1

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service / Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

National Library
of Canada

Bibliothèque nationale
du Canada

0-315-22167-4

Canadian Theses Division Division des thèses canadiennes

Ottawa, Canada
K1A 0N4

PERMISSION TO MICROFILM — AUTORISATION DE MICROFILMER

• Please print or type — Écrire en lettres moulées ou dactylographier

Full Name of Author — Nom complet de l'auteur

ALAN JOHN GLASS

Date of Birth — Date de naissance

JANUARY 5, 1958

Country of Birth — Lieu de naissance

CANADA

Permanent Address — Résidence fixe

P.O. Box 192
GLEN ROBERTSON, ONTARIO
K0B 1H0

Title of Thesis — Titre de la thèse

REPRESENTATION OF BUILDING STRUCTURES
IN A COMPUTER DATABASE

University — Université

CARLETON UNIVERSITY

Degree for which thesis was presented — Grade pour lequel cette thèse fut présentée

MASTER OF ENGINEERING

Year this degree conferred — Année d'obtention de ce grade

1984

Name of Supervisor — Nom du directeur de thèse

NEAL M. HOLTZ

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'auteur se réserve les autres droits de publication, ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

Date

September 19, 1984

Signature

Alan Glass

**REPRESENTATION OF BUILDING STRUCTURES
IN A COMPUTER DATABASE**

by

© Alan Glass, B. Eng.

A thesis submitted to the Faculty of Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Civil Engineering

Faculty of Engineering
Carleton University
Ottawa, Ontario, Canada
September, 1984

The undersigned recommend to the Faculty of Graduate Studies and Research acceptance of the thesis **"The Representation of Building Structures in a Computer Database"** submitted by Alan Glass, B. Eng., in partial fulfillment of the requirements for the degree of Master of Engineering.

Thesis Supervisor

Chairman,
Department of Civil Engineering

Carleton University
Ottawa

Abstract

This thesis proposes an organization for representing building structures in a computer database for the purpose of creating an integrated system for the computer-aided design of building structures. The role of the building structure database in the integrated design system is discussed and the various types of structure data required to be stored are classified. The relational data model is compared to the hierarchical and network data models and the principle operations of relational mathematics are explained. The relations (individual relational data files) that are used to represent a common class of structure, the diagonally braced steel industrial structure, are defined and implemented in a commercial relational database management system. A sample design is performed using the building structure database and a limited design system created by the author.

Keywords:

database; building structure database; integrated design system; database management system (DBMS); hierarchy; network; relation; relational database; data consistency and integrity; data independence; computer-aided design (CAD).

Acknowledgements

I wish to thank to my thesis supervisor, Professor Neal Holts, for his guidance and encouragement throughout the course of this work. I also wish to express my gratitude to Carleton University for financial assistance received.

Table of Contents

Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Figures	viii
1 Introduction	1
1.1 Plan of the Work	2
1.2 Previous Work	3
2 Role of the Building Structure Database	5
2.1 The Integrated Design System	5
2.2 The Building Structure Database	8
2.3 Structure Data Types	10
2.3.1 <i>Traditional Representation</i>	11
2.4 The Design Process	12
3 Data Models	15
3.1 The Hierarchical Model	16
3.2 The Network Model	17
3.3 The Relational Model	18
3.4 Choice of Relational Model	19
3.5 Relations and Relational Operations	19
4 Data Representation	23
4.1 Representation of the Building Structure	23
4.2 Topology vs. Geometry	27
5 The Building Structure Database	29
5.1 The Structure Description	30
5.2 Loading Data	36
5.2.1 <i>Automatic Calculation of Loading Data</i>	38

5.3	Structure Behaviour Data	38
5.4	Design Criteria	40
5.5	Reference Databases	40
6	Programming Procedures	41
6.1	Internal and External Programs	43
6.2	Basic Data Entry	44
6.3	Geometry from Topology	45
6.4	Analysis	46
6.5	Member Selection	46
7	The Design System	48
7.1	Resource Limitations	49
7.2	Data Files	51
7.2.1	Basic Data Files	51
7.2.2	Calculated Data Files	54
7.2.3	Reference Data Files	55
7.3	Programs and Transient Data Files	56
7.3.1	Basic Data Entry	57
7.3.2	Computing Geometry from Topology	58
7.3.3	Computing External Member Forces	58
7.3.4	Computing Internal Member Forces	59
7.3.5	Member Selection	60
7.4	Limits to Usefulness	61
8	Sample Design	63
8.1	Entry of Basic Data	63
8.1.1	Building Description	64
8.1.2	Loading Data	65
8.1.3	Design Criteria	66
8.2	Running the System	66
8.2.1	Geometry from Topology	67
8.2.2	Analysis	67

8.2.3 Member Selection	67
9 Further Work	101
9.1 Potential Use of the Design Database	101
9.2 A Scheme for Data Maintenance	103
9.3 Data Security	105
9.4 Further Uses of the Database	105
9.5 Expanding the Present System	108
10 Conclusion	109
Appendix A: Data Files & Program Listings	112
Appendix B: dBASE II Commands	153
References	157

List of Figures

2-1	An input/output design system	6
2-2	A database linked system.	7
2-3	A central database system.	7
2-4	The design process.	14
3-1	The DBMS manipulates data in physical storage.	15
3-2	A hierarchy, or tree-structure.	17
3-3	A network.	17
3-4	An example beam relation.	19
3-5	"SELECT TABLE 1 for LENGTH < 7.0."	20
3-6	"PROJECT TABLE 1 over BEAM NAME, SECTION."	20
3-7	An example section relation.	21
3-8	"JOIN TABLE 1 TO TABLE 4 over SECTION."	21
3-9	"PROJECT TABLE 5 over BEAM NAME, DEPTH."	22
4-1	Node and member representation.	25
5-1	Location of axes, column lines and columns.	32
5-2	Beam ends are located along their supporting members.	34
5-3	Panels are defined by the location of their corners.	35
5-4	Girts and tie-rods.	36
7-1	Files, programs and data flow.	50
8-1	Plan at Level 01.	69
8-2	Plan at Level 02.	70
8-3	Elevation on column line "B".	71
8-4	AXES—reference axes system, column line directions.	72
8-5	COLINES—column line locations.	72
8-6	LEVELS—floor level elevations.	73
8-7	COLUMNS—column locations.	73
8-8	BEAMS01—beams on Level 01, initial condition.	74
8-9	BEAMS02—beams on Level 02, initial condition.	75
8-10	PANELS01—floor panels on Level 01.	76
8-11	PANELS01—floor panels on Level 02.	77

8-12	PLX01	distributed loads on Level 01	78
8-13	PLX02	distributed loads on Level 02	79
8-14	CONCE01	concentrated loads on Level 01	79
8-15	CASES	loading combinations, load cases and load factors	80
8-16	BMLIST01	levels of independence for beams on Level 01	81
8-17	BMLIST02	levels of independence for beams on Level 02	82
8-18	BLX01	external beam forces on Level 01	83
8-19	BLX02	external beam forces on Level 02	87
8-20	CLX	external column forces	89
8-21	BMDIAG01	beam shears and bending moments on Level 01	91
8-22	BMDIAG02	beam shears and bending moments on Level 02	94
8-23	MOMENT01	maximum beam shears and bending moments on Level 01	96
8-24	MOMENT02	maximum beam shears and bending moments on Level 02	97
8-25	BMSEL	an abbreviated beam selection table	98
8-26	BEAMS01	beams on Level 01, final condition	99
8-27	BEAMS02	beams on Level 02, final condition	100

Chapter 1

Introduction

The aim of this thesis is to present a practical way of representing a building structure in a computer database, and to demonstrate the use of a building structure database for structural design.

