction that a user of the object has with the object itself is through an established interface. This interface would normally comprise a syntax for the messages that this object can receive and act upon. This helps to hide unwanted details from the user of the object, and reduces the complexity of the database.

In our example, the interface to the object would hide details about how the state is calculated from the given T and P. This means that the user needs to be concerned only with providing the input of T and P and not with the method for calculating the state of the object.

Classes

Classes are a means of assembling objects that have the same or similar attributes and methods. Thus, all the different types of molecules known could be included in a

class called SUBSTANCE. All known substances have a boiling point and a melting point. These would be attributes of the class SUBSTANCE. The actual implementation of the methods associated with each object would be different, but the user would not need to keep track of these details due to encapsulation.

Inheritance Hierarchies

Inheritance hierarchies are a means of efficient classification of objects and classes using a tree like hierarchy structure similar to the hierarchical data model structure. The key to the efficiency obtained from an inheritance hierarchy lies in the inheritance of all attributes and methods associated with the parent by the child. Thus, if the class SUBSTANCE was to have a child RADIOACTIVE, then all the attributes associated with SUBSTANCE would be inherited by RADIOACTIVE. This helps to reduce redundancy and to enhance data integrity.

Identity

The identity of an object in the OODM is the equivalent of the primary key found in the relational data model. This identity is a value that never changes. Thus it is maintained at a level external to the data attributes and methods.

One of the primary advantages enjoyed by an object-oriented DBMS is the wide domain range supported. One would find it difficult to store procedures as part of a

record in a relational database. Here the OODM is able to surpass the relational data model because of the wide range of domains supported by this type of DBMS.

Chemical Engineering Data

Chemical engineering data are the framework on which most process and equipment design decisions are based. This makes the source of data used for the design process crucial. Chemical engineering data can be considered a subset of all material properties as shown in Figure 7 (Zeck, 1982). It is also clear from the figure that most of the data used for chemical engineering process design come under the classification of thermodynamic properties data.

Thermodynamic data are primarily applied in two different chemical engineering fields -- model development and process design. Model development comes under the domain of universities and research organizations. Here new models used for thermodynamic property prediction are tested against the available base of experimental thermodynamic data. This helps refine the models being developed to better reflect thermodynamic properties over a wider range of conditions than would be possible using experimental data alone.

Process design can be said to encompass the fields of equipment design, process development, and process optimization. All of these areas require some way of predicting operating conditions to achieve their respective goals. These operating conditions can either be experimentally determined, or they could be predicted by

models. Either way, there needs to be an underlying collection of experimentally determined thermodynamic data. This illustrates the importance of thermodynamic data in the chemical industry and in academia.

The data contained in the GPA Database form a subset of the thermodynamic data classification in Figure 7. The different thermodynamic data types covered by this database include:

- Vapor-liquid equilibrium
- Vapor-liquid-liquid and vapor-liquid-solid equilibrium
- Phase boundary (dew point and bubble point)
- Hydrate formation
- Enthalpy departures (pure components and mixtures)
- Enthalpy of solution

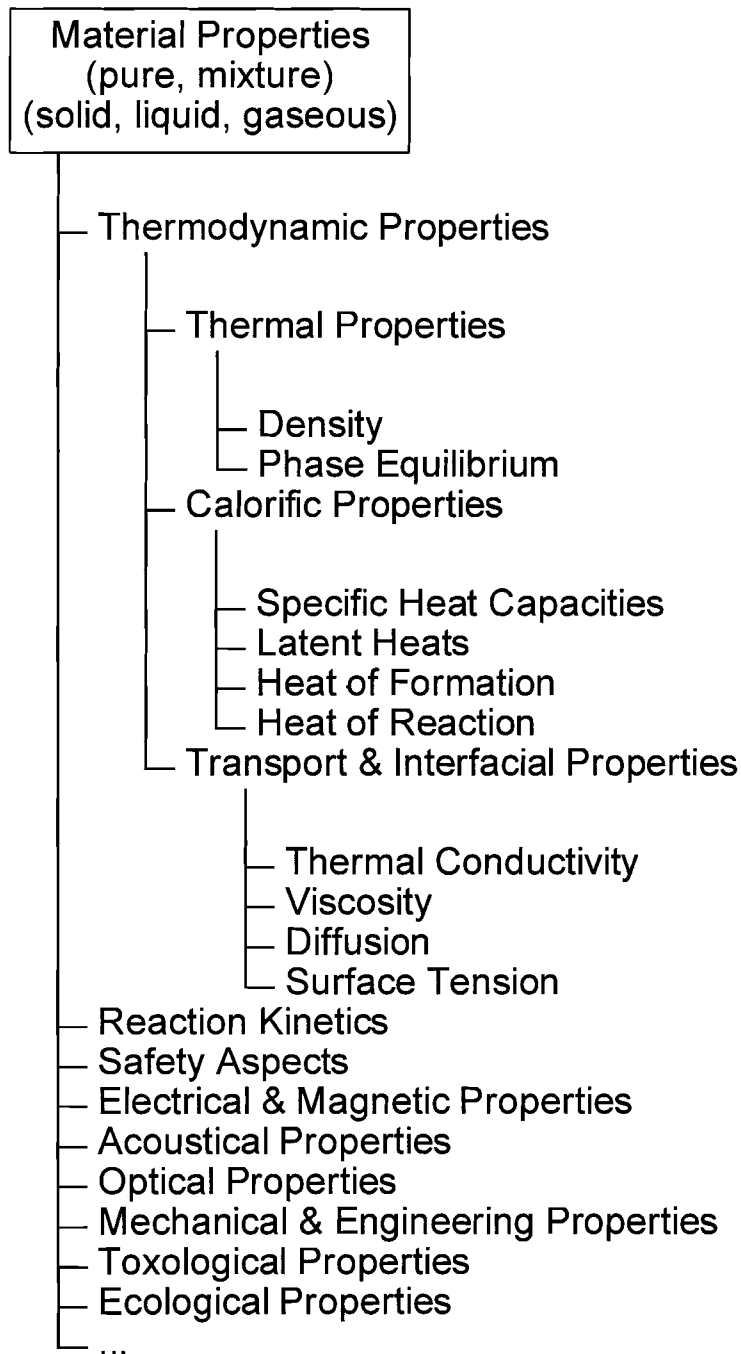


Figure 7 Classification of the Properties of Materials (Modified from Zeck, 1982)

Current Chemical Engineering Database Technology

As explained in the previous section, chemical engineering data lie at the heart of process design and model development. Most chemical processing companies and model developers have accumulated vast quantities of data that are currently being stored as flat text files. Due to the nature of a flat-file database, only a few individuals who are familiar with the format of the data files are able to use these databases effectively.

Fortunately database technology in the chemical process industries (CPI) is at an evolving stage. The evolution is towards structuring the vast quantities of existing data using established models such as the relational data model. The main factors influencing this evolution are the cost effectiveness of the personal computer (PC) and the availability of database management systems to model this data. The result is that the data stored in these databases become accessible to a very wide range of users due to the user-friendly interface provided by these DBMS's.

Table 1 is a list of current chemical engineering and thermodynamic properties databases that are commercially available. All have implemented some form of formal data structuring and have provided a user-friendly interface that allows users not familiar with the structure of the database to access the data.

Table 1

Chemical Engineering Databases (Modified from Green, 1994)

TITLE	CONTENTS	DATA	PROPERTY TYPE	ON-LINE PROVIDER	PLATFORM
DIPPR	pure components (industrial chemicals)	experimental, calculated	physical and safety properties	STN	PC (DOS), mainframe
TRC	pure components, mixtures	experimental, calculated	vapor pressure	TDS	PC (DOS)
Floppy Books	pure components, mixtures	experimental, calculated	liquid viscosity, virial coefficients	---	PC (DOS)
Beilstein	pure components, mixtures (organic compounds)	experimental	physical, other properties	STN, Dialog	---
Gmelin	pure components, mixtures (mostly inorganic and organometallic compounds)	experimental	physical, other properties	STN	---
Detherm	pure components, mixtures	experimental, calculated	thermodynamic, phase equilibrium	STN, TDS	PC (UNIX), VAX, mainframe
DDB	pure components, mixtures	experimental	equilibrium data, thermodynamic properties	---	PC (DOS), VAX-VMS, others
JANAF	pure components (mostly inorganic compounds)	experimental	thermochemical	STN	PC (DOS)
NIST	pure components (mostly inorganic compounds)	experimental	thermodynamic properties	STN	PC (DOS), minicomputer

STN:

Direct dial: 614-447-3781

Internet: <http://www.fiz-karlsruhe.de/stn.html>**Dialog:**Internet: <http://www.dialog.com/>

Telephone Number: 1-800-334-2564

TDS:

Telephone Number: 214-245-0044

CHAPTER III

DATABASE MANAGEMENT SYSTEM SELECTION

A good database management system is the key to the development of a good database. Even though all relational DBMS's are based on the same relational data model, there are vast differences in the implementation of the model. Thus, it is crucial to select the right DBMS for a given database application, keeping in mind the resources available. Some of the salient features of current DBMS's that were considered during the selection of a DBMS suitable for the implementation of the GPA Database are outlined below.

There are a number of different criteria that can serve as the basis for the evaluation of a relational DBMS. Some of these criteria may seem trivial during the selection process, but the added efficiency gained by the proper selection of a DBMS far outweighs the resources expended on the actual evaluation, in the long run.

Current Trends

The current relational DBMS market is filled with many different kinds of DBMS's, each claiming to be better than the others. New versions of older DBMS's are brought out more often as the competition increases. To select a few from the many, one

has to evaluate the underlying technology for each of the more popular relational database management systems (RDBMS).

DBase[®] was one of the first databases to implement the relational data model. This has made it the most widely used RDBMS technology. Many versions of this RDBMS for various platforms exist. All have a similar underlying technology that was created in the 1970's under the name of Vulcan (McFadden, 1993). Today all the RDBMS that are based on this technology are called Xbase DBMS's. Together they form a majority among the currently installed RDBMS's.

Recently, a number of other RDBMS's have reached the market place. The latest is Microsoft Access[®]. Access has captured a large share of the database market due to its ease-of-use. This ease-of-use was easier to implement because, unlike the Xbase products, it did not have to conform to the 1970's technology and operating systems existent during the introduction of RDBMS's. FoxPro[®] is another Xbase product that incorporated ease-of-use, when compared to other Xbase products, but it still is not on par with the latest in RDBMS technology. The other major contender for the microcomputer RDBMS market is Paradox[®] by Borland. It was provided as an alternative to the existing Xbase products.

Based on the current trends in the RDBMS arena, Microsoft Access, Microsoft FoxPro, and Borland Paradox were chosen as the three candidates for the implementation of the GPA Database.

Performance Factors

The performance of a DBMS can be measured in terms of the features available and the processing efficiency. Normally one has to compromise between the two and arrive at an optimum configuration for the application being considered. If the data are difficult to manipulate, the developer would lean towards flexibility in design and operation. If the data were voluminous, processing efficiency would be assigned a high priority. These issues are examined below.

Data Types

The larger the number of data types a DBMS can support, the more flexible it becomes. If a RDBMS supports bitmap files for storing pictures, it would naturally be preferred by the developer of a database that requires extensive storage of pictures.

Microsoft Access had an edge over the other two RDBMS's as far as data types are concerned due to its ability to store nulls without wasting storage space. This was made possible by the recent introduction of the variant data type in Object BASIC[®], which is the underlying 4GL for Microsoft Access. The null data type is a useful option for a chemical engineering database due to the variation in formats for experimental data.

Speed

The speed of the database depends on the routines contained within the database engine. The database engine is the set of routines used for data manipulation. If the engine is fast, the DBMS can handle more voluminous data or larger tables. This factor becomes especially important for the types of DBMS's that we are considering, because they are intended for use on a microcomputer. If a DBMS is to be selected for its performance, then the size of the database, microprocessor speed, and database engine speed are the variables to be considered.

A complete performance analysis has been completed for these three databases by Li-Ron and Montgomery (Li-Ron, 1995). They found that Microsoft FoxPro was 148% faster than MS Access in query performance with Paradox coming in last.

Query Capabilities

The query capabilities of a DBMS are dependent on the different SQL constructs supported by the underlying database engine. Theoretically, the query capabilities of all RDBMS's would be the same if they all conformed exactly to the SQL standard set by the X3H2 Database Technical Committee (X3H2, 1986). However this is not the case; each RDBMS has its own adaptation of the SQL standard. These adaptations are normally subsets of the SQL-92 standard, and they implement only a limited number of SQL commands specified by the standard.

The other important factor to consider while discussing query capabilities is the query by example (QBE) interface of the RDBMS's. This interface is the deciding factor for a major portion of the application development time. If the QBE interface is more intuitive and user friendly, query development is much faster.

Microsoft Access has the best QBE of the three DBMS's under consideration. Access and FoxPro have an edge over Paradox with respect to SQL support. The level of SQL support provided by these DBMS's was estimated to be more than adequate for the GPA Database.

Data integrity

The three main data integrity concerns of a microcomputer RDBMS are referential integrity, domain validation, and storage media backup. Referential integrity is maintained by the Access database engine automatically. This relieves much of the effort on the part of the database developer and keeps the data secure from errors. The referential integrity capabilities of Microsoft Access also include cascading updates and cascading deletes.

Domain Validation

Domain validation is another data integrity feature that can save much programmer effort, if the database engine is capable of validating the data as it is entered

into the database. Microsoft Access has field and table level validation rules to automatically restrict the data being entered to the domain specified by the designer during the creation of the tables. Thus, all the code that would have normally been written to carry out these domain validations programatically is now implemented by the database engine. Domain specifications can include criteria such as the restriction of the value of a mole fraction field to between 0 and 1.

Safety

A microcomputer DBMS normally assumes that storage media backup of the database is going to be carried out uninterrupted and error free. Unfortunately, this is not always the case, and data can be lost during backup. Microsoft Access provides a facility called database repair that can solve problems created during backup. It is also important to have a built-in capability to backup data automatically. However, this feature is yet to be included in any of the current microcomputer DBMS's.

Hardware

Each DBMS requires a different amount of RAM and hard disk capacity. It is important to ensure that the minimum requirements of the DBMS are met by the slowest computer to ensure processing capability by all users. Normally, the hardware

requirements of a RDBMS are a function of the capabilities offered. Selection of a DBMS has to be balanced between software and hardware considerations.

Data Sharing

In the near future, data in a stand-alone database might be placed on a network that is accessible to many users. This is where the data sharing capabilities of a DBMS come into the picture. Facilities such as file, table, record or field level locking play a major role in determining the data sharing capabilities of a RDBMS. All of the RDBMS's considered here support some level of data sharing capabilities and are suitable for network use.

Security

When data sharing becomes necessary, security issues become prominent. Unauthorized changes to the data must be prevented by the security features inherent in the DBMS. This aspect of a database is normally commensurate with the level of data sharing supported by the DBMS and hence is not a very important issue in this case. However, one should be assured that this minimum level of security is actually offered by the DBMS before selecting it.

Interface

In general a database is not static. It is important to consider the transparency of the DBMS interface with other DBMS's and applications. Microsoft Access has excellent capabilities for data exchange with other applications and especially with ASCII based text files. This interface to text files was a crucial factor during initial setup of the GPA Database, because all the data in the original database had been stored as ASCII based flat files.

After considering all the above factors with respect to the existing flat file format GPA Database (Daubert, 1993), it was decided that Microsoft Access would be the DBMS that would best suit the requirements of a thermodynamic properties database.

CHAPTER IV

DESIGN PROCEDURE

There are numerous design procedures for database design and development (Fleming, 1989; Jackson, 1988; McFadden, 1993). Unfortunately most are aimed towards modeling an entire organization. These design procedures are effective for modeling an entity such as a business organization that has certain predefined functions, and the database maps these functions and carries them out in an efficient and error-free manner.

The concepts involved in designing a thermodynamic properties database are radically different, because the entity being modeled is a collection of state information pertaining to a given sample. Thus, the existing design procedures do not adequately address the needs of a chemical engineering database designer.

In this chapter, the outline of a generalized design procedure for a technical database is presented. The relational implementation of the vapor liquid equilibrium data contained in the GPA Database is used as a framework for this process. The difficulties encountered during the relational implementation of this flat-file database are used as a source of refinement for the generalized design procedure given at the end of the chapter. The advantages gained by the relational implementation are highlighted at each stage of the design for the vapor liquid equilibrium data.

Any relational database development procedure involves the following major steps:

1. Data assessment
2. Entity-Relationship Modeling
3. Logical Design
4. Physical Design
5. Application development

Data Assessment

During the development of a database, it is common to encounter an existing database in the form of historical data. These data could be hard copies stored in a filing cabinet, text files stored electronically in an ASCII based flat-file format, or even earlier versions of the database developed using another data model. This historical data may have to be incorporated into the database being developed. The steps preceding the transfer of historical data from one database to another are categorized under data assessment. They frequently involve data analysis, format analysis, error checking, and the actual data transfer.