The present is not only a time of increasingly powerful computers but also a time of increasing awareness of the powers of the computer. There is hardly a field in which the use of the computer is not being developed. The computer is an efficient tool not only for computation but also for information management and decision making. It is inevitable that in the pursuit of efficiency and productivity the computer is being universally adopted.

The trend is towards ever greater use of the computer. Once the commitment to full use of the computer is made the working environment must be changed to accommodate and to fully exploit the new machine. Efficiency is increased by transferring to the computer as much work as possible, doing manually only what the computer is incapable of. As the computer becomes of major importance to an enterprise it will be necessary to store data in a form that can be directly accessed by the machine, that is, in a computer database.

The computer already plays an important role in the design of building structures. Existing computer programs provide a means for handling problems not suited to manual calculation. The computer has also found a role in drafting and project management. The task facing engineers at present is to further extend the use of the computer in all aspects of design, project management and construction and to then link all applications within a common framework.

An integrated design system for structural engineering is a package of the many applications programs that perform separate but related tasks in structural engineering. Information produced by one application may be required for use in another; a necessary element

of the integrated system, therefore, is the transfer of information between or among the many applications. One means of accomplishing this information transfer uses an integrated computer database as a general information pool from which required data can be drawn and into which new information can be added.

Why have an integrated system for structural design? To increase the efficiency and productivity of the design office, and to design better and safer structures. Engineering is a competitive business in which increased efficiency and productivity brings material advantage. Engineering basically involves computation, something that computers do with great efficiency. The computer not only offers speed, but accuracy and thoroughness as well; not only can design progress quickly in an environment that places a premium on time, but greater confidence can be placed in the design itself.

Why have a database? Because an integrated design system cannot operate without one. In order to operate effectively, the computer system must have direct access to the necessary data. The applications systems will be composed of many different programs. To manually input data for each of these programs would eliminate any efficiency of the system. The database also provides central control of data; an important consideration for large projects.

The questions that this thesis addresses are:

- 1) How should the database fit into the integrated design system?
- 2) What form should the database take?
- 3) How might the database be implemented in a design system?

These questions are considered with the view of creating a practical, usable design system, using available computer facilities (hardware and software).

1.1 Plan of the Work

Chapter 2 will discuss the possible ways that an integrated design system can be created and the role that the building structure database is called upon to play in such a system. The database will be required to store different types of data; what these are and the process of design itself will also be examined.

Chapter 3 will look at the different data models available in commercial database management systems and explain why the relational model is most suitable for the building structure database.

Chapter 4 will discuss the problems of data representation in the building structure database in general terms

Chapter 5 will present a general scheme for representing a simple class of structure, the diagonally braced steel industrial structure, in a relational database.

Chapter 6 will discuss the programming required in the design of the simple steel structure

Chapter 7 will present a description of the system that implemented by the author in a commercial relational database management system. This includes both the data files and the applications programming.

Chapter 8 will present a sample design using the building structure database and the design system.

Chapter 9 will present a scheme for data maintenance. It will also discuss some further aspects of using a building structure database and further work that might be done on similar lines to this thesis.

Chapter 10 will present the authors conclusions.

Appendix A contains a listing of the data file structures and the program listings.

Appendix B contains a brief summary of the dBase II relational DBMS commands for data creation and manipulation.

1.2 Previous Work

The use of computer databases for computer aided engineering is actively being developed in nearly all fields of engineering endeavour [15]. In the field of architectural design, several systems are presently operating [16, 17, 18]. It is to be expected that these systems can be developed to include structural engineering as well as architectural applications.

A good example of this type of computer aided building design system is CAEADS - Computer Aided Engineering and Architectural Design System—developed for the U.S. Army Corps of Engineers [18, 19]. This system consists of a central database and a body of individual applications programs. Its present facilities include programs for checking building habitability criteria, thermal analysis, cost analysis and financial feasibility. Further facilities, such as structural design, are being developed. The representation of buildings in CAEADS is based on definition of architectural entities that can further be defined in terms component entities [20]. The system begins with a general building conception and proceeds to define the building in greater detail.

In the Canadian context, mention should be made of "Taskmaster" [21]. This is a commercial system that makes use of a central independent database and existing applications programs relating to structural analysis and design. The database stores information required for or produced by the various applications. This information includes node data, member incidences, member loads, member forces, etc. Data is automatically formatted for the individual applications and the user interacts with the system rather than the individual programs.

Although this thesis does not build specifically on any existing system, it relies a great deal on concepts previously developed. The approach taken in this thesis, while representing general building description, as is done in CAEADS, differs in that building description proceeds from the point of view of the structural engineer and deals with basic structural, rather than architectural, entities.

An important summary and bibliography of work done in the use of databases in computer aided design is provided in reference [15], to which the author refers the reader for further information.

Chapter 2

The Role of the Building Structure Database

It is clear that the computer will come to play an increasingly important role in all aspects of building design, construction and project management. The fullest use of the power offered by the computer will require the development of a fully integrated system of design and management. The structural designer will require an automated system of building structure design; that is, an integrated design system for building structures, of which the building structure database will be an important part. It is therefore necessary to determine both the role that the building structure database is called upon to play in the integrated design system and the means of representing building structure data.

2.1 The Integrated Design System

Engineering is a rational activity; it follows a set of rules and procedures: these can be translated into computer programs. The body of engineering programs is growing steadily. This growth is being spurred by the advent of cheap computing power and the acceptance into the design office of small computers for routine numerical computations (most potential engineering applications do not require much computing power.) A presently attainable goal is the use of computers in all stages of structural design. If all aspects of design could be integrated into one design system the whole would be much greater than the sum of the parts. However, it is no easy task to reduce a body of knowledge as large and diverse as structural engineering to a system of computer programs; yet the power that such a system would give to the engineer demands that the task be accomplished.

There are several ways in which a body of programs could conceivably be integrated into a single design system. Three possible ways would be:

- 1) The creation of a comprehensive computer program package that would accept as input the basic data of building configuration, loading and design constraints; and return building design documents as output (Fig. 2-1.)
- 2) The linking of a series of existing programs with several data files or databases. The output from one program can be stored in a database from which input to another program can be extracted, thus creating a chain of programs linked by different databases (Fig. 2-2.)
- 3) The creation of a central database from which all program input can be extracted and into which program output can be stored (Fig. 2-3.)

Of these options the central database concept promises the greatest flexibility and adaptability.

Figure 2-1: An input/output design system.

The monolithic once-straight-through input/output model denies the engineer the facility for interactive control of the design process; all basic data must be known at the start and front-loaded into the system which carries through design uninterrupted. This system presents difficulties in making necessary revisions to the building structure design resulting from changes in the basic data. The program itself will be cumbersome and difficult to maintain. The system will also be difficult to expand to provide additional services.

Figure 2-2: A database linked design system.

Figure 2-3: A central database design system.

The database linked model has the advantage of using existing programs (if they do already exist.) It relies, however, on the strict compatability of the data files to the particular programs which they link. It also imposes a linear progress on the process of design. Design updating becomes difficult, requiring that at least part of the design process be repeated. The system is inflexible and presents problems to the programmer who would try to add on programs that do not fit the linear scheme.

The central database has the advantage of providing central control of all the building structure data; all data is stored in the database, which may be the sole authoritative data source. Within the database itself redundancy may be minimized and data consistency and integrity maintained.

The central database also provides data independance [1, 2, 3]. The various applications programs can be totally independant of the organisation of data in the database. For example, an existing program could be used with the database without changes. Database specific

software is used to create an input file in the format required by the program and, in a similar way, the program output file is converted into the database storage format. The program, it could be said, is unaware that it is working with a database; the program is totally independent of the database. A change in the database has no effect on the applications program, and conversely, a change in the program has no effect on the database. Database/program communication is filtered through small, easily maintained programs whose sole function is to convert data from the database format to the applications program format and vice versa. This system has clear advantages. The database can be modified or expanded without the need for alterations in the applications programs, and existing programs can be improved or new programs added without requiring changes to the database. The system offers both ease of maintenance and expansion.

The drawback of the central database is that it is difficult to implement. It lies conceptually at heart of the integrated design system and must store a great amount and variety of data. All this data must be organized so that it is easily accessible. The design of a database that can adequately perform this important task presents serious problems. The central database is the key to integrated design, as it will become the authoritative repository of building structure data; but much work must still be done to make it a practical reality.

2.2 The Building Structure Database

A database can be considered, quite simply, any meaningful collection of information. A building structure database is any organized collection of information pertaining to a building structure. Traditionally, a building structure database takes the form of design specifications, design notes and drawings. The database contains all the information pertinent to the design of a building structure and provides a means of both recording and communicating building structure information. This information—if it is to be used by a computer—must be in machine intelligible form. If it is stored in the traditional way, each use of the computer requires a translation of the information into machine code; the creation of an input file. As computers assume an increasingly greater role in building design it will become more important to reduce the time consuming and error prone manual input of data. This is achieved by storing information permanently in machine code, that is, in a computer database.

The role of the building structure database is to contain in an accessible form all the information required for, used in, and created by the process of design. The database, which is solely an information storage device, is independent of all uses made of it. As its principal aim is to describe a building structure, it can be considered a representation of that structure. To fully serve the full range of possible uses, the database must be a representation of the actual structure, as opposed to a representation of any model of that structure.

The database describes the actual structure with as much detail as possible. A particular application may require the creation of a simplified model; for example, matrix analysis requires a mathematical idealization of the structural frame. In such a case the model is derived from the description of the actual structure contained in the database. A complete and accurate description allows maximum utility for any conceivable use. This description, however, cannot be complete at all stages of design; at the beginning of design it will necessarily be incomplete, consisting of perhaps only the most basic structural elements. The database, however, always describes an actual structure with as much detail as it has available. As design progresses, more and more detail is added to the structure, and the description becomes increasingly refined.

Buildings are complicated assemblages of many different interconnected elements. A truly complete building description down to the last brick or screw is not desirable. In practice, a degree of simplification or generalization is necessary. But although not complete, the description can still be accurate. A building is an accretion of detail; as design (or construction) proceeds, increasingly finer details are added to the structure. It is possible to decide upon a degree of detail that would still convey a satisfactory description; sacrificing less important detail but still describing accurately what remains.

The building structure database completely and accurately describes the structure. Some information in that description will be of a general nature with wider application than to just one structure. Steel section properties, for instance, are required information for all structures. Such general information can be grouped apart from the building structure data in a reference database. As the design system becomes more general and encompasses a wider variety of structural systems or elements the greater is the need for reference databases. These databases will include information now found in product catalogues and design handbooks.

A number of structures together may form a project, having some information in common. A project database will store information pertaining to a whole project. Individual structure databases might be considered subsets of the project database.

2.3 Structure Data Types

As the data that is to be stored in the process of structural design is examined more closely, it is possible to discern a difference in data roles. It becomes evident that data can be separated neatly into two distinct types:

- 1) Structural Description;
- 2) Design Data.

Structural Description needs little elaboration; it is simply the information required to accurately describe the physical building structure. This information consists of the location of structural elements, how they are connected to each other, and their individual properties. This information is useful beyond the confines of structural engineering; all other disciplines involved in the design and construction of the total building system will also make use it.

Information that is necessary in the process of design but is not part of the structure description must also be stored; it can be termed design data. This information is primarily of interest only to the structural designer.

Design data can be separated into three parts:

- 1) Loading data;
- 2) Structure behaviour;
- 3) Design criteria.

Loading data consists of the location and magnitude of all loads supported by the structure. Loading data, though usually related to physical sources, cannot be considered as part of the structure description. Some loading, such as structure self-weight, is directly related to the structure description; in fact it is derived from the structure description, and so can be calculated by the applications programming. In general, loads are imposed upon the structure; they will vary from day to day and over the life of the structure. Several different

kinds of load will exist and act in various combinations, design practice requires that loads from different sources be treated differently. Loading data cannot therefore be treated as part of the building description.

Structure behaviour is the response of the building structure to loading. Structure behaviour includes all measurable behaviour such as member stresses and deflections. Behaviour not only varies with loading but also varies within or along a member. Structure behaviour is intimately related to both building description and loading, but as it is calculated data it must be treated differently.

Design criteria are those criteria upon which design decisions are based. These decisions usually involve selection of structural elements. Design criteria may be taken from design codes or may be established by the designer before or during the design process. Design criteria will be expressed as rules or constraints [4, 5]. It is important that design constraints be explicitly recorded in the database, completely separate from the applications programming. This allows flexibility in assigning and changing constraints without revising the applications software.

There is therefore a need to represent several different types of data in the building structure database, and each data type will have its own needs for representation. Each type will require a different manner of data organization and, perhaps, management. The building structure database is really a package of related databases.

2.3.1 Traditional Representation

A building structure database is any organized collection of building structure data. Although means of storing this data may vary, the information stored will be the same. The separation of structure data types therefore applies to building structure data in any data storage device.

The traditional way of recording and communicating building structure information is by written specifications, design notes and drawings. Each of these "devices" stores a different type of structure data in a different way.

Drawings, the tangible product of the structural design office, contain the building structure description. Graphical information is usually more easily understood than either written or numerical information.

Specifications, which include standard and office design codes, are a record of design criteria. These are usually in the form of text that contains some illustrative drawings and mathematical formulas.

Design notes combine many functions. They record the calculations that determine structure behaviour, such as member bending moments and deflections. They also record loading data, collected from various sources, and record the application of design criteria to member selection. Design notes combine graphics and mathematical calculations with sparse written comments.

The traditional means of storing building structure information show great variety. Information is stored as text, as numbers or as drawings. This variety of expression reflects the variety in data type. It can be expected the organization for storing this information in machine accessible form will also have differences. Form follows function; and there are definite functional differences among the data types.

2.4 The Design Process

The division of data into different types—structure description, loading data, structure behavior and design criteria—involved some assumptions about the process of design. Structural design is often not so much an act of creation as it is an act of selection; the selection of structural systems or individual structural elements; and selection is based on criteria. The basic criterion—that members must support the applied load without failing—involves knowledge of structure behaviour. Other criteria, such as maintenance of minimum clearances, involve building description alone.

The total process of design encompasses many computational and decision making processes. The design of any structure is bounded by rules or constraints. Decisions must be made on structure modelling, assigning design loads, evaluating structural options, etc. It is useful to separate the decision making processes of design from the purely computational.

At an elementary level, design can be seen as consisting of

- 1) Analysis,
- 2) Selection

Analysis is the application of mathematically derived or empirical formulas or algorithms to determine structure behaviour. Matrix analysis programs, for example, provide information on member forces and deflections. This information is necessary but not sufficient for the selection of member properties. Analysis does not involve decision making; a structure subjected to a certain loading can behave in only one way, and its behaviour can be determined. Decisions as to the suitability of a member property belong to an altogether different process

Selection is the process of evaluating criteria with the aim of obtaining a satisfactory structural design. In the case of selecting a beam section, each beam section is evaluated against the set of design criteria and the section which best satisfies the criteria is selected. Because selection requires the knowledge of structure behavior, analysis is the necessary first step in the design process. Analysis itself, however, cannot proceed without a structure description. This is somewhat of a paradox. In some cases it is possible to start with a less than full description and determine some behaviour. The analysis of a simply supported beam, for example, will determine some behaviour without having to know all the physical properties of the beam. In the general case, however, member properties must be assumed for analysis. Behaviour is afterwards checked against design criteria. If design criteria are not satisfied a different set of member properties must be tried, with the benefit of insight gained from the previous trial. Some design must proceed in this way; by a process of iteration. Fortunately, this is a process which is suited to computer methods.