Data Analysis

The GPA Database is a compilation of thermodynamic data reported in technical publications and in GPA Research Reports. The data were compiled at The Pennsylvania State University (Daubert, 1993) between 1983 and 1993 and contain six basic data types:

- Vapor-liquid equilibrium (VLE)
- Vapor-liquid-liquid and vapor-liquid-solid equilibrium
- Phase boundary (dew point and bubble point)
- Hydrate formation
- Enthalpy departure (pure components and mixtures)
- Enthalpy of solution

The data included in the database were based on the needs of the gas processing industry and other related operations.

Format Analysis

Depending on the type of information required to be stored, the format may vary between different thermodynamic data types. The basic unit of this data base is a data point. A data point is a record of state information -- temperature, pressure, etc. -- pertaining to a pure component or a mixture. In addition to the basic state information, each record also includes the name of the compound(s) and the data source.

The data format adopted by Daubert (1993) for storing the data was an ASCII based flat-file format. Due to the limitations existing at the time of conception of the database, the maximum record length of the flat-files was set at 80. This limitation resulted in loss of data integrity, because an individual record or a data point had to be split between multiple lines. Each data attribute, or field for the data point, was allotted a fixed width and a fixed starting column in the file.

A sample three-component VLE data file is given in Appendix A. The format is as specified in the original reference (Daubert, 1993). As noted, each data point had to be split across multiple lines due to the inherent limit of 80 characters per line, thus leading to loss of data integrity.

Error checking

Checking the existing data for errors is necessary because there is no information on data integrity prior to the transfer to the new database. The extent to which it is possible to accomplish this depends on the capabilities of the current database management system. In the case of the GPA Database, there was no database management system available to manipulate the existing data. Thus the decision was made to first transfer the data to the new DBMS (MS Access) before carrying out error checking.

After transfer of data to MS Access, validation checks were run to test for the consistency of the composition measurements. Data points where the sum of the liquid,

vapor or feed mole fractions deviated by more than 2% from unity were flagged as outliers and cross checked with the original reference for correctness. Referential integrity constraints were used to check for correct cross referencing of data such as reference numbers and component ID's. The concept of referential integrity is explained during the logical and physical design steps discussed later.

Comprehensive error checking for typographic errors on a point by point basis was recommended to be carried out after the database structure had been finalized. This type of error checking procedure must be done manually.

Data Transfer

Data transfer from the earlier flat-file version of the GPA Database to the latest MS Access version involved the use of data formatting routines written in FORTRAN (Appendix B). The FORTRAN code reads data from the input file containing VLE data (Appendix A) and writes the data in a format suitable for MS Access (Appendix C). This procedure was necessary because MS Access requires each record to be on a single line. Thus, multiple lines from the flat-file version were converted to a single line per record format (Appendix C) and then imported into MS Access.

MS Access can recognize ASCII based flat-file data provided the format specifications have been incorporated into the database. Format specifications involve the width of each field and the starting column number. Using these specifications, the tables allocated for the data are populated during the import procedure.

Entity-Relationship Model

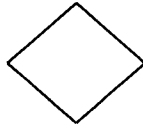
The Entity-Relationship model (E-R model) was first introduced in a key article by Chen (Chen, 1976) in which he described the main constructs of the E-R model -- entities and relationships -- and their associated attributes. It is the most popular conceptual data modeling technique in use today. This popularity can be attributed to ease of use, emulation of the real world concepts of entities and relationships, and easy availability of CASE tools for E-R model implementation. The E-R model is used to represent the structure of the database, independently of the DBMS used to implement it. This is called the conceptual model of the database. The E-R model can be used as a framework for refining the database structure by successive modifications of the entities and relationships contained in the database.

An E-R model has a standard nomenclature for representing entities and relationships as shown in Figure 8 (McFadden, 1993). The concepts in the figure that have not been discussed are multivalued attributes, gerund, relationship degree, and ISA. These concepts will be clarified during the discussion on the E-R model for the GPA Database.

Basic Symbols



Entity



Relationship



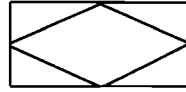
Attribute



Primary
Key

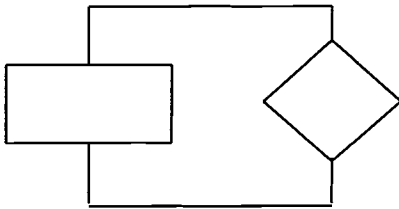


Multivalued
Attribute

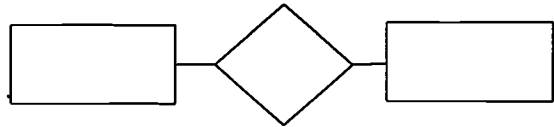


Gerund

Relationship Degree

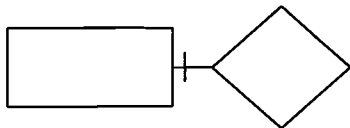


Unary

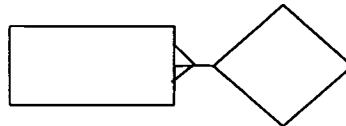


Binary

Relationship Cardinality



1 Cardinality



Many Cardinality

Class-Subclass Relationship



Figure 8 E-R Diagram Nomenclature

The first step in creating an E-R model is to list the entities that can be visualized for the application to be modeled. The next step is to plot them as part of an E-R diagram and to map all relationships that might exist between the various entities. Finally the E-R diagram is refined by a process called pseudo-normalization to remove any unwanted entities or relationships and to add any that might have been omitted during the preliminary analysis. The resulting E-R model can then be used as a framework during normalization. This process is explained in more detail with the GPA Database E-R model for VLE data as a framework.

Preliminary E-R Model

The basic unit of storage for the GPA Database is a data point, as mentioned earlier. Some of the other entities that exist in the GPA Database are “component” and “reference.” Thus the entity “data point” would have temperature, pressure, components and reference number as some of its attributes. Similarly the entity “component” would have component ID, component name, and molecular weight as some of its attributes. A condensed version of the preliminary E-R diagram for the GPA Database is given in Figure 9.

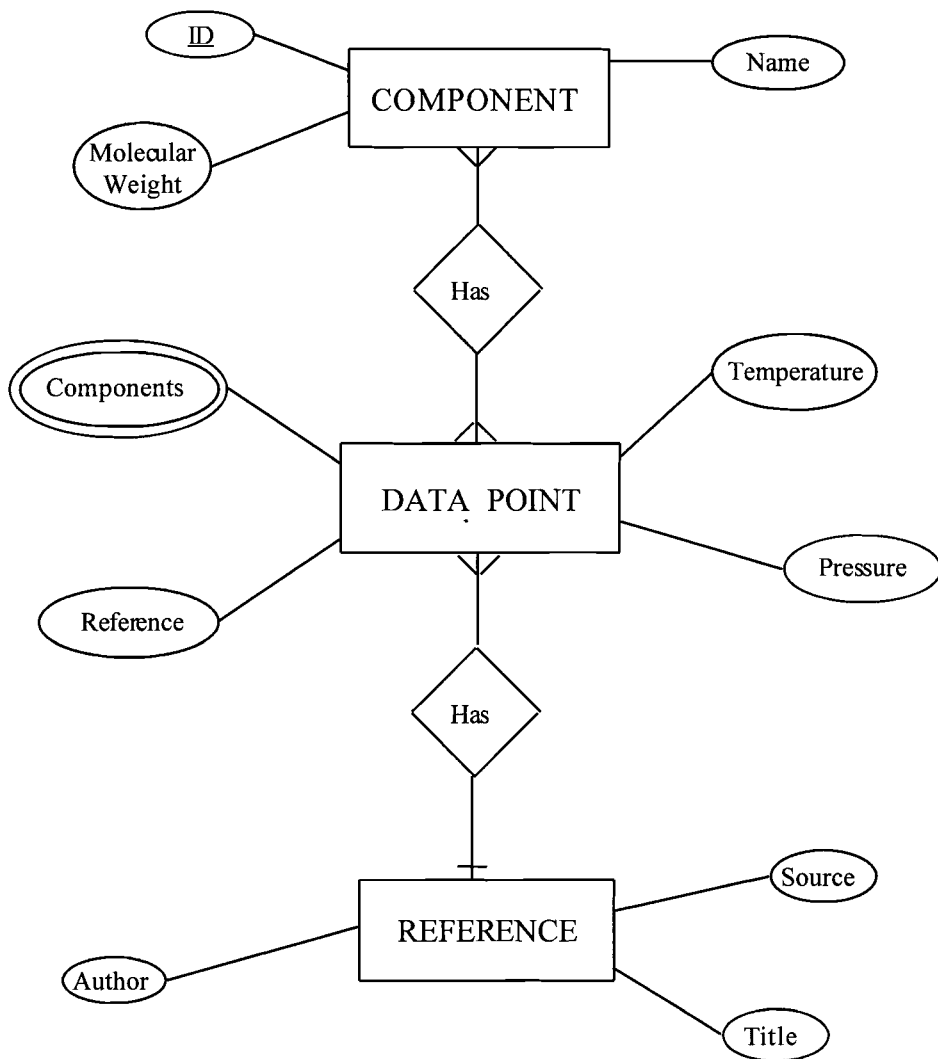


Figure 9 Preliminary E-R Model for the GPA Database

Pseudo Normalized E-R Model

Pseudo normalization is the process of removing obvious non-relational constructs from the preliminary E-R diagram. These non relational constructs could be in the form of multivalued attributes, gerunds, or class-subclass relationships.

In the GPA Database, it was thought pertinent to introduce another entity called “Data Set” to better model the data. A data set is a collection of data points that have certain attributes in common. This can be illustrated in terms of VLE data by saying that all data points of a given data type that have the same components and data source are considered part of the same data set. For example, all binary VLE data points that have “methane” and “ethane” as the two components and that were obtained from the same reference are considered part of the same data set in general. This rule can be made stricter by including more conditions. Additional conditions for classifying data sets might include raw or smoothed, data source, presence of feed mole fractions, etc. The relationship between the entities “Data Set” and “Data Point” will be one-to-many, i.e., each data set will have one or more associated data points, but each data point can be associated with only one data set.

The inherent advantage of this kind of implementation is that all the information common to a data set is stored only once, together with the record of the data set. It is not necessary to duplicate this information for each data point associated with a data set. Thus, if some information relevant to a data set were to be updated at a later time, changes would be required only for the one set where this information has been recorded

and not at every data point. This removal of redundancy solves update anomalies and saves storage space. Another advantage associated with this two layered structure is that the search for a given set of data can be carried out on a smaller table containing data set information instead of a larger table containing all data points. This increases the speed of searches by a significant factor. This two layered implementation for the GPA Database is shown in Figure 10.

The entity “Data Point”, if left unchanged, cannot adequately model all the different data types existing in the GPA Database, because each data type has certain attributes that are relevant only to that data type. For example, the field reserved for storing the enthalpy content of a mixture is relevant only for enthalpy data. Storage space would be wasted by including this field for the VLE data type. Thus, the entity “Data Point” can be linked to the six data types through an ISA relationship. An ISA relationship is a class-subclass relationship where a subclass *is a* type of superclass. In this context, the entity “VLE” *is a* data point. Similarly, the entity “Enthalpy” *is a* data point. The advantage of modeling the data in this manner is that all the common attributes are stored along with the superclass, and the remaining attributes are stored along with their pertinent subclasses. An example of how this is implemented is shown in Figure 11.

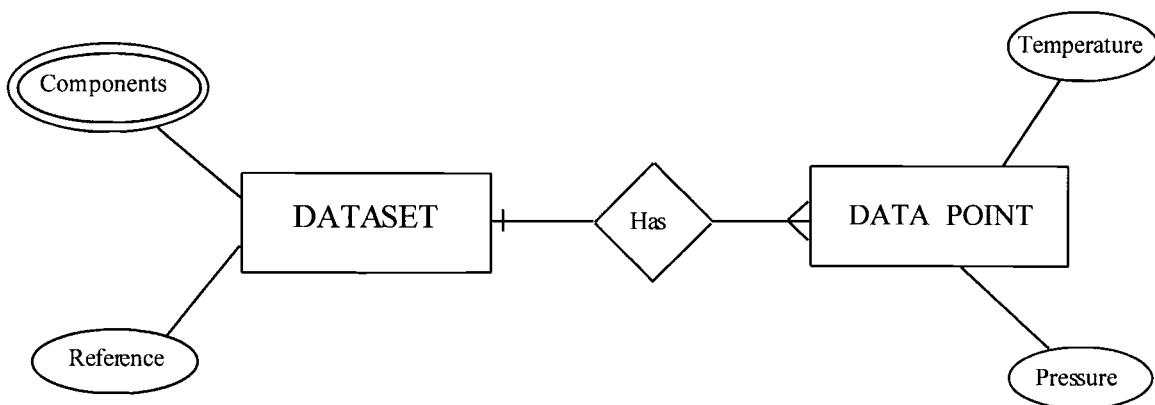


Figure 10 A One-to-Many Relationship in the GPA Database

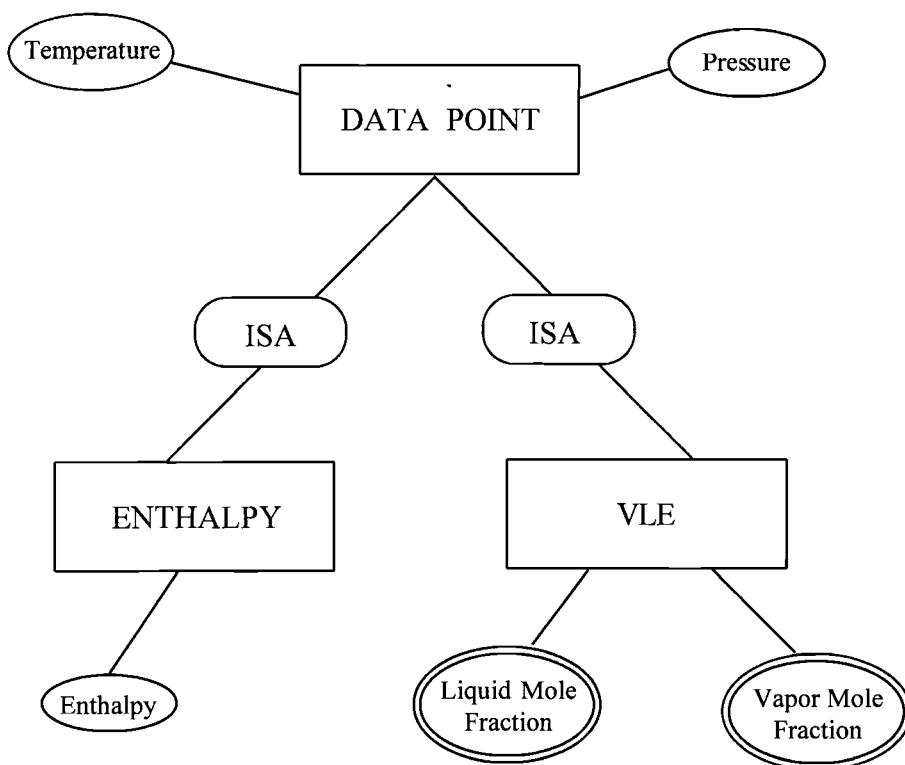


Figure 11 ISA Relationships in the GPA Database

Multivalued attributes require multiple values of the same type for a given entity. For the entity VLE, liquid mole fraction is a multivalued attribute because it stores a value for each component in the mixture. For example a ternary VLE would require three (two if experimental mole fractions are assumed to add up to 1) values for liquid mole fraction. This implementation puts a limit on the flexibility of the database because a multivalued attribute can be repeated only a predetermined number of times. Thus if it was specified that the attribute liquid mole fraction can be repeated only 10 times, a data set containing 11 components could not be incorporated into this entity even though it logically belongs there. The solution to this problem is the creation of a new entity "Liquid Mole Fraction." The entity "VLE" will now have a one-to-many relationship with the entity "Liquid Mole Fraction" with no limit for the number of records on the many side. Thus, the entity VLE can now model a mixture containing any number of components. This type of implementation is shown in Figure 12.

The various changes discussed above constitute what is known as pseudo normalization. This process helps to remove some of the flagrant anomalies in a database structure. The end product of the E-R modeling stage is the pseudo-normalized E-R diagram shown in Figure 13. This diagram shows the complete pseudo-normalized database structure for the VLE data, and it can be used as a guideline for creating the logical database design for VLE data.