The design process uses and creates data. Each data type has its own place in the design process. The process starts with building description and loading data. From these, analysis determines structure behaviour. Selection evaluates structure behaviour against design criteria to determine building description. Analysis and design are a means of improving the structure description until a desired degree of description is achieved. Structure description is both a starting and an end point. If the concept of the central database is to be applied,

additions or modifications to the description as a result of the design process are made to the original data description (Fig 2-4)

Figure 2-4: The design process.

The process of design begins with a building description and loading. Building configuration and loading are usually not determined by, but presented to, the structural designer. The function of a building structure is to enclose space and support load. These requirements are determined by other disciplines. The structural designer works within narrow limits. The choice of structural systems and individual members must satisfy criteria other than those of purely structural concern.

Integrated design is not necessarily the same as "automatic" design. In automatic design the computer performs design without the need for human interaction; that is, automatically. It would not be provident, however, to dispense with the experience and skill of the human designer. A good integrated design system should make provision for the designer to interact with the computer system, to direct the progress of design and to make decisions involving engineering judgement. The system should act as a tool in the designer's hands; to aid the designer by performing computations and making routine decisions, freeing the designer to devote his energies to the creative aspects of design.

Chapter 3

Data Models

The role of the building structure database is to store all the information pertaining to a building structure in such a way that it can be readily retrieved. This data must, of course, be stored on some physical device. Manipulation of data in terms of physical storage, as is currently done in most programming languages is, to say the least, an inconvenience. It is really necessary to have a more sophisticated means of manipulating data; a higher level of data organization. This higher level of organization is provided by the Database Management System (DBMS.)

The Database Management System (DBMS) maintains an apparent organization of the stored data. The user views the data only as presented by the DBMS. When the user makes a change to the data in the apparent organization the DBMS makes the correct changes to the physical storage to maintain consistency. The actual method of physical storage is not of direct concern to the DBMS user (Fig. 3-1.)

Figure 3-1: The DBMS manipulates data in physical storage.

The user communicates with the DBMS by means of special languages. Database content is defined by means of a Data Definition Language, which describes the kinds of data that

are to be stored. Data is manipulated by means of a Data Manipulation Language. The user, with these languages, can set up the DBMS to store and manipulate data.

There are several ways in which data can be organized by a DBMS. The three principal organization models [1, 3] are:

- 1) Hierarchical Model;
- 2) Network Model;
- 3) Relational Model.

While each of these models may be used to store the same information the suitability of each model and the ease of data manipulation will differ, depending on the data. The database stores information that has meaning in the real world; that is, it models the real world. This modelling entails both storage of discrete data and relationships between data. The Building structure database must handle many types of data, and express complicated relationships among data. Since the DBMS is expected to accommodate a variety of structures, it must also be flexible. There must be an easy way of creating new categories of data and new relationships. The model must also present a clearly understandable organization of data. If the user cannot easily interpret the data as presented by the DBMS, there will be occasion for confusion and error. The DBMS must also support changes in data and relationships, as the structure design is constantly updated.

3.1 The Hierarchical Model

The hierarchical model organizes data into hierarchies, sometimes called tree-structures (Fig. 3-2.) A record is a group of related data and is the basic unit of storage. Data relationships are expressed by relative positions of records within the hierarchy. In the simple hierarchical structure each position in the hierarchy may have only one superior although each superior may have many dependants. Access to a particular data is made by tracing the hierarchy down from the top. The hierarchical model may be adapted to represent other than purely hierarchical relationships, but remains most suitable when used for storing data which has hierarchical relationships in the real world.

Figure 3-2: A hierarchy, or tree-structure.

3.2 The Network Model

The network model allows more complicated relationships than the hierarchical model. Each record may have many superiors and many dependants. Relationships between records are established by another kind of record, a connector. Thus, a network of data records linked together by connectors is created (Fig. 3-3.) Complicated relationships may be modelled, as one-to-many and many-to-many relationships are permitted. Records are accessed by tracing paths through the connecting links. Data manipulation requires an understanding of the network structure, which may become complicated.

Figure 3-3: A network.

3.3 The Relational Model

The relational model stores data in the form of flat tables (cf. Fig. 3-4). Each column within the table is a data field type. Each row is a data record instance. In the relational model relationships are treated exactly the same as all other data, and stored in tables. There is absolutely no restriction on the types of relationships permitted; one-to-many or many-to-many relationships are handled with equal ease. The relational model allows manipulation of entire tables, owing to the mathematical properties of relations, and also permits ready access to individual data records. The data, as organized in tables, is easily comprehended, this allows for effective use of the database and efficient programming.

3.4 Choice of Relational Model

The relational model offers great advantages over both the hierarchical and network models for the building structure database. The major advantage of the relational model is its basic simplicity and underlying clarity [5]; the organization of data in tables is logical and easily understood. The relational model handles complicated data relationships easily, expressing relationships in the same way as other data. Also, the mathematical properties of relations permit easy manipulation of data. The primary disadvantage of the relational model relative to the other models is the lower efficiency of accessing [3].

The hierarchical model is unsatisfactory because the building structure database must represent complicated relationships among data. This data is not naturally of a hierarchical nature, and so would require a forced fit into a hierarchical organization. The organization of data would itself be a hindrance to effective programming.

The necessity of storing complicated relationships also argues against the use of the network model, in which these relationships must be represented by links. The network required for a building structure would itself be complicated, and manipulation of data would require that these links be fully understood by the user. The network model would be cumbersome to use and difficult to maintain.

The relational model, although the most recent of the data models to be implemented in commercial DBMS, appears to be the winner in the contest for the most suitable data

model. It has an organization that is clear and logical, it can express complex relationships, it offers ease of data manipulation, and it is flexible. These are the requirements of a building structure database.

3.5 Relations and Relational Operations

This section gives a brief introduction to relations and their operations. For a more complete treatment see references [1, 6, 7].

The relational data model is based on the mathematics of relations. A relation can be viewed, in essence, as a two-dimensional table consisting of rows and columns. Columns are data fields and rows are stored records. The column headings (data field names) are known as "attributes." The rows are called "tuples."

A relation is a set of tuples that have the same attributes. The ordering of the tuples within the relation is irrelevant, and each tuple must be unique; there are no duplicate tuples. The order of the columns is also unimportant. Each data field must contain only one discrete data value. As each tuple is unique, every tuple can be individually identified. However, a relation usually consists of both independent and dependant data. An example of a simple relation would be a table consisting of the attributes BEAM NAME, LENGTH, and SECTION (Fig. 3-4.) In the case of this relation, the attribute BEAM NAME consists of unique data that will exist in no other tuple and so acts as a "key." Any specific tuple can be identified by its key. To specify a single tuple may require the knowledge of more than one attribute, or perhaps all attributes. Therefore a key may consist of more than one attribute.

TABLE 1

BEAM NAME	LENGTH	SECTION
B-1	3.0	C200×17
B-3	7.0	W410×54
B-2	5.0	W200×31
B-4	5.0	W360×33
B-5	7.5	W410×54

Figure 3-4: An example beam relation.