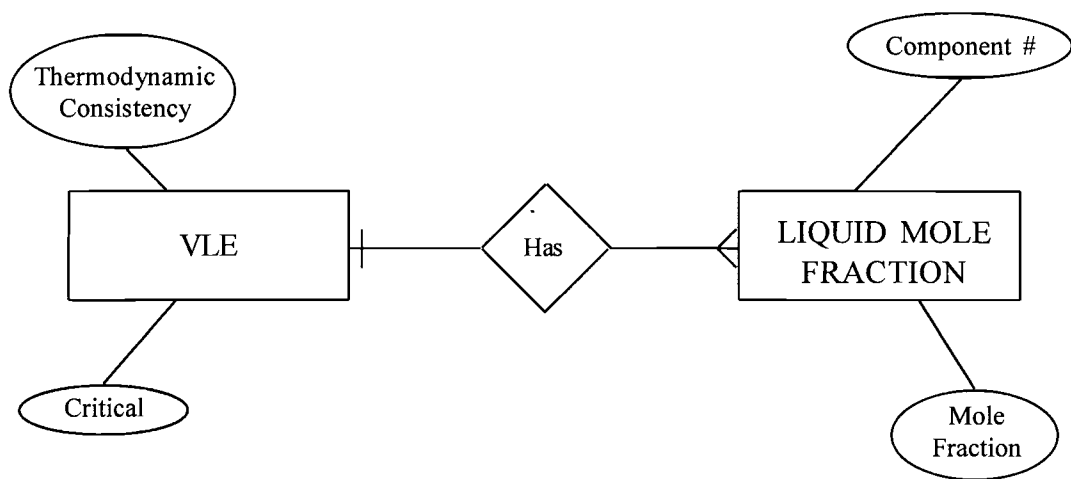


Figure 12 Representation of Multi-component VLE Data

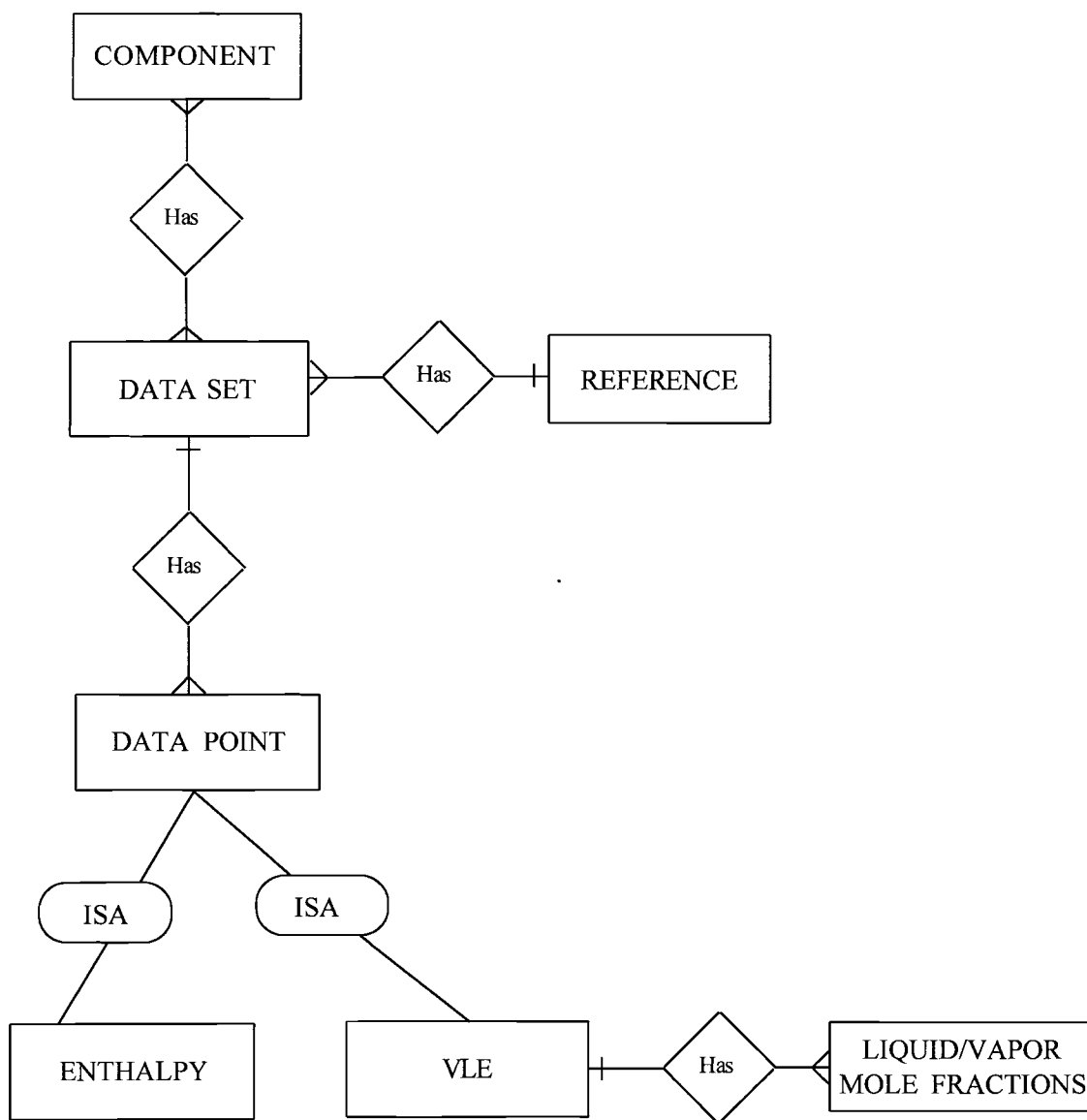


Figure 13 Pseudo-Normalized E-R Model for the GPA Database

Logical Database Design

Logical database design is the process of mapping the conceptual data model onto a logical data model, such as the relational data model. A logical data model is a design that satisfies the rules of a data model for a class of DBMS's. Thus, if we map the E-R diagram developed in the previous section onto the relational data model, we are creating a relational database.

The process of logical database design, when applied to the relational data model involves a two-step process of transformation and normalization. Transformation is more often used when creating a database which does not already have data existing in another format. Normalization, on the other hand, is always carried out regardless of whether the database is being created from a previous database or from new data. The tables created from flat-file data, or by transforming an E-R diagram into a logical data model, are not guaranteed to be in a normalized form and must be taken through each step of the normalization process. Each of these steps is discussed below in detail using the VLE data in the GPA Database as a framework. In actuality the GPA Database was created from an existing flat-file database, and the process of transformation was not used. However, the pseudo-normalized E-R diagram shown in Figure 13 and the tables resulting from its transformation were used as a guide during the normalization process. The final logical data model obtained after the process of normalization is also represented as an E-R Diagram and compared with the initial pseudo-normalized E-R model to better understand the two processes of transformation and normalization.

Transformation

Transformation of an E-R diagram into a relational data model involves the representation of entities and relationships, in terms of tables and relationships between tables respectively. The table is the basic concept of the logical design (analogous to an entity being the basic concept of an E-R model). Thus, the relational data model must be fully definable in terms of tables and relationships between tables. This can be best accomplished by first representing every entity as a table, and then representing the relationships between entities as a connection between two relational tables.

Entities

When an entity from an E-R diagram is transformed into a relational table, each attribute of the entity being transformed is converted to a data field in the form of a column heading. The table is named after the entity and assigned a primary key. If a suitable candidate key does not exist, then a new attribute is created solely to provide unique identification and serves as the primary key for the table.

Taking the entity “Component” from Figure 9 as an example, the corresponding relational table is shown below as Table 2. The primary key for this table is present in the form of Component ID, and hence it is not necessary to create a new primary key.

Table 2

Component -- Tabular Representation of the Entity “Component”

Component ID	Component	Formula	Molecular Weight
1	METHANE	CH4	16.043
2	ETHANE	C2H6	30.07
3	PROPANE	C3H8	44.097
4	ISOBUTANE	C4H10	58.123
5	N-BUTANE	C4H10	58.123
6	ISOPENTANE	C5H12	72.15
7	N-PENTANE	C5H12	72.15

Relationships

Representation of relationships between tables depends upon the cardinality of the relationship as defined in the E-R diagram. A simple one-to-one relationship can be implemented by comparing the primary keys of the two related tables. An ISA relationship is a one-to-one relationship. Thus the ISA relationship shown in Figure 11 can be represented as Tables 3 and 4. The one-to-one relationship here includes all the records from tables “Data Point” and “VLE” that have the same primary key value. The primary key here is “Data Point ID” for both the tables.

Table 3

Data Point -- Data Point Records

Data Point ID	Temperature (F)	Pressure (psia)
1	14.5	1212
2	13	1012
3	2	875
4	1	800
5	-10.4	660
6	-24	500
7	-46.9	310

Table 4

VLE -- Data Point Attributes Specific to VLE Data

Data Point ID	Thermodynamic Consistency	Critical Key
1	T	CR
2	T	CR
3	T	CR
4	T	
5	T	
6	T	
7	T	

A one-to-many relationship is represented as a link between the primary key of one table and the foreign key of another table. A foreign key is a field that is added to the table on the many side to establish a link with the table on the one side. The one-to-many relationship shown in Figure 10 can be represented by Tables 5 and 6. Here “Data Set ID” is the primary key for the table “Data Set.” For the table “Revised Data Point” the two fields “Data Point ID” and “Data Set ID” together form a compound primary key. The field “Data Set ID” is the foreign key in table “Revised Data Point.” Thus, each record from table “Data Set” is related to all the records from table “Revised Data Point” that have the same “Data Set ID.” This is an example of a one-to-many relationship in relational tables.

Table 5

Data Set -- Information Pertaining to Data Sets

Data Set ID	Reference ID	# of Components	# of Points	Comments
1	320	2	110	4-11-95 Checked by NL
2	324	2	18	4-13-95 checked by NL
3	325	2	173	4-13-95 checked by NL
4	325	2	118	4-13-95 checked by NL
5	325	2	114	8-14-95 checked by NL

Table 6

Revised Data Point -- Data Point Records with a Foreign Key Included

Data Point ID	Data Set ID	Temperature (F)	Pressure (psia)
1	2	14.5	1212
2	2	13	1012
3	2	2	875
4	2	1	800
5	2	-10.4	660
6	2	-24	500
7	2	-46.9	310
8	2	-73.1	190
9	2	-80	794
10	2	-80	764
11	2	-80	774
1	3	40	100.4
2	3	40	150.4
3	3	40	200.4
4	3	40	300.4
5	3	40	400.4
6	3	40	500.4
7	3	40	600
8	3	40	700
9	3	40	800

A relational table cannot be related to another relational table in a many-to-many relationship. To overcome this limitation a gerund can be created. A gerund is a many-to-many relationship that has been converted to an entity. In Figure 13 the entities “Data Set” and “Component” are related to each other in a many-to-many relationship. Each data set can contain many components, and each component can be included in many data sets. The relationship that connects these two tables must be converted to an entity, and subsequently, to a table. This table will have a many-to-one relationship with the tables “Data Set” and “Component.” The fields normally included in a gerund are the primary keys of the two related tables. These fields are the foreign keys in the gerund

and form the primary key when taken together with the “Component #” field. The “Component #” field is used to store the order in which the components present in a data set are referenced. The gerund that would result from the transformation of the many-to-many relationship between “Data Set” and “Component” is shown as Table 7. A gerund is normally named by combining the names of the two entities that it relates.

Table 7

Data Set-Component -- Representation of a M:N relationship

Data Set ID	Component #	Component ID
1	1	1
1	2	53
2	1	1
2	2	53
3	1	1
3	2	5
4	1	1
4	2	7
5	1	1
5	2	9
6	1	1
6	2	14
7	1	1
7	2	7

Normalization

Until now, the structuring of the database has been carried out in a more or less intuitive manner by means of a few general relational guidelines. This structuring must now be implemented using a formal set of rules to satisfy the requirements of a relational database. The procedure used to achieve this is normalization. It is the process of

reducing a complex data structure into simpler tables using a set of rules. This process is best implemented in a step-by-step manner as shown in Figure 14 (McFadden, 1993). The sequence of normal forms in this figure are the different states of a relational table. These states are determined by simple dependency rules applied to the fields contained in the table.

The VLE data from the GPA database are used to illustrate the process of normalization. The state of the data in the form of tables is shown at each stage. The disadvantages associated with the normalization process at each stage are discussed along with the benefits obtained by normalization.

First Normal Form

The first normal (1NF) form requires that a relational table contain no repeating groups or multivalued attributes. A repeating group is the relational table equivalent of multivalued attributes found with entities. The removal of repeating groups can be carried out by observation of the data in the table.

Let us examine the process of normalizing a flat-file to a first normal form table using VLE data as an example. Importing the flat-file in Appendix C into MS Access resulted in Table 8. This table contains an additional 7 data points taken from another binary VLE data set for illustrative purposes. The key to the names of the fields in this table is given in Appendix D.

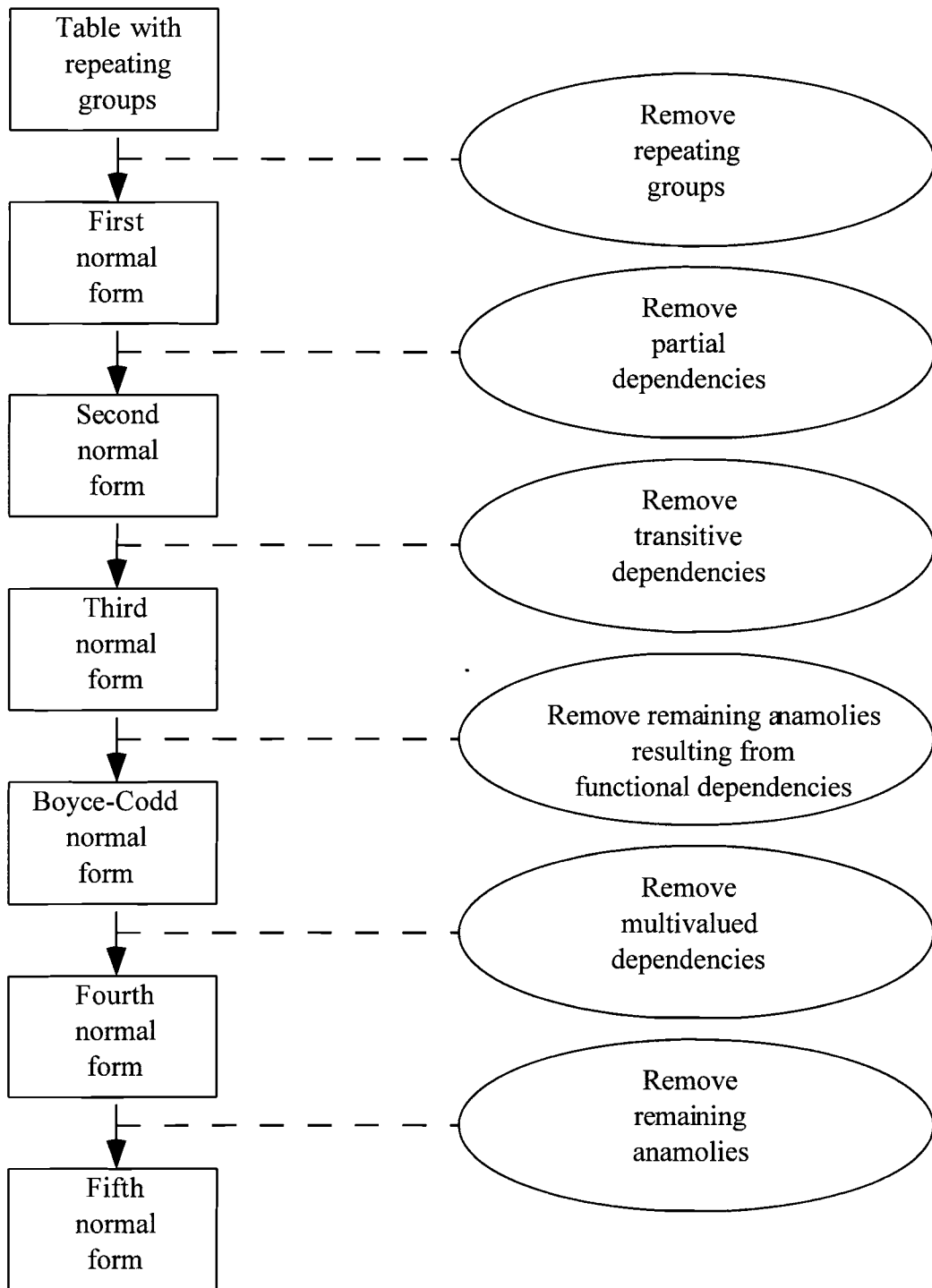


Figure 14 Steps in Normalization (Modified from McFadden, 1993)

Table 8

VLE Data Flat-file Import Table with Multivalued Attributes

T	P	X1	Y1	X2	Y2	X3	Y3	TC	DT	CR	DS	NC	C1	C2	C3	RN	DPN
-100	500	0.2942	0.4586	0.6925	0.0791	0.0133	0.4623		VL		G	3	1	2	5	43	1
-100	500	0.3255	0.5230	0.6611	0.0830	0.0134	0.3940		VL		G	3	1	2	5	43	2
-100	500	0.3859	0.6153	0.6021	0.0785	0.0120	0.3062		VL		G	3	1	2	5	43	3
-100	500	0.4325	0.6913	0.5574	0.0779	0.0101	0.2308		VL		G	3	1	2	5	43	4
-100	1000	0.0441	0.0324	0.9208	0.0351	0.0351	0.9325		VL		G	3	1	2	5	43	5
-100	1000	0.1153	0.1102	0.8447	0.0526	0.0400	0.8372		VL		G	3	1	2	5	43	6
100	59	0.0136	0.9864	0.9864	0.0136			T	VL			2	31	1		33	1
100	309	0.0583	0.9953	0.9417	0.0047			T	VL			2	31	1		33	2
100	680	0.1299	0.9960	0.8701	0.0040				VL			2	31	1		33	3
100	1003	0.1699	0.9966	0.8301	0.0034			T	VL			2	31	1		33	4
100	1323	0.2214	0.9960	0.7786	0.0040			T	VL			2	31	1		33	5
100	1680	0.2540	0.9952	0.7460	0.0048			T	VL			2	31	1		33	6
100	1993	0.2954	0.9946	0.7046	0.0054				VL	C		2	31	1		33	7

Table 8 has three multivalued attributes -- X, Y and C (liquid mole fractions, vapor mole fractions, and component ID's, respectively.) These multivalued attributes must be converted to a form that is more consistent with the tabular form of Table 9. This representation of multivalued attributes facilitates the accommodation of data sets containing any number of components without adding additional columns to the table. Unfortunately, Table 9 violates one of the basic constructs of a relational table in 1NF, viz., no cell in the table can have more than one value. Table 10 is created to remove this anomaly by eliminating repeating groups.

Table 9

VLE Data with repeating groups

DPN	T	P	X	Y	TC	DT	CR	DS	NC	C	RN
1	-100	500	0.2942 0.6925 0.0133	0.4586 0.0791 0.4623		VL		G	3	1 2 58	43
2	-100	500	0.3255 0.6611 0.0134	0.5230 0.0830 0.3940		VL		G	3	1 2 58	43
3	-100	500	0.3859 0.6021 0.0120	0.61530 0.0785 0.3062		VL		G	3	1 2 58	43
4	-100	500	0.4325 0.5574 0.0101	0.6913 0.0779 0.2308		VL		G	3	1 2 58	43
5	-100	1000	0.0441 0.9208 0.0351	0.0324 0.0351 0.9325		VL		G	3	1 2 58	43
6	-100	1000	0.1153 0.8447 0.0400	0.1102 0.0526 0.8372		VL		G	3	1 2 58	43
1	100	59	0.0136 0.9864	0.9864 0.0136	T	VL			2	31 1	33
2	100	309	0.0583 0.9417	0.9953 0.0047	T	VL			2	31 1	33
3	100	680	0.1299 0.8701	0.9960 0.0040		VL			2	31 1	33
4	100	1003	0.1699 0.8301	0.9966 0.0034	T	VL			2	31 1	33
5	100	1323	0.2214 0.7786	0.9960 0.0040	T	VL			2	31 1	33
6	100	1680	0.2540 0.7460	0.9952 0.0048	T	VL			2	31 1	33
7	100	1993	0.2954 0.7046	0.9946 0.0054		VL	C		2	31 1	33

Table 10

VLE -- VLE Data in 1NF

DSN	DPN	CN	T	P	X	Y	T	DT	CR	DS	NC	C	RN
1	1	1	-100	500	0.294	0.458		VL		G	3	1	43
1	1	2	-100	500	0.692	0.079		VL		G	3	2	43
1	1	3	-100	500	0.013	0.462		VL		G	3	58	43
1	2	1	-100	500	0.325	0.523		VL		G	3	1	43
1	2	2	-100	500	0.661	0.083		VL		G	3	2	43
1	2	3	-100	500	0.013	0.394		VL		G	3	58	43
1	3	1	-100	500	0.385	0.615		VL		G	3	1	43
1	3	2	-100	500	0.602	0.078		VL		G	3	2	43
1	3	3	-100	500	0.012	0.306		VL		G	3	58	43
1	4	1	-100	500	0.432	0.691		VL		G	3	1	43
1	4	2	-100	500	0.557	0.077		VL		G	3	2	43
1	4	3	-100	500	0.010	0.230		VL		G	3	58	43
1	5	1	-100	1000	0.044	0.032		VL		G	3	1	43
1	5	2	-100	1000	0.920	0.035		VL		G	3	2	43
1	5	3	-100	1000	0.035	0.932		VL		G	3	58	43
1	6	1	-100	1000	0.115	0.110		VL		G	3	1	43
1	6	2	-100	1000	0.844	0.052		VL		G	3	2	43
1	6	3	-100	1000	0.040	0.837		VL		G	3	58	43
2	1	1	100	59	0.013	0.986	T	VL			2	312	33
2	1	2	100	59	0.986	0.013	T	VL			2	1	33
2	2	1	100	309	0.058	0.995	T	VL			2	3121	33
2	2	2	100	309	0.941	0.004	T	VL			2	1	33
2	3	1	100	680	0.129	0.996		VL			2	3121	33
2	3	2	100	680	0.870	0.004		VL			2	1	33
2	4	1	100	1003	0.169	0.996	T	VL			2	31	33
2	4	2	100	1003	0.830	0.003	T	VL			2	1	33
2	5	1	100	1323	0.221	0.996	T	VL			2	31	33
2	5	2	100	1623	0.778	0.004	T	VL			2	1	33
2	6	1	100	1680	0.254	0.995	T	VL			2	31	33
2	6	2	100	1680	0.746	0.004	T	VL			2	1	33
2	7	1	100	1993	0.295	0.994		VL	C		2	312	33
2	7	2	100	1993	0.704	0.005		VL	C		2	1	33

Closer examination of Table 10 reveals the addition of two new columns: DSN (Data Set Number) and CN (Component Number). These new fields were added because none of the existing keys were suitable primary keys. Each row of Table 10 can now be uniquely be identified by specifying a DSN, DPN (Data Point Number), and a CN. These three fields, taken together, form the composite primary key for the table. DSN was introduced to distinguish between data sets and CN was introduced to distinguish between components within a data set.

Table 10 is now in first normal form and can be used to store VLE data. It is a relational database. The advantage associated with converting a flat-file to a 1NF table is that all the relational constructs contained in SQL can now be used with this 1NF table to query the data. Unfortunately, the following anomalies are associated with the table in 1NF:

- Insertion Anomaly : We cannot add a new Reference Number (RN) without the existence of at least one data set from that reference.
- Update Anomaly : If a Temperature value (T) were to be changed, the change would have to be made in multiple rows even if the change were only for one data point.

It is prudent to convert the table to second normal form due to the presence of these anomalies.

Second Normal Form

A relational table is in second normal form (2NF) if (a) it is in first normal form and, (b) if every nonkey attribute is functionally dependent on the whole primary key. Here the term nonkey attribute refers to any attribute that is not part of the primary key. Functional dependency is the case where one attribute will assume a value depending on the value of another attribute; the former is said to be functionally dependent on the latter. This means that no partial dependencies can exist for a table to be in 2NF. A partial dependency is the case where a nonkey attribute is functionally dependent on only part of the primary key.

Continuing with the example presented in Table 10, it can be seen that the table contains several partial dependencies. Let us examine them one at a time and take steps to remove these partial dependencies.

First, consider the fields that are functionally dependent only on DSN

- **DT** : Data type is fully functionally dependent on DSN, because a data set can contain only one data type -- in this case VL.
- **DS** : Data Source is also fully functionally dependent on DSN, because each data set can be obtained from only one source.
- **RN** : Reference Number is another field that is fully functionally dependent on DSN, because each data set can be taken from only one reference (when a data set was defined, it was decided that a different reference qualified a set of data points to belong to a different data set).

- **NC** : The number of components in a data set always remains the same. Thus, NC is fully functionally dependent on DSN.

To take Table 10 closer to 2NF, the partial dependencies listed above must be removed by decomposing the table into two smaller tables (Table 11 and Table 12). Note that during this process of decomposition, the fields in Table 12 do not have values repeated across rows as in Table 10. Removal of redundancy and the savings in storage space is one of the many advantages of normalization. Tables 11 and 12 can be related in a one-to-many relationship using DSN as the key.

We can now examine the fields that have partial dependencies with respect to data points:

- **T** : Temperature is unique for each data point but not for each component; T is fully functionally dependent on DPN and DSN. Note here that T is not functionally dependent only on DPN because DSN and DPN must be taken together to uniquely identify a data point. In other words DSN and DPN together form a composite primary key for a data point.
- **P** : Similar to T, P is fully functionally dependent on DSN and DPN.
- **TC** : Thermodynamic consistency tests can be carried out point by point and, therefore, require the field TC (Thermodynamic Consistency) to be functionally dependent on DSN and DPN.
- **CR** : A given data point can be identified for proximity to the critical point. Hence the field CR (Critical Region) is fully functionally dependent on DSN and DPN.

Table 11

VLE Data without DSN Partial Dependency

DSN	DPN	CN	T	P	X	Y	TC	CR	C
1	1	1	-100	500	0.294	0.458			1
1	1	2	-100	500	0.692	0.079			2
1	1	3	-100	500	0.013	0.462			58
1	2	1	-100	500	0.325	0.523			1
1	2	2	-100	500	0.661	0.083			2
1	2	3	-100	500	0.013	0.394			58
1	3	1	-100	500	0.385	0.615			1
1	3	2	-100	500	0.602	0.078			2
1	3	3	-100	500	0.012	0.306			58
1	4	1	-100	500	0.432	0.691			1
1	4	2	-100	500	0.557	0.077			2
1	4	3	-100	500	0.010	0.230			58
1	5	1	-100	1000	0.044	0.032			1
1	5	2	-100	1000	0.920	0.035			2
1	5	3	-100	1000	0.035	0.932			58
1	6	1	-100	1000	0.115	0.110			1
1	6	2	-100	1000	0.844	0.052			2
1	6	3	-100	1000	0.040	0.837			58
2	1	1	100	59	0.013	0.986	T		312
2	1	2	100	59	0.986	0.013	T		1
2	2	1	100	309	0.058	0.995	T		3121
2	2	2	100	309	0.941	0.004	T		1
2	3	1	100	680	0.129	0.996			3121
2	3	2	100	680	0.870	0.004			1
2	4	1	100	1003	0.169	0.996	T		31
2	4	2	100	1003	0.830	0.003	T		1
2	5	1	100	1323	0.221	0.996	T		31
2	5	2	100	1623	0.778	0.004	T		1
2	6	1	100	1680	0.254	0.995	T		31
2	6	2	100	1680	0.746	0.004	T		1
2	7	1	100	1993	0.295	0.994		C	312
2	7	2	100	1993	0.704	0.005		C	1

Table 12

DS_VLE -- Data Set Level VLE Data in 2NF.

DSN	DT	DS	NC	RN
1	VL	G	3	43
2	VL		2	33

The next step required for the normalization of Table 11 to 2NF is to decompose it into Tables 13 and 14 in order to remove the existing partial dependencies on DSN and DPN, taken together. This decomposition further reduces redundancy and saves storage space. Tables 13 and 14 can be linked using DSN and DPN as a composite key in a one-to-many relationship. Table 12 can be related to Tables 13 and 14 using the DSN key in one-to-many relationships.

Finally, let us check for partial dependencies based on a combination of the fields DSN and CN:

- **C** : Component ID is functionally dependent on DSN and CN, because the components remain the same for all the data points in a given data set.

To remove the partial dependency of C on DSN and CN, Table 13 is decomposed into Tables 15 and 16. As with the other decompositions, this normalization brings about a reduction in redundancy and a savings in storage space.

The removal of partial dependencies from a table in first normal form (Table 10) has resulted in a set of four tables in second normal form (Tables 12, 14, 15 and 16). The procedure used to achieve this normalization is summarized in Figure 15. These tables are related by several one-to-many relationships as shown in Figure 16. Also shown in Figure 16 are two more tables: Tables 17 and 18. These two tables contain information about the references and components, respectively, and are related as shown in Figure 16.

Table 13

VLE Data without DSN and DPN partial dependency

DSN	DPN	CN	X	Y	C
1	1	1	0.2942	0.4586	1
1	1	2	0.6925	0.0791	2
1	1	3	0.0133	0.4623	58
1	2	1	0.3255	0.5230	1
1	2	2	0.6611	0.0830	2
1	2	3	0.0134	0.3940	58
1	3	1	0.3859	0.615300	1
1	3	2	0.6021	0.0785	2
1	3	3	0.0120	0.3062	58
1	4	1	0.4325	0.6913	1
1	4	2	0.5574	0.0779	2
1	4	3	0.0101	0.2308	58
1	5	1	0.0441	0.0324	1
1	5	2	0.9208	0.0351	2
1	5	3	0.0351	0.9325	58
1	6	1	0.1153	0.1102	1
1	6	2	0.8447	0.0526	2
1	6	3	0.0400	0.8372	58
2	1	1	0.0136	0.9864	312
2	1	2	0.9864	0.0136	1
2	2	1	0.0583	0.9953	3121
2	2	2	0.9417	0.0047	1
2	3	1	0.1299	0.9960	3121
2	3	2	0.8701	0.0040	1
2	4	1	0.1699	0.9966	31
2	4	2	0.8301	0.0034	1
2	5	1	0.2214	0.9960	31
2	5	2	0.7786	0.0040	1
2	6	1	0.2540	0.9952	31
2	6	2	0.7460	0.0048	1
2	7	1	0.2954	0.9946	312
2	7	2	0.7046	0.0054	1

Table 14

DP_VLE -- Data Point Level VLE Data in 2NF

DSN	DPN	T	P	TC	CR
1	1	-100	500		
1	2	-100	500		
1	3	-100	500		
1	4	-100	500		
1	5	-100	1000		
1	6	-100	1000		
2	1	100	59	T	
2	2	100	309	T	
2	3	100	680		
2	4	100	1003	T	
2	5	100	1323	T	
2	6	100	1680	T	
2	7	100	1993		C

Table 15

DP_VLE_XY -- Data Point Level VLE Data for Mole Fractions in 2NF.

DSN	DPN	CN	X	Y
1	1	1	0.2942	0.4586
1	1	2	0.6925	0.0791
1	1	3	0.0133	0.4623
1	2	1	0.3255	0.5230
1	2	2	0.6611	0.0830
1	2	3	0.0134	0.3940
1	3	1	0.3859	0.615300
1	3	2	0.6021	0.0785
1	3	3	0.0120	0.3062
1	4	1	0.4325	0.6913
1	4	2	0.5574	0.0779
1	4	3	0.0101	0.2308
1	5	1	0.0441	0.0324
1	5	2	0.9208	0.0351
1	5	3	0.0351	0.9325
1	6	1	0.1153	0.1102
1	6	2	0.8447	0.0526
1	6	3	0.0400	0.8372
2	1	1	0.0136	0.9864
2	1	2	0.9864	0.0136
2	2	1	0.0583	0.9953
2	2	2	0.9417	0.0047
2	3	1	0.1299	0.9960
2	3	2	0.8701	0.0040
2	4	1	0.1699	0.9966
2	4	2	0.8301	0.0034
2	5	1	0.2214	0.9960
2	5	2	0.7786	0.0040
2	6	1	0.2540	0.9952
2	6	2	0.7460	0.0048
2	7	1	0.2954	0.9946
2	7	2	0.7046	0.0054

Table 16

DS_COMP -- Data Set Level Data for Component Information in 2NF.

DSN	CN	C
1	1	1
1	2	2
1	3	58
2	1	312
2	2	1

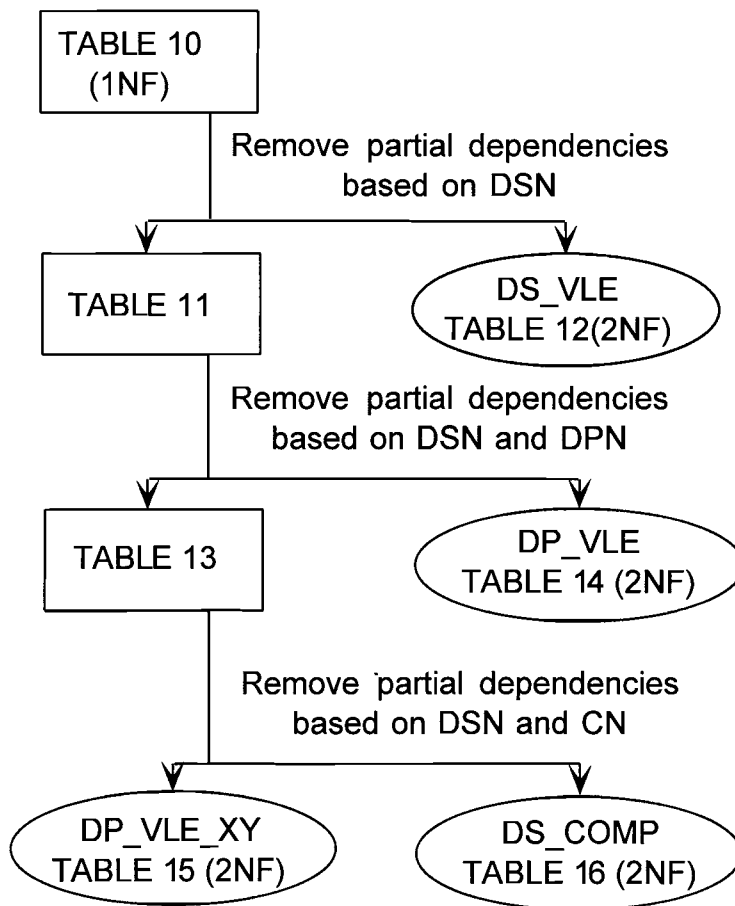


Figure 15 Normalization of Table 10 (VLE Data) from 1NF to 2NF.