The mathematics of relations provides some operations that can be performed on relations. The three principal operations are SELECT, PROJECT and JOIN (these operations are sometimes called by different names)

The operation SELECT creates a subset of a relation based on the qualification of an attribute. For example, the operation is performed on the example beam relation: "SELECT TABLE 1 for LENGTH is less than 7.0." This results in another relation, TABLE 2, in which only the tuples that have a value of LENGTH less than 7.0 appear (Fig. 3-5)

TABLE 2

BEAM NAME	LENGTH	SECTION
B-1	3.0	C200×17
B-2	5.0	W200×31
B-4	5.0	W360×33

Figure 3-5: "SELECT TABLE 1 for LENGTH < 7.0."

The operation PROJECT results in a relation similar to the original relation in which only specified attributes are included. For example, the operation is performed on the example beam relation: "PROJECT TABLE 1 over BEAM NAME, SECTION." The result is the relation TABLE 3, consisting only of the attributes BEAM NAME and SECTION (Fig. 3-6.) Any duplicate tuples resulting from this operation would be eliminated.

TABLE 3

BEAM NAME	SECTION
B-1	C200×17
B-3	W410×54
B-2	W200×31
B-4	W360×33
B-5	W410×54

Figure 3-6: "PROJECT TABLE 1 over BEAM NAME, SECTION."

The operation JOIN is performed on two relations. Two relations are joined together over a common attribute. Wherever the value of the common attribute in the second relation matches the value of the same attribute in the first relation the tuples are combined. Using

the example beam relation, desiring to join to it a Section Relation, TABLE 4 (Fig. 3-7), consisting of the attributes SECTION, DEPTH and S_{xx} , the operation is "JOIN TABLE 4 to TABLE 1 over SECTION." This results in the relation TABLE 5, consisting of five attributes: BEAM NAME, LENGTH, SECTION, S_{xx} and DEPTH (Fig. 3-8.)

TABLE 4

SECTION	DEPTH	S_{xx}
C200×17	203	133
W200×27	207	249
W200×31	210	299
W410×54	403	924
W360×33	349	474
W410×39	399	634

Figure 3-7: An example section relation.

TABLE 5

BEAM NAME	LENGTH	SECTION	DEPTH	S_{xx}
B-1	3.0	C200×17	203	133
B-3	7.0	W410×54	403	924
B-2	5.0	W200×31	210	299
B-4	5.0	W360×33	349	474
B-5	7.5	W410×54	403	924

Figure 3-8: "JOIN TABLE 1 to TABLE 4 over SECTION."

From this table, a relation consisting of only beam name and depth could be produced by the operation: "PROJECT TABLE 5 over BEAM NAME, DEPTH." The result would be TABLE 6 (Fig. 3-9.)

The mathematics of relations provides useful operations for manipulating relations, but individual records or data fields can also be manipulated easily. Specific records are quickly located by specifying the data file (table) and a key. A specific data field is located by specifying an attribute. For example, to find the length of beam "B-1" in the example beam relation, the operation "SELECT TABLE 1 for BEAM NAME equals 'B-1'" gives the unique

TABLE 6

BEAM NAME	DEPTH
B-1	203
B-3	403
B-2	210
B-4	349
B-5	403

Figure 3-9: "PROJECT TABLE 5 over BEAM NAME, DEPTH."

tuple. The value of the attribute LENGTH can then be enquired. The relational model provides a very logical organization for storing the most complex data combined with an easy and powerful means of manipulation.

Chapter 4

Data Representation

Several types of data must be stored in the process of designing a building structure. This data can be divided into four types:

- 1) Building structure description;
- 2) Loading data;
- 3) Structure behaviour;
- 4) Design criteria.

The most important data is building description. Building description is needed in all stages of design and construction of the building structure, and is also needed by all other disciplines involved in the design and construction of the total building system. Equally important to the structural designer is the loading data. With building description and loading data structure behaviour can be determined and member selection based on design criteria can be performed.

4.1 Representation of the Building Structure

The building structure database must describe a physical building structure. It is necessary to recognize the problems in trying to represent any physical reality in a database [8]. In the first place, our perception of the reality is necessarily incomplete. From our perception we construct a conceptual model in which we make further simplifications. When the conceptual model is to be contained in a database, we must describe it in a form the computer can use; this usually introduces more simplifications. These simplifications make it possible to understand and predict behaviour, and to record and communicate relevant information.

The danger is in over simplifying. If the maximum use is to be made of the building structure database it must be as complete and accurate a description as possible. The database must contain information which pertains directly to the real structure and avoid as much as possible any simplified modelling of the structure.

In reality, building structures are composed of connected individual elements. This is very clear in steel structures. Steel members are individual pieces which are connected through physical connections. Other structural elements such as wall cladding are connected to the structural framing members. Although a concrete structure is monolithic with beams, columns and slabs continuous, the various members can still be differentiated, and a description in terms of discrete members and connections is still possible. Therefore, an accurate description of a real structure can be created in terms of discrete elements.

Since a building structure is composed of recognizable individual elements it is possible to accurately describe a structure by recording the location of the individual elements, their connections to other elements and a description of the elements themselves. The location of elements can be called the "geometry" of the structure. The connectivity of elements can be called "topology."

The representation of a building structure in numerical form that is most familiar to civil engineers is that which is used in the common matrix analysis programs. The structure is represented as straight line "members" connected to "nodes" which are fixed in location (Fig. 4-1.) Structure geometry is provided by the location of the nodes; structure topology by the member connections. This representation of the structure is determined by the nature of the analysis program. It is a simplified model of the real structure created only with regard to analysis. As it is now used, it only describes a mathematical model of the structural frame; other structural elements are not required for the analysis. If this method were to be used to describe the entire building structure it would have to be further developed to describe geometry and topology of a great many kinds of structural elements, not just beams and columns.

If the attempt to use a node-and-member representation to describe a complete building structure is made, several problems arise. Each connection requires the existence of a node since members are connected to nodes rather than to other members. There must be a means

of indicating that a member is continuous through a node (this is not usually important in matrix analysis) There must also be a way of representing plate elements such as floor slabs and wall cladding.

Figure 4-1: Node and member representation.

The major argument against the node-and-member representation is simply that the nodes and members do not give a true picture of the real structure. If a node were to be specified at some point on a column, the part of the column below the node would be one member, the part above the node another member; but the real column might be a single member with no actual physical presence of a joint. This representation introduces an unnecessary and confusing level of abstraction to the structure description, which may be useful in a specific application such as matrix analysis but detracts from the full use of the database that would eventually be made.

A much simpler description can be made by disregarding the requirements of any particular application and merely recording real member location and connectivity. To describe a simple frame is a simple matter:

- 1) record location of columns relative to a fixed point and column orientation;
- 2) record the elevation at which a beam connects to a column;

- 3) record the location at which a beam connects to another beam;
- 4) record the physical properties of the various members.

Although simple, such a description accurately represents the real structure. Structural elements are considered as three-dimensional solid objects. The location of elements is given with reference to some point on or within the element. A column location, for example, would be given with reference to the column centerline. For a beam, vertical location would be given with reference to top of beam, and horizontal location would be given with reference to the beam centerline. All other kinds of members could be similarly located. The dimensions of the members themselves will not be known at all stages of design, but the members are always regarded as having finite dimensions even if unknown.

Of course a structure is composed of a much wider variety of elements, but each has a specific location and specific connections. It will always be possible to first describe an element, locate the position of connection points on the element itself and then locate these same connection points on the supporting element. Of course, this representation does little to simplify the structural description, but simplification is not the role of the structure database.

There is absolutely no requirement that the data contained in the building structure database be immediately usable for any application. The only requirement is that the description be complete and accurate. The data contained in a complete and accurate description can be manipulated to provide the wanted information in the desired format. The representation in the database itself is completely independent of any application.

Once attention is given to only describing the structure as accurately as possible without regard to possible applications, many problems are resolved. The location of column splices for instance might pose a problem if representation was in the form of nodes and members. Splices are usually taken to be at a floor level for matrix analysis, when in fact they are positioned above the floor beam connections. Using a natural description, it is only necessary to define the location of the splice. The columns sections above and below the splice would be treated as separate members, which they are. The location of the splice would be approximated as occurring at floor level only within the process of analysis. Other

possible problems can similarly be resolved by separating the function of description from the requirements of design

In effect, the actual organization of data within the database is unimportant. As long as the description itself is complete and accurate it can be converted to any other form. This conversion will not necessarily be easy but it will always be possible. In all cases the database must only describe the structure as it actually is.

There is also a question of member differentiation by function. The same steel section may equally serve as a column or a beam. Is it necessary to describe a member as beam or column? In this case function is really a matter of location. Because a member is vertically aligned it is a column; a horizontal member is a beam. There is no difference in the real members and there should therefore be no differences made in the database description. It is totally sufficient to record location, treating all steel members the same. If at some time there is a need to label a member as a beam or column, some applications program could be set up to evaluate the criteria that determines beam-ness or column-ness, and attach the necessary labels based on the database description; similarly for all members.

4.2 Topology vs Geometry

Each part of a structure is connected in some way to some other part of the structure. Description consists of member location (geometry), member connectivity (topology) and description of the member itself. Geometry and topology are related and it may be redundant to include both in a structure description.

If geometry alone is stored, the location of all members is known. If it is assumed that members in very close proximity are connected the topology of the structure can be deduced. It would then be possible to store only structure geometry, not only saving storage space that would otherwise be devoted to topology but also eliminating redundancy in the database, thus making it easier to maintain database consistency. Since topology is not stored it will have to be re-calculated whenever topological information is required. Calculating topology from geometry requires searching for proximity. In a large structure where there are many members this process would require considerable computing power.

Even if topology is not stored, other data relating to connections must be. A connection, for example, may be a simple shear connection or it may be moment resistant. There will also occur cases where members that are in close proximity are not connected; provision for this possibility must be made.

It might also be possible to store topology rather than geometry, to eliminate any redundancy. A structure cannot, however, be described purely by topology. Topology gives the connectivity of members but does not provide member location or structure dimensions. But if only enough geometry data is included in the description to dispel ambiguity, no redundant data needs to be stored. If the location of an individual member is given only in relation to its supporting member, no redundancy is created. Given only the location of members relative to their supports, their absolute location relative to a reference point can be quickly calculated without the need for any searching, making little demand on computing power.

Redundancy resulting from the storage of both the geometry and topology of a structure can easily be eliminated by storing topology and *relative* geometry only. *Absolute* geometry can be generated quickly and easily as needed.

The other important matter is the amount of data to be stored. Geometry alone is more compact, but is arguably so only at the cost of some ambiguity. It is not possible to totally disregard connectivity.

There are two benefits that result from eliminating either geometry or topology from the structure description:

- 1) elimination of redundancy;
- 2) reduction of the amount of storage.

Of these, the reduction of storage is much less important than the elimination of redundancy. As the topological representation provides an adequate description with low computational demands and only marginally greater storage requirements, there seems little reason to adopt the more difficult geometry representation. In the balance, the topological representation is the better alternative.

Chapter 5

The Building Structure Database

The relational data model gives the structural designer an efficient organization concept for the storage of building structure data. It offers ease of data creation and data manipulation, as well as the possibility of easily expanding the database to include new levels of description.

The most important data to be stored in the building structure database is the building description. A building structure consists of connected discrete elements. A building description must contain:

- 1) a description of the individual component elements, and
- 2) their location and manner of connection.

Exactly how this information is recorded in the database is not critical as long as the description is complete and accurate.

There is however a difficulty in using the database description directly in an analysis program. In analysis engineers usually deal with a mathematical idealization of a structure, a model. Simplifying assumptions made in the creation of the model are often a matter of judgement. This does not mean that modelling is not programmable—engineering judgement is not arbitrary and could be reduced to a set of rules to be processed by the computer. The difficulty of creating the model, however, concerns only the applications programming, not the database.

The aim of this thesis is to create a usable database and to use it for design. This has meant that the rules for data representation as previously discussed could not be followed rigorously. To create a database to describe a general structure was not within the limits of time and resources available. In order to create a working database it was necessary to limit the scope of the problem. The development of a detailed description for all structures

in general has not been attempted, nor has the description of any complex structure in particular. To be practical, the database has been limited to describe one common class of structure that can be easily described and straight-forwardly designed.

5.1 The Structure Description

The class of structure this thesis will consider is the multi-storey diagonally braced steel industrial structure. This type of structure can be generalized as consisting of simply supported beams, pin-ended columns and diagonal bracing; this is the skeletal structure. The skin consists of floor or roof slab, wall cladding, girts and tie-rods. The design of this type of structure makes low demands on computational power; individual elements can be designed without iteration.

Of course any steel industrial structure will have many other types of elements: beams continuous over supports, cantilever beams, hangers, posts, trusses, stairways, etc.; the list goes on. Each of these additional elements may be fitted into the database in future without greatly altering the existing basic description.

This again opens the question of member classification. A wide-flanged steel section might equally well serve as a beam, column or a bracing member and might be treated differently in each case. As function is solely a matter of location and location is also stored in the database, member classification is redundant; therefore unnecessary and undesirable.

In this database however it was found necessary, given the limited computer resources, to retain member classification in order to simplify the database description. The classification of members, of course, restricts the database to a description of only those members that fit a pre-defined classification, but it does not introduce any added redundancy and the description can still be accurate. For, just as location dictates function, so location can be deduced from function. If columns are assumed to be vertical members, it is necessary only to record horizontal location to accurately position the member. The same applies for a beam taken to lie in the plane of a floor level. Although a description with member classification would be unsatisfactory in the general case, it is still a valid use of the building structure database as it has been outlined. This can be considered only a variation on the theme. The description is still accurate with no added redundancy.

For the diagonally braced steel industrial structure the basic scheme of description is as follows

- 1) location of column lines,
- 2) location of columns on column line intersections;
- 3) elevation of floor levels;
- 4) location of floor and roof beams;
- 5) location of floor slab and roof cladding,
- 6) location of diagonal braces,
- 7) location of girts,
- 8) location of tie-rods,
- 9) location of wall cladding.

The location and connection data for these elements describes the basic building structure. The data organization is concise and efficient and provides a clear description of the actual structure. It is necessary only to store this information in the database

All information in the database is stored in the form of tables called relations. The relations that correspond to the above description scheme can be created as follows:

• AXES, COLINES

The first relation, called AXES, records the possible orientations of column lines, relative to a reference direction; in effect this relation defines a set of cartesian coordinate systems (Fig. 5-1.) Each possible column line orientation is given a name attribute, NAME, and a rotation in degrees clockwise from the reference direction, ROTATION. Each column line orientation must be one of a pair, its partner lying at right angles. The name of the orthogonal orientation is the attribute ORTHO.

The location of column lines is stored in the relation COLINES. Each column line has a name, NAME, an orientation, ORIEN, and a right angle distance to a reference point, DIST. The orientation is defined in the AXES relation.

It was previously asserted that the building description should include the description of only physical things. It is questionable whether column lines belong in this description. I include column lines because it is presently unthinkable to describe a structure without them (e.g., description in structural drawings) and they provide a handy reference for the human designer. The computer can get along quite well without column lines as long as column locations are otherwise defined. Their inclusion, serving only as part of the column location data, does not compromise any basic principles.