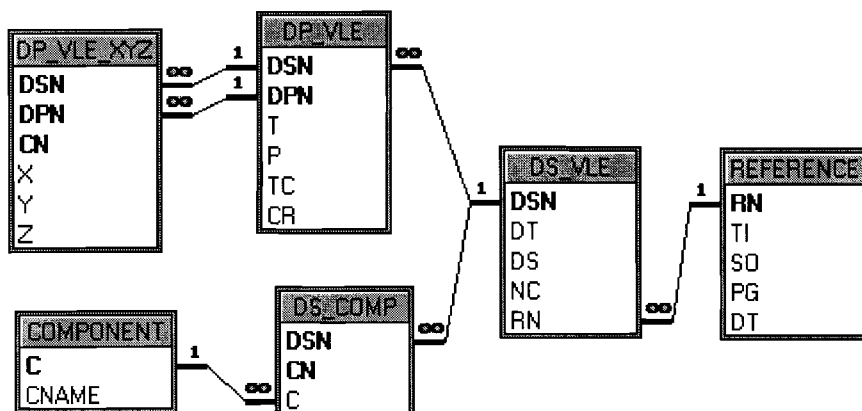


Figure 16 Relationships between VLE Data Tables in 2NF.

Table 17

REFERENCE -- Reference Information such as Title and Source.

RN	TI	SO	PG	DT
33	Solubility of Volatile Gases in Hydrocarbon	Ind. Eng. Chem. Fund.	3, 355	1964
37	Liquid-Vapor Phase Equilibrium in Mixtures of	Teor. Osn. Khim. Teknol.	3, (5) 766	1969
38	Vapor-Liquid Equilibrium of the Methane-	Cryogenics	13, (7) 405	1973
39	Nitrogen-Methane Vapor-Liquid Equilibria	Chem. Eng. Progr. Symp. Ser.	49, (6) 1	1953
40	Experimental Vapor-Liquid Equilibrium Data for	Chem. Eng. Progr. Symp. Ser.	63, (81) 10	1967
41	Vapor-Liquid Equilibrium Ratios for Four Binary	API Div. of Ref. Proceedings	45 (3) 62	1965
43	Low Temperature Vapor-Liquid Equilibria in	AIChE J.	5, (1) 46	1959
44	High-Pressure Rectification, n-Pentane-n-	Ind. Eng. Chem.	25, 728	1933
46	Vapor-Liquid Equilibria in Three Hydrogen-	Ind. Eng. Chem.	38, 389	1946

Table 18

COMPONENT -- Component Names.

C	CNAME
1	METHANE
2	ETHANE
3	PROPANE
4	ISOBUTANE
5	N-BUTANE
6	ISOPENTANE

The process of normalizing a 1NF table into two or more 2NF tables also removes anomalies (such as insertion, update, or deletion) that might have existed with a 1NF table. The normalization process in our example has removed the insertion and update anomalies associated with the 1NF table (Table 10). Unfortunately, there are still anomalies associated with tables in 2NF as shown below.

- Update Anomaly : If the reference number (RN) of a particular data set were changed, the corresponding data source (DS) would also need to be changed.
- Deletion Anomaly : If a particular data set were the only one that cross-referenced a given reference (RN), then the deletion of that data set would entail the loss of information on the data source (DS) of that reference (RN).

To solve these anomalies existing in the 2NF tables, we will convert these tables to third normal form as shown below.

Third Normal Form (3NF)

Relational theory states that a table is in third normal form if (a) it is in second normal form, and (b) no transitive dependencies exist within the table. A transitive dependency is one where one nonkey attribute is functionally dependent on another nonkey attribute. Consider table “Data Point” (Table 14). If there was a functional dependency between any of the four nonkey attributes “Temperature”, “Pressure”, “Thermodynamic Consistency”, or “Critical Region”; then the table would not be in third normal form.

On observation of the tables in 2NF (Tables 12,14,15,16,17, and 18), it is noted that Table 12 is not in 3NF, because of the transitive dependency of DS on RN. This transitive dependency arises because a given reference (RN) can belong to only one data source type (DS). This implies that once a RN is specified, a corresponding DS is implicitly specified. To remove this transitive dependency, DS was removed from the table DS_VLE (Table 12) and added to the table REFERENCE (Table 17). The modified DS_VLE is shown as Table 19, and the modified REFERENCE is shown as Table 20. The tables containing the VLE data of the GPA Database are shown as an E-R diagram in Figure 17. A detailed description of this E-R diagram is given in Figure 18.

Table 19

DS_VLE (3NF) -- Data Set Level VLE Data in 3NF.

DSN	DT	NC	RN
1	VL	3	43
2	VL	2	33

Table 20

REFERENCE (3NF) -- Modified Reference Information Table in 3NF

RN	TI	SO	PG	DT	DS
33	Solubility of Volatile Gases in Hydrocarbon	Ind. Eng. Chem. Fund.	3, 355	1964	
37	Liquid-Vapor Phase Equilibrium in Mixtures of	Teor. Osn. Khim. Teknol.	3, (5) 766	1969	
38	Vapor-Liquid Equilibrium of the Methane-	Cryogenics	13, (7) 405	1973	
39	Nitrogen-Methane Vapor-Liquid Equilibria	Chem. Eng. Progr. Symp.	49, (6) 1	1953	
40	Experimental Vapor-Liquid Equilibrium Data	Chem. Eng. Progr. Symp.	63, (81) 10	1967	
41	Vapor-Liquid Equilibrium Ratios for Four	API Div. of Ref.	45 (3) 62	1965	
43	Low Temperature Vapor-Liquid Equilibria in	AIChE J.	5, (1) 46	1959	G
44	High-Pressure Rectification, n-Pentane-n-	Ind. Eng. Chem.	25, 728	1933	
46	Vapor-Liquid Equilibria in Three Hydrogen-	Ind. Eng. Chem.	38, 389	1946	

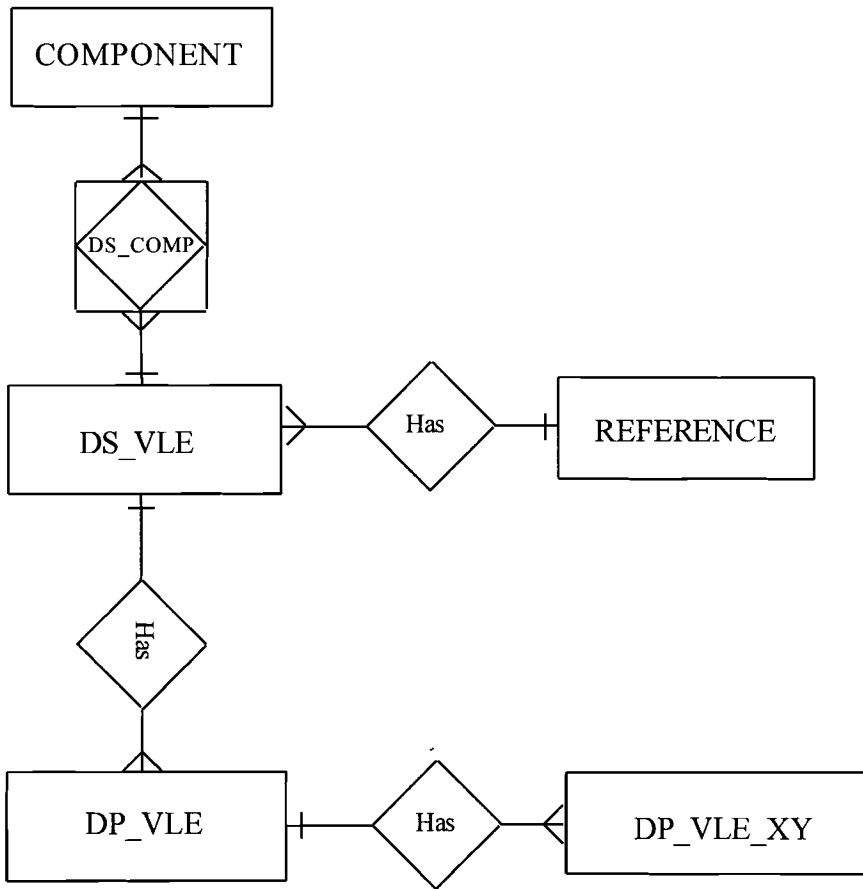


Figure 17. E-R Diagram of VLE Data in 3NF.

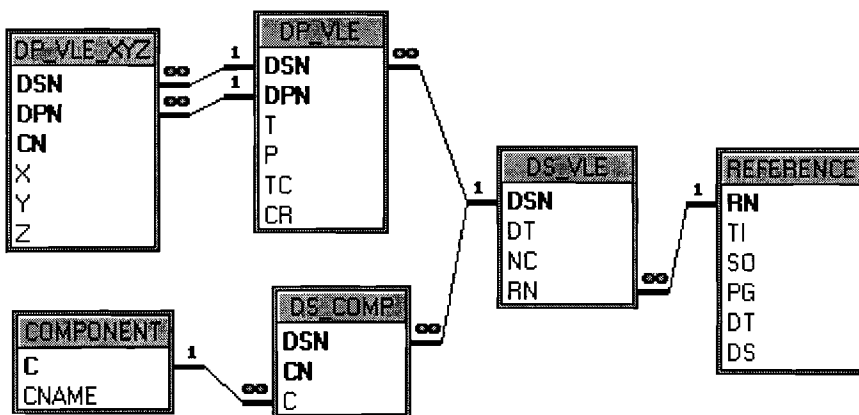


Figure 18 Relationships between VLE Data Tables in 3NF.

This normalization step also results in a removal of redundancy and a savings in storage space. The anomalies present in the tables in 2NF have been removed.

Most database designs are aimed at bringing the tables they contain to third normal form. The anomalies removed by bringing all the tables of a relational database to 3NF are sufficient for most data manipulation and integrity requirements. It is not common to take a database beyond 3NF, because all normalization typically involves a decrease in performance. This decrease in performance is evident during data retrieval. To retrieve data in their original format, all the tables that contain portions of the original data have to be logically reconstructed. Thus, the greater the number of simple tables that contain the original entity, the higher the complexity of data retrieval and data update operations.

Tables in third normal form still exhibit a few anomalies that occur due to updates, deletions, or insertions. Given below are examples of some of these anomalies.

- Insertion Anomaly : Adding a new component ID in the DS_COMP table (Table 16) without a corresponding entry existing in the table COMPONENT (Table 18) would result in a referential integrity violation.
- Update Anomaly : Changing the component ID (C) of one of the components in the COMPONENT table (Table 18) would entail changing values in multiple rows in DS_COMP (Table 16).
- Deletion Anomaly : If a reference were to be deleted from the table REFERENCE (Table 17), the entry in RN in the table “DS_VLE”(Table 12) would be pointing to an empty record.

Fortunately, the level of sophistication of current RDBMS's includes validation rules to prevent the anomalies listed above. The MS Access database engine has the capability to implement referential integrity constraints. This means that the insertion anomaly discussed above will not be permitted, because an internal validation rule applied by the DBMS would be violated. In this case, the user would have to create a corresponding entry on the one side of the 1:M relationship before adding a record on the many side of the relationship.

To handle the update anomaly discussed above, the MS Access database engine provides "cascading updates." If a value in a field that is used for connecting to another table in a 1:1 or a 1:M relationship is modified, then a cascading update will automatically modify the corresponding value on the related table.

The deletion anomaly is a problem similar to the update anomaly in that a change on the one side of a 1:M relationship can violate referential integrity rules. The Jet[®] Database engine handles deletion anomalies by deleting all corresponding records on the many side. These concepts of referential integrity, cascading updates, and cascading deletes are discussed in more detail in the section on relationships included under physical database design later in this chapter.

The database designer has the option of implementing all of these functions provided by MS Access (referential integrity, cascading updates, and cascading deletes) during physical database design. For the GPA Database, it was decided to stop normalization of the tables at 3NF and to allow the DBMS to handle the update, deletion, and insertion anomalies.

The condensed E-R diagram of the structure of the GPA Database is shown in Figure 19. Note the differences between this E-R diagram and the pseudo-normalized E-R diagram shown in Figure 13. The most obvious difference is that the entity “Data Point” is absent in Figure 19. This is because the disadvantages associated with the creation of this entity outweighed the advantages. The advantages here would have been that if a new attribute (or field) were to be added to the entity “Data Point”, it would have to be added only once to the table that would store data point information common to all data types. On the other hand, the disadvantage of this kind of implementation is that the data retrieval time would be very large, because the data point table would be very large. Since performance is an important consideration, and since the absence of a data point table did not violate any normalization rules; it was decided not to create a data point table.

Data Point was the entity used for connecting the various data types present in the GPA Database in the original pseudo-normalized E-R diagram. With the removal of this entity from the database structure, it was only natural to link the various data types together at the data set level using ISA relationships with a common DATA_SET table (Table 21). The DATA_SET table consists of the attributes common to all data types at the data set level. This new structure conformed with the normalization rules for 3NF as shown above and was fixed as the final logical data model for the GPA Database. The final logical data model for the VLE data tables is given in Figure 20. The logical data model for the complete GPA Database is given in Appendix E as tables and the relationships between them.

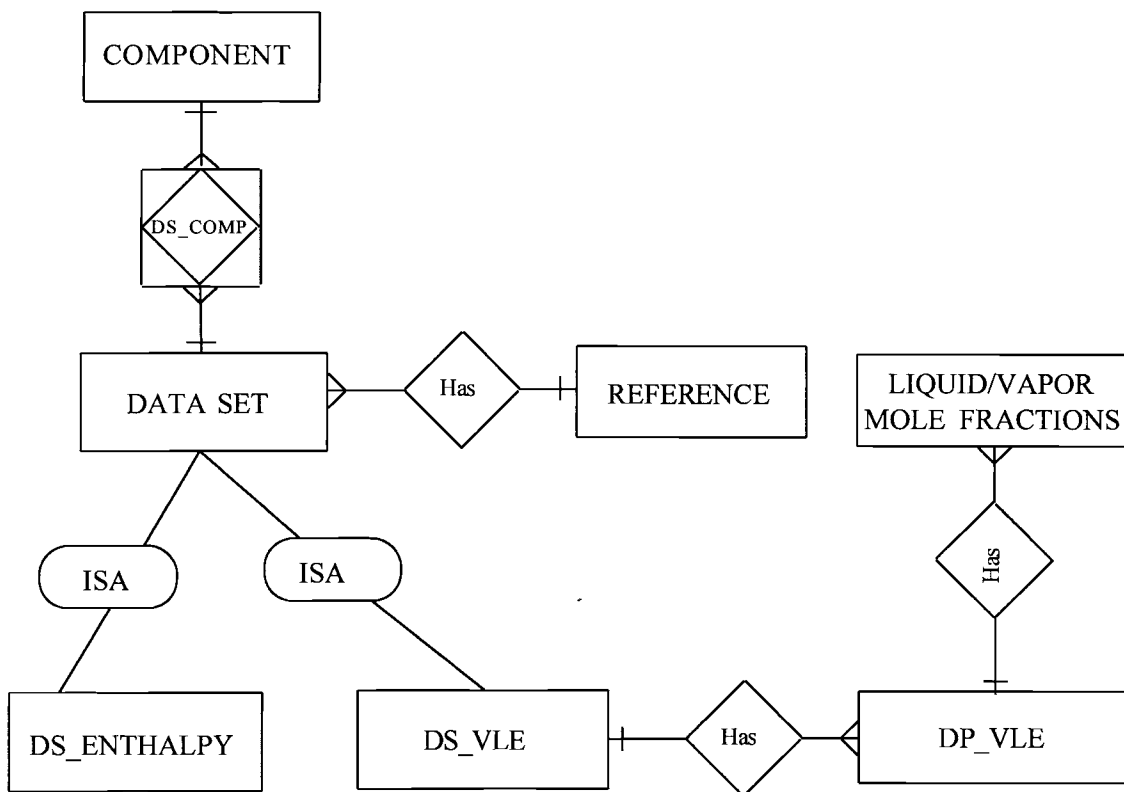


Figure 19. E-R Diagram of GPA Database in 3NF.

Table 21

DATA SET -- Common Data Set Level Information for all Data Types

DSN	# of Comp	Reference #	DT	# of Points	Unit Codes	Min. T	Max. T	Min. P	Max. P	Comm
1	2	320	1	110	AAA	-184	-65	84.4	939	4-11-95
2	3	324	2	18	ACA	-80	14.5	190	1212	4-13-95
3	2	325	5	173	AAE	-10.01	40	19.8	1865	4-13-95

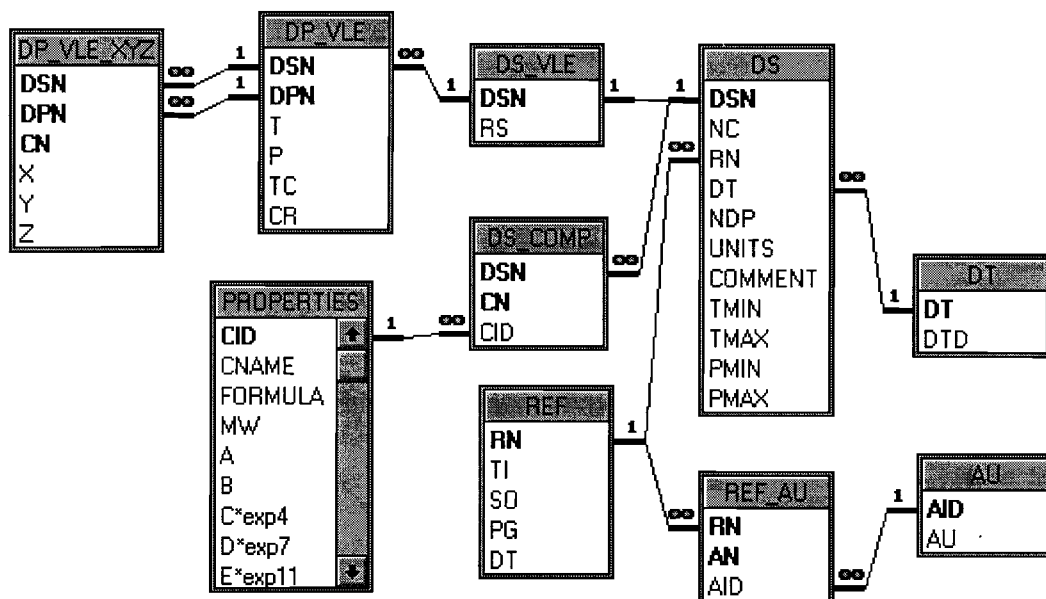


Figure 20 Logical Data Model for VLE Data.

Physical Database Design

Physical database design involves the creation of database tables using the selected DBMS. The logical data model of the GPA Database given in Appendix E was translated into the physical data structure given in Appendix F during this stage of the design process. This level of structure was physically implemented using queries to modify the original flat-file data to fit into the relational tables created for the database. Action queries, implemented using SQL, are most often used for this kind of data manipulation. Any query that modifies, deletes, or creates new data is called an action query.

The main consideration during this stage of the design process is to optimize performance by means of a good physical database design. Some of the issues involved in performance optimization are storage space, data integrity, and data retrieval time.

Most physical database design is DBMS specific. However there are certain aspects to physical design that have to be considered, irrespective of the DBMS being used to create the database. They involve data types, domains, indexes, and relationships. Each of these topics is discussed below with respect to performance optimization for the implementation of the GPA Database using MS Access.

Data Types

Each DBMS offers its own set of data types depending on the type of databases for which it was developed. Thus a DBMS catering mainly to multimedia databases might have picture and sound files as native data types. MS Access provides a wide range of data types to fit the needs of many types of databases. Prudent selection of data types can lead to considerable savings in storage space requirements, because the saving in space per field is multiplied by the number of records in the database.

The basic storage requirement of a technical database is floating point numbers. All contemporary DBMS's offer floating point data types either as single precision or double precision, or both. Single precision normally implies seven significant digits and an exponent varying from 38 to 45, depending on whether the number is positive or negative. This uses 32 bits. Double precision, as the name implies, takes 64 bits and

provides 15 significant digits with an exponent varying from 308 to 324. These capabilities are sufficient for most technical storage requirements. All non-integer numerical values in the GPA Database were stored as single precision. This helps to save space when none of the original numeric data are double precision. Most thermodynamic data are not reported as double precision, because the instrumentation used for collecting the data does not reach that level of precision. Also, the margin of experimental error present in most data makes it unnecessary to store the numerical values as double precision, even if they were generated as double precision.

Another important data type that is required to be supported by many technical databases is the ability to store a drawing. Most DBMS's today do not directly provide a "drawing" data type, but do provide a work around. This work around is implemented using object linking and embedding (OLE). Using OLE, any drawing application that is an OLE server can embed its drawings in the DBMS, provided the DBMS is capable of acting as an OLE client. Fortunately, most DBMS's today do support OLE client capabilities and are capable of supporting the demands of technical databases that might need to store specialized drawings, such as equipment drawings or process flowsheets. The GPA Database did not have drawings as part of its data and did not require OLE to embed external drawings.

Domains

The data type of a field provides a broad domain range for the attribute that it represents. A single precision data type for a field limits the allowable values for that field to real numbers within the range supported by single precision. However, this inherent domain range of the data type is normally too broad for most fields. To overcome this, the database designer can incorporate domain validation rules into the database. Provision for implementing domain validation is sometimes built into the DBMS; this is the case with MS Access. Validation can either be at the field level or at the table level.

Field level validation rules are simple rules designed to limit the range of values that a field may contain. For example, a field that is to store mole fractions could use a field level validation rule that allows only numbers between 0 and 1. This helps to ensure data integrity by decreasing the chances of typographic data entry errors. These validation rules are constructed using the comparison syntax used for SQL statements.

Table level validation rules are implemented at the level of the table. Thus a table that is designed for VLE data might have a validation rule to ensure that the liquid or vapor mole fractions sum to unity. These table level validation rules can also be made more thorough. For example, the data being entered might be compared with values predicted by a model. This can be implemented by incorporating the model equations into the validation rules.

For the GPA Database field level validation rules were implemented for all the fields in the database. In addition to the numerical validations, validation rules were also implemented for fields containing text. If a field or table level rule is violated, a text message is automatically displayed which informs the user of the permissible range of values for that field. Validation rules can reduce the speed of the database due to the necessity to check all values being entered. Since data integrity is a higher priority than performance, validation rules need to be implemented.

Indexes

Indexes are a means of speeding up data retrieval. This increase in speed is achieved by storing a condensed version of the indexed field as the index itself. Use of indexes introduces extra processing overhead during updates. Each time the indexed field has to be updated, the index also must be updated. Indexes take up extra storage space, and it is imperative to employ a prudent strategy for index selection.

For the GPA Database, indexes were employed on all the fields that were to be used as part of the overall search strategy. Update speed was not an important factor because this database is intended primarily for data retrieval and not for data entry. Retrieval speed was improved at the cost of increased storage space because the current size of the database was not a limiting factor.

Relationships

The implementation of relationships is the last step in physical database design. All the tables that have been created are linked to other related tables. The important consideration during this stage is referential integrity. MS Access provides built-in support for referential integrity. The implementation of referential integrity is accomplished using validation rules supported by the MS Access database engine. One of the other issues that must be addressed during the implementation of referential integrity is cascading updates and cascading deletes. During a cascading update, if the related field on the one side of a 1:1 or 1:N relationship is updated, the corresponding field in the related table is updated by the database engine. Similarly, if a record on the one side of a 1:1 or 1:N relationship is deleted, the corresponding records in the related table are automatically deleted. Cascading updates and cascading deletes are a means of ensuring referential integrity as a result of application level validation rules. In the GPA Database, all the relationships shown in Appendix F are implemented with referential integrity, cascading updates, and cascading deletes.

Application Development

This part of the database design procedure involves the addition of a graphical user interface (GUI) as a layer over the existing database. This makes it easy for a novice user to query and manipulate the underlying data without having to understand all the

details of the database structure. The two basic aims of the GUI are (1) to allow the user to query the database for specific data, and (2) to add data to the database. The user interface can also provide for data manipulation features that allow the user to manipulate the queried data into a form suitable for further processing. The application development stage also extends the use of the database by making it independent of the DBMS that was used to create it. Each of these steps is discussed below in the order in which they were applied to creating the stand-alone version of the GPA Database.

Data Retrieval

Structured Query Language (SQL) allows a user to query data contained in a database in an almost limitless number of configurations. Fortunately, only a small fraction of these query configurations are ever used for any given database. The aim of this part of the application development process is to reduce the complexity inherent in SQL by providing a limited set of query configurations to the novice user.

The key step in creating data retrieval capabilities is the adoption of a satisfactory set of criteria that a user can specify in order to obtain the required data from the database. These criteria will depend on the data being queried. For the GPA Database, some of the possible search criteria obtained from a preliminary survey of the end users were “Component Name,” “Author,” “Temperature Range,” and “Pressure Range.” Of these, “Component Name” and “Author” were selected for implementation in the overall

search strategy. Since the GPA Database includes six types of data, “Data Type” was also included as one of the search criteria.

Once the search strategies have been defined, the various queries required for implementing these criteria are created. The generalization of these queries lies in the use of parameters in the SQL statements. Thus, the specification of the parameters by the end user completes the query at run time. Parameters were used in the GPA Database for obtaining the component and author names.

Once the queries for data retrieval have been created, the forms required for parameter input and data output need to be designed. Forms are the backbone of any database application and hence need to be designed with care. Visual impact of the forms must be considered. A consistent look and feel goes a long way in increasing the usability of the database.

The setup of the forms for the GPA Database has a “Main Menu” as the anchor form. The user navigates in various directions from this form depending on the type of data required and the search strategy selected.

Data Maintenance

Most technical databases will need the data to be updated from time to time. Some will also require data to be added. At times, data found to be in error might need to be deleted. All of these functions are grouped under data maintenance activities. Data maintenance functions usually require a security scheme to protect the data already in the

database. This security scheme will also prevent inadvertent data modifications or deletions.

To modify data in the GPA Database, a user must logon with a password specific to that particular copy of the database. This password can be changed if required. Data display forms created for data retrieval were modified for data entry and update. These modifications involved the implementation of application level validation rules. Another feature required in data entry forms was the automatic creation of primary keys.

Data Manipulation

It is not always sufficient to simply view technical data. They need to be presented to the user in a form that permits further processing. For thermodynamic data, this could be a graph, a spreadsheet, or a text file. These forms are the substitutes for reports used in conventional databases.

Graphs

Considering that most experimental data stored in a technical database more coherent to the end user when presented in the form of graphs, it becomes an essential requirement for a DBMS to be able to generate basic graphs from numerical data. Most DBMS's implement plot generation by means of OLE. This form of OLE technology usage is normally made invisible to the end-user because it is usually included as part of

the DBMS package. However, a given plotting application might not satisfy the plotting requirements of the data being stored in the database. In such cases, it becomes necessary to employ a third party application. This third party application need not necessarily be an OLE server. In such cases, the application's in-built library of plotting functions can be used to create the necessary graphs.

MS Access includes an OLE plotting application called MS Graph. This application is capable of plotting common plots such as x-y plots, scatter graphs, bar graphs, pie charts, etc. However this capability is limited to a user of the full application and is not fully supported for databases created as stand alone applications. To overcome this limitation, Graphics Server SDK[®] (Pinnacle, 1995), a third party plotting application, was employed. The two applications were linked using the standard application programmers interface (API) provided with the plotting application. The GPA Database uses the Graphics Server SDK library provided as a dynamically linked library (DLL) to create the required plots at run time.

Spreadsheets

Most technical data can be represented well in the form of spreadsheets. Spreadsheet applications are also convenient for manipulating data, and many end-users of technical databases require compatibility with spreadsheet applications. The GPA Database provides export capabilities to MS Excel[®]. MS Excel, like MS Access, is developed and supported by Microsoft and is one of the most widely used spreadsheet

application on personal computers (PC). Most other PC spreadsheet applications have maintained compatibility with the MS Excel spreadsheet format. Due to this high level of compatibility with the MS Excel spreadsheet format, the spreadsheet export capabilities provided by the GPA Database are sufficient for most end users of the database.

Text Files

A text file is the most basic mode of communication between applications. Export capabilities to a simple ASCII based text file format are crucial if a database is to provide data manipulation capabilities. The GPA Database provides export capabilities to a generic text file and to MS Word. MS word is one of the most widely used word processing applications, and this compatibility was retained . The generic text file that is generated by MS Access during the export procedure can be in one of two formats -- delimited or fixed width. The delimited format inserts a special delimiting character between fields. The use of the fixed width format requires the specification of the different fields to be exported and their widths.

The GPA Database exports data only as delimited text, because specification of the fixed width format is not possible for all the different formats that might be required by the various end users. The delimited text can be converted to any required format using formatting routines written in any common programming language such as FORTRAN, C, BASIC, etc.. This procedure makes the exported data suitable for use by any user supplied routine that requires input files in the form of ASCII based text files.

Database Independence

This part of the design process is required for databases that must be distributed to end users who do not have the DBMS used to create it. It involves the creation of a run-time version that can be executed on the platform it was designed for without the use of other software. Many technical databases are created for commercial use and require database independence. Some of the steps required to accomplish database independence are discussed below.

One of the first steps required for creating a run-time version of the database application is to obtain a run-time version of the DBMS, and the appropriate licensing agreements. MS Access offers a run-time version of the DBMS that can be distributed royalty free. Most run-time versions of DBMS's severely limit the capabilities of the user, and thus, the application developer is required to create an error free and user friendly application.

To make an application user friendly the application developer must make the various forms work together to provide the appearance of a self contained application. This can normally be done by creating routines that integrate form navigation with other functions. In the GPA Database, form navigation was accomplished using Access Basic code stored in modules and behind forms.

Error checking routines need to be incorporated in order to prevent the application from crashing during execution. Bug detection must be carried out before release of the database. Another approach toward debugging is the release of a beta version of the

database to receive feedback from the end-users. These forms of error checking were carried out for the beta version of the data base.

Any application, however user friendly, will always need hard copy documentation and/or on-line help to be truly user friendly. The documentation for a technical database is not limited to a description of the software capabilities of the application. Also required is a description of the data contained within the database and a general outline of the format used for storing them. The GPA Database documentation includes printed details of the various functions and data formats of the application.

Summary

The design procedure used above for the design of the GPA Database is summarized below in the form of an algorithm. This generalized design procedure is aimed at the design of chemical engineering databases using the relational data model.

Data Assessment

1. Analyze the problem definition for the presence of any historical data that might need to be incorporated into the database being created.
2. Analyze the format of the historical data, if present.
3. Carry out error checking routines on the existing data.
4. Transfer the historical data to the selected DBMS.

E-R Modeling

5. Create a preliminary E-R model of the data using the concept of entities.
6. Pseudo-normalize the preliminary E-R model by observation.

Logical Design

7. Transform entities into relational tables, and represent attributes attached to the entities as fields in the table.
8. Transform the relationships between entities into logical links between tables. Create foreign keys and gerunds where required.
9. Normalize the logical design created from the E-R model using normalization rules. Bring the database at least to 3NF.

Physical Design

10. Select data types based on requirements specified by logical design. Consider factors such as storage space and future modifications while selecting data types.
11. Specify table level validation rules for individual fields.
12. Select indexing requirements for frequently queried fields.
13. Implement relationships with referential integrity constraints. Consider the use of cascading updates and deletes.

Application Development

14. Finalize search strategy. Create queries for data retrieval and forms for viewing the retrieved data.

15. Create data entry forms. Implement a security scheme to protect data integrity.
16. Create plotting routines for the underlying data.
17. Implement data export capabilities to spreadsheets and text files.
18. Write and debug code for form navigation and automation of other functions.
19. Create and distribute a run-time version of the database for evaluation.
20. Implement suggestions obtained from feedback and remove all known errors.
21. Create a final runtime version of the database.

The above design procedure can be very effective when used in conjunction with the relational implementation examples of the previous sections. The examples demonstrate exceptions to business oriented applications created by chemical engineering data. Handling of these exceptions was addressed and generalized for technical data. This allows the procedure to be used for most kinds of chemical engineering data and not solely thermodynamic data.

CHAPTER V

SUMMARY

The flat-file data contained in the GPA Database were re-structured using the relational data model as explained in the previous chapter. During the design process various advantages obtained by the relational implementation of the GPA Database were discussed. Given below is a summary of these advantages.