Figure 5-1: Location of axes, column lines and columns.

• COLUMNS

Column locations are given in the relation COLUMNS. Columns lie on the intersection of orthogonal column lines (cf. Fig. 5-1.) Two column lines determine the location of a column. Each column has an orientation of its strong axis (as defined in the relation AXES). Each column will also have a top and bottom elevation. The bottom of a column is given as the elevation of the bottom of its baseplate. The top is determined by the elevation of the roof.

level. The relation COLUMNS therefore consists of the attributes NAME, LINE1, LINE2, ORIEN, BOT COL and TOP COL.

• LEVELS

Beams are conveniently grouped by floor level. Floor level elevations are given in the relation LEVELS. This relation consists of the level name, NAME, and its top of steel elevation, TOS.EL. Sloping floors could be described in this relation by the inclusion of some additional attributes that would fix the elevation of one point on a roof section and give the degrees of slope in right angle directions.

• BEAMS

Beams at each level belong to a separate relation. The relation for beams at Level 01, for example, is called BEAMS01. All beams are simply supported, that is, supported only at their ends. Each beam has two ends; each end may connect either to a column or to another beam. The information required to describe a connection is the name of the supporting member and the location of the connection on that member; the connection elevation is given by the floor level elevation. If the end connects to a column the location in the horizontal plane is determined by the column location. If the end connects to a beam the location is given as the position of the connection along the supporting beam (Fig. 5-2.) This position is measured from one end of the supporting beam. It is therefore necessary to label the beam ends to know from which end the measurement is made. The rule that is followed in this regard is: the beam end that has the minimum principal horizontal cartesian coordinate is labeled END1; if both ends have the same principal coordinate the end with the least secondary coordinate is END1. The need to label beam ends is an inconvenience, but this duty could be performed by the computer, the form of the database in no way dictates the form of the user/system interaction.

The attributes in the beam relation are: NAME, CONTYPE1, CONNAME1, POSITN1, CONTYPE2, CONNAME2, POSITN2, SECTION. CONTYPE can be either "B" for a beam or "C" for a column. SECTION is the name of the beam steel section, which will be the result of design.

Figure 5-2: Beam ends are located along their supporting members.

• PANELS

Floor slabs are continuous over a part or the whole of a floor level and should be so described; but because of the difficulty of translating such a description into a model that could be used directly in a simple design program, liberties have been taken with the floor panel description. Floor slabs are described as simply supported panels which span between a pair of beams. This is an unsatisfactory description for the design of the floor slab but presents a simple way of calculating tributary loads on the supporting beams. Such a description may seem misleading, but if it is assumed that adjacent panels are continuous, it will not necessarily be inaccurate.

Floor panels are grouped, as the beams are, by floor level. The relation PANELS01 describes the panels lying on Level 01. Panels are rectangular flat elements of finite dimensions. Panels are simply supported by two beams. Panel location is determined by the name of the supporting beams and the location of the panel corners along those beams (Fig. 5-3.) Corner positions are measured in the same way as beam connection positions; relative to the ends of the supporting beams. The panel relation consists of the attributes: NAME, JOIST1, POSITN1A, POSITN1B, JOIST2, POSITN2A, POSITN2B.

Figure 5-3: Panels are defined by the location of their corners.

- BRACING

The location of the diagonal braces is contained in the relation BRACING. A brace has two ends that connect to columns at floor elevations. To describe a brace therefore requires the attributes NAME, COLUMN1, LEVEL1, COLUMN2, LEVEL2 and SECTION. This description applies only to full diagonal X-bracing, the most common kind.

- GIRTS

The description of the girts is given in the relation GIRTS. Girts are horizontal members that span between columns at given elevations (Fig. 5-4.) Girt description will therefore consist of NAME, COLUMN1, COLUMN2, ELEVATION and SECTION.

- TIERODS

The description of tie-rods is given in the relation TIERODS. Tie-rods transfer wall cladding dead load to the top of the structure where it is taken by the columns (cf. Fig. 5-4.) Each tie-rod will have the attributes NAME, COLUMN1, COLUMN2, POSITN. The attribute POSITN is the distance that the tie-rod is from COLUMN1 towards COLUMN2. It is implicitly understood that a tie-rod supports each girt that it intersects, that is, all the girts that are supported by COLUMN1 and COLUMN2.

- WPANELS

The description of the wall cladding is given in the relation WPANELS. Wall cladding

Figure 5-4: Girts and tie-rods.

is assumed to be of a standard type throughout the structure and therefore no member properties are included in the description of any individual panel section. Wall panels are taken as being simply supported along the full length of their supporting girts. There is no provision for unclad wall areas; the structure is fully enclosed. To locate a wall panel it is sufficient to name the two supporting girts; the attributes would be NAME, GIRT1 and GIRT2.

5.2 Loading Data

Two types of load need to be considered:

- 1) concentrated loads;
- 2) uniformly distributed loads.

Concentrated loads will be located only on beams. Distributed loads will be located on panels. The location of a concentrated load is determined by the name of the supporting beam and the position of the load along the beam. If a uniform load is assumed to be constant across the width of a panel, its location is determined by the name of the supporting panel and two positions along the length of the panel; the two positions define the loaded portion of the panel.

Inclusion of the attribute LOADCASE in the loading data files is important because Limit States Design requires that load factors be applied to loads according to the type of load, whether dead load, live load, wind load, etc.

The main structure loading files are:

- CONCL

The relation CONCL01 describes the values and locations of concentrated loads on Level 01. Each load has a loadcase name. The attributes of the concentrated load relation are NAME (name of supporting beam), LOADCASE, LOAD and POSITN. The inclusion of the source of the concentrated loading can also be recorded by type and name, SOURCET and SOURCEN.

- PLX

The relation PLX01 describes the values and positions of uniformly distributed loads on Level 01. Each load is also given a loadcase name. During analysis load factors will be applied to the loads according to their loadcase. The attributes of the uniform load relation are: NAME (name of the supporting panel), LOADCASE, LOAD, POSITN1 and POSITN2.

- WPLX

Horizontal loads originate with the wall cladding which also has vertical dead load. The main horizontal loads to be considered are wind load and earthquake load. Earthquake load is basically generated from the building description so no additional loading data is required. Wind load however is a function of environmental factors and building location.

Wind load is applied to the wall cladding. It can be assumed to be uniformly distributed with step-wise increases at certain elevations. The wind load description contained in the relation WPLX will consist of the attributes WPANEL, LOADCASE and LOAD. WPANEL is the name of the wall panel that is being loaded. The load is assumed to be constant along the full length and width of the wall panel.

Vertical wall cladding load consists only of the dead load of the wall cladding itself and it can be treated in a way similar to the horizontal load. The vertical load relation will consist of WPANEL, LOADCASE and LOAD. As with the horizontal load the vertical load

is uniformly distributed over the whole wall panel. Because the relations for horizontal and vertical wall panel loadings are essentially the same, they can be combined in a single relation with the addition of a single attribute to tell whether the load is horizontal or vertical. The attribute DIRECTN will have one of two values: "H" for horizontal or "V" for vertical.

The horizontal wall panel loads are taken by the girts, which transfer them to the columns. Vertical wall panel loads are also taken by the girts and transferred to both the tie-rods and the columns. The tie-rod loads are picked up at the top of the columns.

Once the basic data of building description and loading have been entered into the database this information can be used to determine structure behaviour that is necessary for the selection of required member properties.

5.2.1 Automatic Calculation of Loading Data

It is possible to reduce the amount of loading data manually input by creating programs that will calculate snow loads and wind loads given the geographical location of the building structure and some local environmental factors. Loading that comes from fixed mechanical equipment can also be automatically applied, given a description of the piece of equipment (information that should already be stored in some other database and can be electronically transferred) and its location in the structure.