1. Superior data integrity provided by the relational data model due to removal of redundancy.
2. Faster data retrieval due to the availability of SQL to query relational data.
3. Easier maintenance of the database due to the greater flexibility afforded by the relational data model. For example, any number of components in a mixture can now be accommodated by the database.
4. Ability to be ported to any relational DBMS on any platform due to the underlying relational data model common to all RDBMS's.
5. Longer life expectancy of the database due to wide acceptance of the relational model.
6. Easy to understand the database structure because of the underlying set theory and the vast amount of literature published on the relational data model.

In addition to structuring data using the relational model it is also necessary to add functionality to the database using a relational database management system. Given

below is a summary of the functionality obtained through the use of MS Access as a RDBMS for the GPA Database.

1. Standard search routines built into the database to facilitate easy querying of data without prior knowledge of relational theory or SQL.
2. Superior data integrity due to automated error checking using built-in validation rules such as automatic checking of mole fraction range.
3. Transparent unit conversion leading to the ability to store data in the units in which they were originally reported.
4. Potential for automated output format conversion in order to better interface the database with chemical engineering software such as simulation and modeling applications.
5. Self-contained application using run-time version of the RDBMS for ease of distribution.

In order to realize the gains associated with the above functionality, it is important to implement easier access to data. This ease of access was implemented using a form based graphical user interface (GUI) for the GPA Database and resulted in the following advantages.

1. Implementation of more search parameters, such as T and P, without increasing complexity, which leads to more efficient searches.
2. Better availability due to various output options such as tables, plots, text files, and spreadsheets.
3. Improved data entry efficiency due to ease of use of data entry forms.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

Given below are the conclusions drawn from the results of the relational implementation of the GPA Database.

1. Chemical engineering databases can be efficiently implemented using the relational data model.
2. The relational data model is better suited for the implementation of a chemical engineering database than the flat-file model.
3. Implementation of a chemical engineering database using the relational data model makes it possible to provide easier access to the underlying data using a user-friendly interface. This interface if implemented, can reduce the barrier between the end-user and the data and hence provide a potential for increased use of the database.

The life of a database depends on how well it was designed at conception as well as how well it is maintained. Given below are recommendations for the maintenance of and future additions to the GPA Database:

1. Carry out systematic analysis of existing data using standard tests such as thermodynamic consistency tests and closure methods.
2. Add new experimental data to the database as they become available in order to keep the database current.

3. Create new interfaces to chemical engineering software applications such as ASPEN PLUS[®], HYSIM[®], etc.
4. Implement porting of data to a database server capable of serving data over a Local Area Network (LAN) or possibly the Internet, thus ensuring easier access to data.
5. Incorporate established models or equations-of-state to predict other thermodynamic properties, and to interpolate or extrapolate from existing data. These can be implemented using procedural calls to a dynamically linked library (DLL) using the Application Program Interface (API).
6. Create built-in output formats based on popular standards existing in the chemical process industries such as STEP (STAndards for the Exchange of Process model data) proposed by the Process Data Exchange Institute (PDXI) (Books, 1994).
7. Periodically evaluate the status of Object Oriented DataBase Management Systems (OODBMS) for suitability, because relational database technology may be displaced by object oriented database technology as the premier database model in the future.

GLOSSARY OF TERMS
(Modified from McFadden, 1993)

Anomalies Errors or inconsistencies that may result when a user attempts to update a table containing redundant data. There are three types of anomalies: insertion, deletion, and modification anomalies.

Application program interface (API) Software that allows a specific front-end program development platform to communicate with a particular back-end application or database engine, even when the front-end and back end were not built to be compatible.

Attribute A named property or characteristic of an entity that has informational value.

Boyce-Codd normal form (BCNF) A relation in which every determinant is a candidate key.

Candidate Key An attribute (or combination of attributes) that uniquely identifies each instance of an entity type.

Cardinality The number of instances of entity B that can (or must) be associated with each instance of entity A.

Complex network data model A type of network data model that supports N:M (as well as 1:M) relationships.

Composite Key A primary key that contains more than one attribute.

Computer-aided software engineering (CASE) tools Software tools that provide automated support for some portion of the system development process.

Conceptual data model A detailed model that captures the overall structure of the data, while being independent of any database management system or other implementation considerations.

Data Facts concerning things such as people, objects or events.

Database A shared collection of logically related data, designed to meet the information needs of multiple users.

Database management system (DBMS) A software application system that is used to create, maintain, and provide controlled access to data.

DBMS engine The central component of a DBMS. Provides access to the repository and the database and coordinates all of the other functional elements of the DBMS.

Degree The number of entity types that participate in a relationship.

Determinant The attribute on the left hand side of the arrow in a functional dependency; A is a determinant in the following functional dependency: $A \rightarrow B$.

Encapsulation The property that the attributes and methods of an object are hidden from the outside world and do not have to be known to access its data values or invoke its methods.

Entity A thing (e.g., person, place, event, or concept) about which it is chosen to record data.

Entity-relationship data model (E-R model) A detailed, logical representation of the entities, associations, and data elements for an organization, or structure.

Entity-relationship diagram (E-R diagram) A graphical representation of an E-R model.

Fifth normal form (5NF) A relation is in fifth normal form if it is in fourth normal form and does not have a join dependency.

First normal form (1NF) A relation that contains no repeating groups.

Foreign key An attribute that appears as a nonkey attribute in one relation and as a primary key attribute (or part of a primary key) in another relation.

Fourth normal form (4NF) A relation is in fourth normal form if it is in BCNF and contains no multivalued dependencies.

Functional Dependency A particular relationship between two attributes. For any relation R, attribute B is functionally dependent on attribute A if, for every valid instance of A, that value of A uniquely determines the value of B. The functional dependency of B on A is represented as $A \rightarrow B$.

Generalization The concept that some things (entities) are subtypes of other, more general things.

Generalization hierarchy A hierarchical grouping of objects that share common attributes and methods.

Gerund A many to many relationship that the data modeler chooses to model as an entity type with several associated one-to-many relationships with other entity types.

Hierarchical database model A data model in which records are arranged in a top-down structure that resembles a tree.

Homonym A single name that is used for two different data items (for example, the term invoice is used to refer to both a customer invoice and a supplier invoice).

Information Data that have been processed and presented in a form suitable for human interpretation, often with the purpose of revealing trends or patterns.

Inheritance The property that, when entity types or object classes are arranged in a hierarchy, each entity type or object class assumes the attributes and methods of its ancestors (that is, those higher up in the hierarchy).

Integrated CASE (I-CASE) toolset A set of case tools that can support all phases of the system development process.

ISA relationship The relationship between each subtype and its supertype, where each subtype *is a* supertype.

Join A relational operation that causes two tables with a common domain to be combined into a single table.

Join dependency A relation that has a join dependency cannot be divided into two (or more) relations such that the resulting tables can be recombined to form the original table.

Logical database design The process of mapping logical database models to structures that are specific to a target DBMS.

Logical database model A design that conforms to the data model specific to a class of database management systems.

Method (or service) A processing routine that is encapsulated in an object and operates on the data described within that object.

Methodology A process (or related series of steps) to accomplish a design goal, together with a set of design objects that are manipulated to support the process.

Multivalued attribute An attribute that can have more than one value for each entity instance.

Multivalued dependency A type of dependency that exists when there are at least three attributes (for example A, B, and C) in a relation, and for each value of A there is a well-defined set of values for B and a well-defined set of values for C, but the set of values of B is independent of set C.

Network database model A data model in which each record type may be associated with an arbitrary number of different record types.

Normal form A state of a relation that can be determined by applying simple rules regarding dependencies to that relation.

Normalization The process of converting complex data structures into simple, stable data structures.

Null value A special column value, distinct from 0, blank, or any other value, that indicates that the value for the column is missing or otherwise unknown.

Object A structure that encapsulates (or packages) attributes and methods that operate on those attributes.

Object class A logical grouping of objects that have the same (or similar) attributes and behavior (or methods).

Object instance One occurrence (or materialization) of an object class.

Object-oriented database model A database model in which data attributes and methods that operate on those attributes are encapsulated in structures called objects.

Partial functional dependency A dependency in which one or more nonkey attributes are functionally dependent on part (but not all) of the primary key.

Physical database design The process of mapping the database structures from a logical design to physical storage structures such as files and tables. Indexes are also specified, as well as access methods and other physical factors.

Primary key A candidate key that has been selected as the identifier for an entity type. Primary key values may not be null.

Record type A named entity, instances of which describe individual occurrences of the entity.

Referential integrity An integrity constraint that specifies that the value (or existence) of an attribute in one relation depends on the value (or existence) of the same attribute in another relation.

Referential integrity constraint A business rule that addresses the validity of references by one object in a database to some other object (or objects) in the database.

Relation A named two-dimensional table of data. Each relation consists of a set of named columns and an arbitrary number of unnamed rows.

Relational database model A data model that represents data in the form of tables or relations.

Relational DBMS (RDBMS) A database management system that manages data as a collection of tables in which all data relationships are represented by common values in related tables.

Relationship An association between the instances of one or more entity types that is of interest to the database user.

Repeating group A set of two or more multivalued attributes that are logically related.

Run-time version The portion of the DBMS needed to run an existing database application.

Schema A description of the overall logical structure of a database, expressed in a special data definition language.

Second normal form (2NF) A relation is in second normal form if it is in first normal form and every nonkey attribute is fully functionally dependent on the primary key. Thus no nonkey attribute is functionally dependent on part (but not all) of the primary key.

Simple network model A type of network data model that supports 1:M (but not N:M) relationships.

Structured query language (SQL) A standard fourth generation query language for relational database systems.

Subtype A subset of a supertype that shares common attributes or relationships distinct from other subsets.

Supertype A generic entity type that is subdivided into subtypes.

Synonym Two different names that are used to describe the same data item (for example, car and automobile).

Third normal form (3NF) A relation is in third normal form if it is in second normal form and no transitive dependencies exist.

Transitive dependencies A functional dependency between two (or more) nonkey attributes in a relation.

Tree A data structure that consists of a set of nodes that branch out from a node at the top of the tree (thus the tree is upside down).

View A virtual table in the relational data model in which data from real (base) tables are combined so that programmers can work with just one (virtual) table instead of the several of more complete base tables.

GLOSSARY OF ACRONYMS

1:1	One-to-one
1:M	One-to-many
1NF	First Normal Form
2NF	Second Normal Form
3NF	Third Normal Form
4GL	Fourth Generation Language
4NF	Fourth Normal Form
5NF	Fifth Normal Form
ANSI	American National Standards Institute
API	Application Program Interface
ASCII	American Standards Code for Information Interchange
BCNF	Boyce-Codd Normal Form
C	Component ID
CASE	Computer-Aided Software Engineering
CN	Component Number
COBOL	COmmon Business Oriented Language
CODASYL	COmmittee On DATA SYstem Languages
CPI	Chemical Process Industries
CR	Critical Region
DBMS	DataBase Management System
DLL	Dynamically Linked Library
DOS	Disk Operating System
DPN	Data Point Number
DSN	Data Set Number
DS	Data Source
E-R	Entity-Relationship
FORTRAN	FORmula TRANslator
GPA	Gas Processors Association
GUI	Graphical User Interface
IBM	International Business Machines
I-CASE	Integrated Computer-Aided Software Engineering
ID	IDentifier
IMS	Information Management System
ISA	<i>is a</i> relationship
LAN	Local Area Network
MS	MicroSoft
NC	Number of Components
OODBMS	Object-Oriented DataBase Management System

OODM Object-Oriented Data Model
OLE Object Linking and Embedding
P Pressure
PC Personal Computer
QBE Query By Example
RAM Random Access Memory
RDBMS Relational DataBase Management System
RN Reference Number
RR Research Report
SQL Structured Query Language
SSN Social Security Number
T Temperature
TC Thermodynamic Consistency
VLE Vapor Liquid Equilibrium
X Liquid Mole Fraction
Y Vapor Mole Fraction

APPENDICES

APPENDIX A

SAMPLE DATA FILE
CONTAINING
VLE DATA

1	2	58	METHANE	ETHANE	HYDROGEN		6
-100.00	500.00	0.2942E+00	0.4586E+00	0.6925E+00	0.7910E-01	VL G	3058001043001
0.1330E-01	0.4623E+00					VL G	3058001043001
-100.00	500.00	0.3255E+00	0.5230E+00	0.6611E+00	0.8300E-01	VL G	3058001043002
0.1340E-01	0.3940E+00					VL G	3058001043002
-100.00	500.00	0.3859E+00	0.6153E+00	0.6021E+00	0.7850E-01	VL G	3058001043003
0.1200E-01	0.3062E+00					VL G	3058001043003
-100.00	500.00	0.4325E+00	0.6913E+00	0.5574E+00	0.7790E-01	VL G	3058001043004
0.1010E-01	0.2308E+00					VL G	3058001043004
-100.00	1000.00	0.4410E-01	0.3240E-01	0.9208E+00	0.3510E-01	VL G	3058001043005
0.3510E-01	0.9325E+00					VL G	3058001043005
-100.00	1000.00	0.1153E+00	0.1102E+00	0.8447E+00	0.5260E-01	VL G	3058001043006
0.4000E-01	0.8372E+00					VL G	3058001043006

APPENDIX B

FORTRAN CODE
FOR
FORMAT CHANGE

```

C      VLE3

      INTEGER C1,C2,C3,STDP,CNO,RNO,DP,TTDP
      REAL T,P,X1,X2,X3,Y1,Y2,Y3
      CHARACTER*1 TC,CR,DS
      CHARACTER*3 DT

      OPEN (UNIT = 8,FILE = 'VLE3.TXT',STATUS = 'UNKNOWN')
      OPEN (UNIT = 9,FILE = 'VLE3A.TXT',STATUS = 'UNKNOWN')

      READ(8,1000) C1,C2,C3,STDP

      DO 100 J = 1,STDP,1

      READ(8,1010) T,P,X1,Y1,X2,Y2,TC,DT,CR,DS,CNO,RNO,DP

      READ(8,1020) X3,Y3
      WRITE(9,1030) T,P,X1,Y1,X2,Y2,X3,Y3,
      *TC,DT,CR,DS,CNO,C1,C2,C3,RNO,DP

100    CONTINUE

1000   FORMAT(3I3,67X,I3)
1010   FORMAT(2F8.2,4E11.4,A1,A3,2A1,I2,6X,2I3)
1020   FORMAT(2E11.4)
1030   FORMAT(2F8.2,6E11.4,A1,A3,2A1,I2,5I3)
      STOP
      END

```

APPENDIX C

FORMATTED TEXT FILE

FOR MS ACCESS

-100.00	500.00	.2942E+00	.4586E+00	.6925E+00	.7910E-01	.1330E-01
.4623E+00	VL G 3	1	2 58 43	1	↓	
-100.00	500.00	.3255E+00	.5230E+00	.6611E+00	.8300E-01	.1340E-01
.3940E+00	VL G 3	1	2 58 43	2	↓	
-100.00	500.00	.3859E+00	.6153E+00	.6021E+00	.7850E-01	.1200E-01
.3062E+00	VL G 3	1	2 58 43	3	↓	
-100.00	500.00	.4325E+00	.6913E+00	.5574E+00	.7790E-01	.1010E-01
.2308E+00	VL G 3	1	2 58 43	4	↓	
-100.00	1000.00	.4410E-01	.3240E-01	.9208E+00	.3510E-01	.3510E-01
.9325E+00	VL G 3	1	2 58 43	5	↓	
-100.00	1000.00	.1153E+00	.1102E+00	.8447E+00	.5260E-01	.4000E-01
.8372E+00	VL G 3	1	2 58 43	6	↓	

APPENDIX D

VLE DATA FORMAT

T (Temperature) This field stores the temperature associated with the given data point in Fahrenheit.

P (Pressure) Pressure associated with the data point in psia.

X1, X2, X3 (Liquid Mole Fractions) These three fields store the liquid composition information as individual mole fractions.

Y1, Y2, Y3 (Vapor Mole Fractions) Vapor composition information is stored in these three fields as individual mole fractions.

TC (Thermodynamic Consistency) This field contains a “T” if the data point that it belongs to has been tested to be thermodynamically consistent. “I” signifies thermodynamic inconsistency and a blank or “?” denotes that it has not been tested.

DT (Data Type) Data type is used to signify the kind of data that the data point represents. “VL” stands for vapor liquid equilibrium data.

CR (Critical Region) This field contains a C if the temperature or pressure for this data point are within 2% of the critical temperature or critical pressure of the system.

DS (Data Source) Data source is used for identifying the origin of the data. A value of “G” for this field implies that the data was obtained from a reference list supplied by the Gas Research Institute (GRI). “K” signifies that the data was obtained from Knapp (Berlin Data Book).

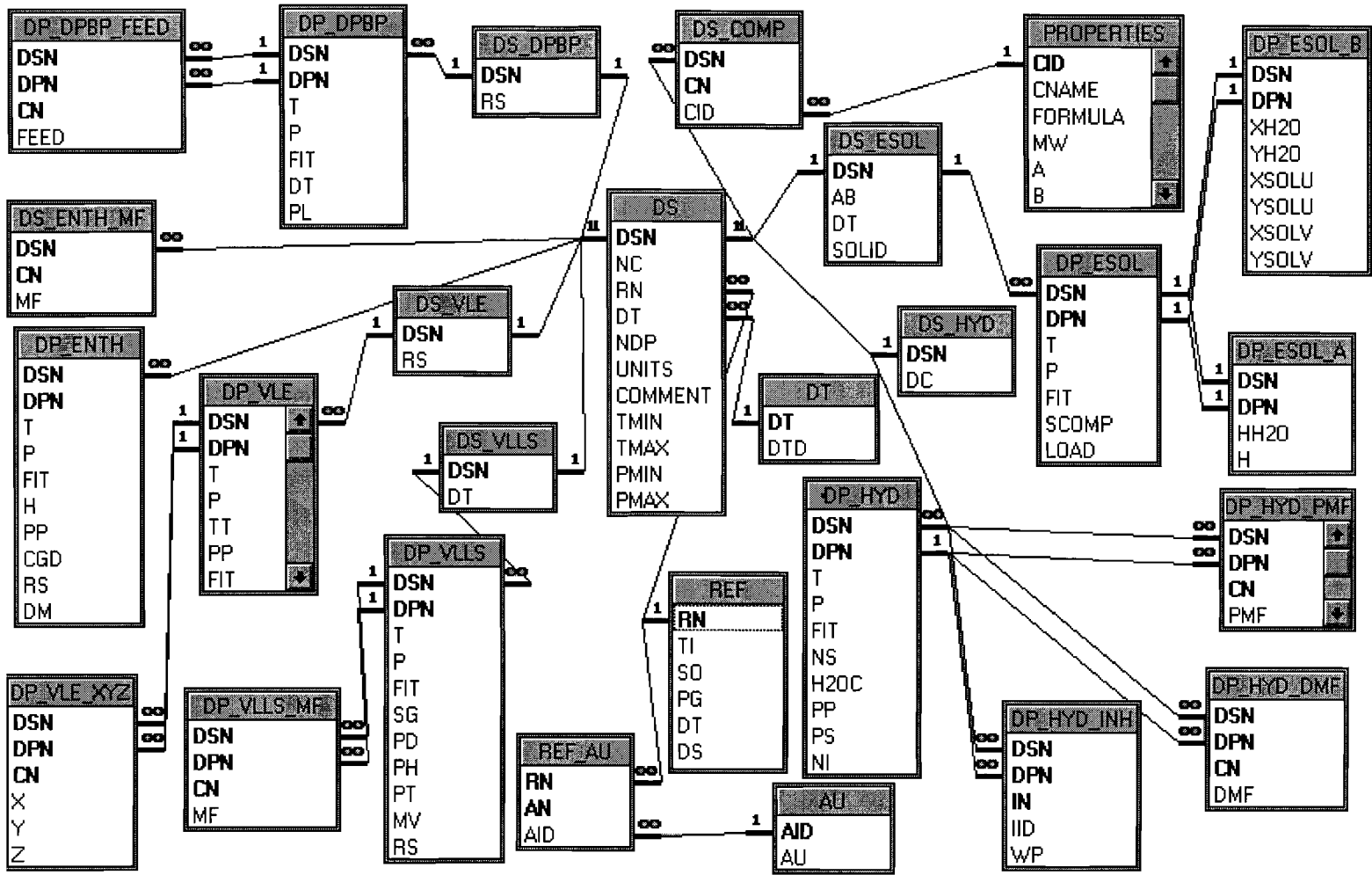
NC (Number of Components) NC stores information on the number of components contained in the mixture to which the concerned data point belongs.

C1, C2, C3 (Components) These fields store the component ID’s of the components present in the mixture of which the data point is a part.

RN (Reference Number) Information on the source of the data is stored in this field in the form of a reference number that indicates the publication information of the data.

DPN (Data Point Number) This field acts as a counter for a given data set by containing a unique and incrementing value for each data point.

APPENDIX E
.
DATABASE
RELATIONSHIPS



APPENDIX F

DATABASE

DICTIONARY

AU	Fields: 2	Records: 427
Field Name	Field Type	Description
AID	Integer (2)	Used for Author identification.
AU	Character (50)	Author's Name
Index Name	Fields	Properties
PrimaryKey	1	Primary, Unique, Required

COMP	Fields: 2	Records: 108
Field Name	Field Type	Description
CID	Integer (2)	Used for component identification.
CNAME	Character (35)	Chemical name of the component.
Index Name	Fields	Properties
PrimaryKey	1	Primary, Unique, Required

DP_DPBP	Fields: 7	Records: 994
Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.
T	Single (4)	Temperature for this data point in Farenheit.
P	Single (4)	Pressure for this data point in psia.
FIT	Single (4)	Estimated normalized accuracy of this data point.
DT	Character (2)	DP = Dew Point, BP = Bubble Point.
PL	Single (4)	Percent liquid.
Index Name	Fields	Properties
PrimaryKey	2	Primary, Unique, Required

DP_DPBP_FEED	Fields: 4	Records: 2426
Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.
CN	Integer (2)	Component number in the mixture.
FEED	Single (4)	Feed mole fraction for this mixture.
Index Name	Fields	Properties
PrimaryKey	3	Primary, Unique, Required

DP_ENTH	Fields: 10	Records: 18660
Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.

T	Single (4)	Temperature for this data point in Farenheit.
P	Single (4)	Pressure for this data point in psia.
FIT	Single (4)	Estimated normalized accuracy of this data point.
H	Single (4)	Enthalpy in (BTU/lb).
PP	Character (5)	Phase code of the mixture.
CGD	Character (1)	# indicates that enthalpy departure or phase was changed.
RS	Character (1)	R = Raw, S = Smooth, ? = Unevaluated.
DM	Character (1)	Departure method used.
Index Name	Fields	Properties
PrimaryKey	2	Primary, Unique, Required

DP_ESOL Fields: 7 Records: 1879

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.
T	Single (4)	Temperature for this data point in Farenheit.
P	Single (4)	Pressure for this data point in psia.
FIT	Single (4)	Estimated normalized accuracy of this data point.
SCOMP	Single (4)	Weight percent in water of solvent.
LOAD	Single (4)	Loading (mol solute / mole H2o free solvent).
Index Name	Fields	Properties
PrimaryKey	2	Primary, Unique, Required

DP_ESOL_A Fields: 4 Records: 1868

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.
HH2O	Single (4)	Enthalpy of solution (BTU / lb of H2O free solvent)
H	Single (4)	Enthalpy of solution (BTU / lb solute).
Index Name	Fields	Properties
PrimaryKey	2	Primary, Unique, Required

DP_ESOL_B Fields: 8 Records: 11

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.
XH2O	Single (4)	Liquid mole fraction of H2O.
YH2O	Single (4)	Vapor mole fraction of H2O.
XSOLU	Single (4)	Liquid mole fraction of solute.
YSOLU	Single (4)	Vapor mole fraction of solute.
XSOLV	Single (4)	Liquid mole fraction of solvent.
YSOLV	Single (4)	Vapor mole fraction of solvent.
Index Name	Fields	Properties

PrimaryKey 2 Primary, Unique, Required

DP_HYD Fields: 10 Records: 1963

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.
T	Single (4)	Temperature for this data point in Farenheit.
P	Single (4)	Pressure for this data point in psia.
FIT	Single (4)	Estimated normalized accuracy of this data point.
NS	Integer (2)	Total number of T, P states in set.
H2OC	Single (4)	H2O content for this data point.
PP	Character (9)	Phases present at this data point.
PS	Character (3)	Phase being measured.
NI	Integer (2)	Number of inhibitors present for this data set.
Index Name	Fields	Properties
PrimaryKey	2	Primary, Unique, Required

DP_HYD_DMF Fields: 4 Records: 6478

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.
CN	Integer (2)	Component number in the mixture.
DMF	Single (4)	Dry mole fraction of the component in the mixture.
Index Name	Fields	Properties
PrimaryKey	3	Primary, Unique, Required

DP_HYD_INH Fields: 5 Records: 315

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.
IN	Integer (2)	Inhibitor number in the hydrate.
IID	Integer (2)	Used for identifying the inhibitor.
WP	Single (4)	Weight percent of the inhibitor.
Index Name	Fields	Properties
PrimaryKey	3	Primary, Unique, Required

DP_HYD_PMF Fields: 4 Records: 1145

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.
CN	Integer (2)	Component number in the mixture.

PMF	Single (4)	Phase mole fractions of the components in the system.
Index Name	Fields	Properties
PrimaryKey	3	Primary, Unique, Required

DP_VLE Fields: 7 Records: 11453

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.
T	Single (4)	Temperature for this data point in Farenheit.
P	Single (4)	Pressure for this data point in psia.
FIT	Single (4)	Estimated normalized accuracy of this data point.
TC	Character (1)	T = Thermodynamically consistent, I = inconsistent.
CR	Character (1)	C implies Temp is within 2 % of critical temp.
Index Name	Fields	Properties
PrimaryKey	2	Primary, Unique, Required

DP_VLE_XYZ Fields: 6 Records: 25917

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.
CN	Integer (2)	Component number in the mixture.
X	Single (4)	Liquid mole fraction of the component in the mixture.
Y	Single (4)	Vapor mole fraction of the component in the mixture.
Z	Single (4)	Feed mole fraction of the component in the mixture.
Index Name	Fields	Properties
PrimaryKey	3	Primary, Unique, Required

DP_VLLS Fields: 11 Records: 4492

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.
T	Single (4)	Temperature for this data point in Farenheit.
P	Single (4)	Pressure for this data point in psia.
FIT	Single (4)	Estimated normalized accuracy of this data point.
SG	Character (5)	Sub group identifiactation.
PD	Character (10)	Phase designation.
PH	Character (3)	Phase being analyzed.
PT	Character (5)	Phase type.
MV	Single (4)	Molar volume (ml / gram-mole).
RS	Character (1)	R = Raw, S = Smoothed, "" = Unevaluated.
Index Name	Fields	Properties
PrimaryKey	2	Primary, Unique, Required

DP_VLLS_MF	Fields: 4	Records: 13239
Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DPN	Integer (2)	Used as a primary key for data points.
CN	Integer (2)	Component number in the mixture.
MF	Single (4)	Mole fraction of the component in the mixture.
Index Name	Fields	Properties
PrimaryKey	3	Primary, Unique, Required

DS	Fields: 10	Records: 757
Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
NC	Integer (2)	Number of components in the mixture.
RN	Integer (2)	Literature reference number.
DT	Integer (2)	Used for identifying the data type.
NDP	Integer (2)	Total number of data points in the data set.
COMMENT	Memo	Special comments relating to this data set.
TMIN	Single (4)	Minimum temperature (F) for the data set.
TMAX	Single (4)	Maximum temperature (F) for the data set.
PMIN	Single (4)	Minimum pressure (psia) for the data set.
PMAX	Single (4)	Maximum pressure (psia) for the data set.
Index Name	Fields	Properties
DT	1	
PrimaryKey	1	Primary, Unique, Required

DS_COMP	Fields: 3	Records: 1888
Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
CN	Integer (2)	Component number in the mixture.
CID	Integer (2)	Used for component identification.
Index Name	Fields	Properties
CID	1	
Reference	1	Foreign

DS_DPBP	Fields: 2	Records: 21
Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
RS	Character (1)	R = Raw, S = Smooth, ? = Unevaluated.
Index Name	Fields	Properties
DSN	1	Unique
Reference25	1	Foreign, Unique

DS_ENTH_MF Fields: 3 Records: 385

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
CN	Integer (2)	Component number in the mixture.
MF	Single (4)	Mole fraction of the component in the mixture.
Index Name	Fields	Properties
PrimaryKey	2	Primary, Unique, Required

DS_ESOL Fields: 4 Records: 10

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
AB	Character (1)	Used for determining the data type of the record.
DT	Integer (2)	Used for determining additional attribute values of the record.
SOLID	Integer (2)	Used for determining Solvent Name for the mixture.
Index Name	Fields	Properties
PrimaryKey	1	Primary, Unique, Required

DS_HYD Fields: 2 Records: 114

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DC	Character (2)	UH = Uninhibited, IH = Inhibited Hydrate.
Index Name	Fields	Properties
PrimaryKey	1	Primary, Unique, Required

DS_VLE Fields: 2 Records: 341

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
RS	Character (1)	R = Raw, S = Smooth, ? = Unevaluated.
Index Name	Fields	Properties
PrimaryKey	1	Primary, Unique, Required

DS_VLLS Fields: 2 Records: 77

Field Name	Field Type	Description
DSN	Integer (2)	Used as a primary key for data sets.
DT	Character (3)	Vapor Liquid Liquid or Vapor Liquid Solid Equilibria.
Index Name	Fields	Properties
PrimaryKey	1	Primary, Unique, Required

PROPERTIES Fields: 30 Records: 108

Field Name	Field Type	Description
CID	Integer (2)	Used of rcomponent identification.
FM	Character (50)	Chemical Formula
MW	Single (4)	Molecular Weight
BP	Single (4)	Boiling Point (F)
VP	Single (4)	Vapor Pressure (psia)
FP	Single (4)	Freezing Point (F)
RI	Single (4)	Refractive Index
CRP	Single (4)	Critical Pressure (psia)
CRT	Single (4)	Critical Temperature (F)
CV	Single (4)	Critical Volume (cuft./lb)
RDL	Single (4)	Relative Density (liq.)
LD	Single (4)	Liquid Density (lbm/gal)
TCD	Single (4)	Temp. Coef. of Density (1/F)
AF	Single (4)	Accentric Factor
CF	Single (4)	Compressibility Factor
RDG	Single (4)	Relative Density (Gas)
GD	Single (4)	Gas Density (cuft./lb)
GLD	Single (4)	Gas/Liq. Density (cuft. gas/gal liq.)
CPG	Single (4)	Cp Gas (BTU/lbm F)
CPL	Single (4)	Cp Liq (BTU/lbm F)
NHVG	Single (4)	Net Heat Value (BTU/cuft. Ideal Gas)
NHVL	Single (4)	Net Heat Value (BTU/lbm liq.)
GHVG	Single (4)	Gross Heat Value (BTU/cuft. Ideal Gas)
GHVL	Single (4)	Gross Heat Value (BTU/lbm liq.)
HV	Single (4)	Heat of Vaporization (BTU/lbm)
C	Single (4)	Combustibility (cuft./cuft.)
FLL	Single (4)	Flammability Limit (Low)(Vol%)
FLH	Single (4)	Flammability Limit (High)(Vol%)
ON357	Single (4)	Octane Number (D-357)
ON908	Single (4)	Octane Number (D-908)
Index Name	Fields	Properties
PrimaryKey	1	Primary, Unique, Required

REF Fields: 6 Records: 414

Field Name	Field Type	Description
RN	Integer (2)	Literature reference number.
TI	Memo	Reference title.
SO	Memo	Reference source.
PG	Character (255)	Volume and page details for the reference.
DT	Character (50)	Year of publication.
DS	Character (1)	Stores information on data source. G, K, P, or L.
Index Name	Fields	Properties
PrimaryKey	1	Primary, Unique, Required

REF_AU Fields: 3 Records: 996

Field Name	Field Type	Description
RN	Integer (2)	Literature reference number.
AN	Integer (2)	Author number for the reference.
AID	Integer (2)	Used for Author identification.
Index Name	Fields	Properties
PrimaryKey	2	Primary, Unique, Required
Reference3	1	Foreign

VITA

PARIKSHIT SANGHAVI

Candidate for the Degree of

Master of Science

Thesis: DESIGN AND DEVELOPMENT OF A THERMODYNAMIC
PROPERTIES DATABASE USING THE RELATIONAL DATA
MODEL

Major Field: Chemical Engineering

Biographical:

Personal Data: Born in Madras, Tamil Nadu, India, On May 8, 1971, the son of Harkisandas N. Sanghavi and Nalini Sanghavi.

Education: Graduated from Vidya Mandir Higher Secondary School, Madras, India, in May 1988; received Bachelor of Engineering degree in Chemical Engineering from Birla Institute of Technology and Science, Pilani, India, in June 1992. Completed the requirements for the Master of Science degree with a major in Chemical Engineering at Oklahoma State University in December 1994.

Experience: Summer internship at KCP, Madras, India, 1990; six month internship at GRASIM Industries, Nagda, India, 1992; employed as Management Trainee (Technical) at Britannia Industries, Madras, India, August 1992 to December 1992; employed as Research Assistant by the School of Chemical Engineering, Oklahoma State University, Stillwater, Oklahoma, August 1993 to December 1995.

Professional Memberships: American Institute of Chemical Engineers.