5.3 Structure Behaviour Data

All structure behaviour is derived from the building structure description and loading data. There is no manual entry of structure behaviour data. The structure behaviour that is of most interest to the structural designer are member end reactions, shear forces and bending moments. Deflections, though they must always be checked, are not usually the critical factor in member selection.

Before member forces can be calculated the individual member loadings must be known. The loading data files give the values and locations only of applied loads, they do not give loading that results from member reactions. Panels do not support other members but both beams and columns do, and they will require member loading files.

The structure behaviour files are

- BLX

A BLX file exists for each floor level. The file for Level 01, for example, is BLX01. This file contains the location of the external forces exerted on the beam by applied loads or member reactions, whether it is a concentrated or uniform load and the loadcase. The attributes are: NAME, LOAD, LOADTYPE, LOADCASE, POSITN1 and POSITN2. LOADTYPE can be either "C" for a concentrated load or "U" for a distributed load. The attributes POSITN1 and POSITN2 locate the load on the beam; for a concentrated load, only POSITN1 is used. Further attributes that can be added indicate the source of the loading; these are SOURCET and SOURCEN for source type (panel or beam) and source name. These are necessary if duplicate records are to be avoided, as strict accordance with the rules of relational mathematics dictates.

- CLX

Column loading is stored in the file CLX:—The file contains the value of the load and the loadcase, location not being required. The attributes of the relation are: NAME, LOAD, LOADTYPE, LOADCASE, SOURCET, SOURCEN and SOURCEC. The last three attributes are source type, name and level. This is the same as the loading file for beams with the only difference being that the load is located by floor level elevation.

In calculating member loadings it is important to maintain the proper loadcase for each load to ensure that the proper load factors are applied and the desired load combinations correctly analysed.

- BMDIAG

Once the individual member loadings are determined the internal member forces can be calculated. For the beams which are not part of the braced frame axial load plays no part in the design and the designer only requires the shear force and bending moment diagrams for the beam. These beam diagrams are calculated and stored in a BMDIAG file for each level. This file stores numerically the value of the beam diagrams at critical points. The beam diagrams will be different for every load combination. Of course, in the general case, all forces for each member must be stored.

5.4 Design Criteria

The storage and handling of design criteria presents serious difficulties to the creator of a structural design system. It is not obvious how design criteria can be organized in the building structure database. Actually, it shouldn't matter how the criteria are organized as long as the information is in the database and accessible. There must, however, be an organization—exactly what form it should take is a matter of future investigation.

One design criteria, the factoring of loads for Limit States Design, does fit nicely in the relational model. Load factors are related to the type of load, being live load, dead load, wind and earthquake load, etc. There also exists a relation with load combination factors. A relation, CASES, relating load combination, load case, load factor and combination factor is necessary for the analysis of any structure. The attributes would be LOADCOMB, LOADCASE and FACTOR. FACTOR would be the product of the load and combination factor, for each loadcase in each load combination.

Design criteria involved in the selection of member sections can be embedded in the programming. This however should be only a temporary measure and generally regarded as unsatisfactory. Selection criteria should be stored in the database and the appropriate applications programming written to make use of it that way. If this database scheme were to be further developed the storage and handling of design criteria should be a foremost concern.

5.5 Reference Databases

The most important reference database for use in design of steel structures is the steel section properties table. This table is required for the selection of member sections. The section properties must satisfy the design criteria. In a more extensive design system other reference databases defining the properties of other materials would also be needed; the properties of concrete, reinforcing bars, fasteners, various types of cladding, etc. Other reference data that might be stored in reference databases are environmental data relating to geographical location. All data that would otherwise be taken from a design handbook or product catalog would properly be stored in reference databases.

Chapter 6

Programming Procedures

This chapter deals with the programming procedures required for the design of a simple class of structure, unfortunately, it has not been possible to implement all the desirable procedures discussed due to the limited resources. The actual implemetations are presented in Chapter 7.

The purpose of the structural design system programs is to

- 1) determine structure behaviour from structure description and loading data, and
- 2) evaluate design criteria for member selection.

The building structure database describes a diagonally braced steel structure. The description contains enough detail to permit the design of the main frame elements of the structure; the columns, beams, bracing, girts and tie-rods. Of course, the structure as described is limited to a very small number of different type elements; the database has only a limited vocabulary. As the vocabulary of the database is upgraded it will be possible to perform more design tasks. The equivalent effect of some elements that are not in the database vocabulary can be entered in the database as applied loads. Stairways for example are not handled at the present time by the database, but the stairway loading on the supporting beams can be calculated and applied directly to those beams. Whether this is a suitable procedure in any particular case will of course be decided by the designer.

The main feature of the diagonally braced steel industrial structure is that it is designed to be determinate and therefore no iterative procedures are required in the analysis of the structure behaviour. As a generalization, in this type of structure columns are pin-ended, beams are simply supported and all horizontal forces are resisted by the diagonal bracing.

Both in consideration of structure description and design there were many reasons for choosing this type of structure to demonstrate the use of the building structure database.

First of all, steel structures are easier to describe than concrete or composite structures. Steel structures are usually composed of homogeneous members while concrete member properties vary with the distribution of reinforcing. In a concrete structure it would not only be necessary to describe the size and length of a member but also the internal reinforcing, concrete members when described in drawings are usually described by both external dimension and reinforcing drawings. For the purpose of demonstration the simpler description of the steel structure is preferable.

To simplify the design programs it was desirable to have a determinate structure that could be designed one member at a time with simple computations. Concrete structures are by their nature continuous and are indeterminate. Steel structures may be either determinate or indeterminate, resisting horizontal loads with either rigid or braced frames. The simplest form of bracing, diagonal or X-bracing, is a common feature of the steel industrial structure. These structures have little requirement for clear space in wall bays, being fully enclosed and window-less, so bracing can extend fully from floor level to floor level between neighbouring columns. The analysis of a braced structure for horizontal loads is easily accomplished with simple calculations.

The industrial structure is interesting to design both for the variety and magnitude of the loads it must support. The structural framing is often complicated, being determined by the locations of industrial equipment. In designing the industrial structure the design of the structure itself progresses in step with the design of the industrial equipment and structural design must often be performed with estimated loading. This leads to the making of many revisions to the structural design. A design system such as is here proposed, although limited in scope would prove a useful design tool. The repercussions of changes to the loading or description data could be evaluated quickly by the computer with greater thoroughness and accuracy than could be done by hand calculation.

A logical programming scheme is to follow the methods used in hand calculation. It is normal practice when analysing a structure to consider the member forces resulting from vertical and horizontal loads separately, later combining them for member design. It is convenient to follow this practice in the programming procedures. Evaluation of vertical forces alone is needed for the design of interior columns and beams that are not part of a braced frame. Horizontal forces alone are needed for the design of the diagonal bracing.

It is also beneficial to break up the programming into manageable single function programs. Each program can be responsible for a specific task, and these programs can act as subroutines to more comprehensive programs. The maintenance of the design system programs are much simplified, if this method is followed.

6.1 Internal and External Programs

There are two ways of programming in conjunction with a database management system (DBMS) [9], these are

- 1) program within the DBMS in the Data Manipulation Language,
- 2) program outside the DBMS in a standard programming language.

Of these two options the second is generally to be preferred as it allows the programs to be independent of the organization of data in the database and to be portable to other database systems. There are reasons, however, for programming within the DBMS. Essentially, when a program is mostly manipulation of data it is more efficient to operate in the DBMS. External programs rely on small DBMS programs to create input files and interpret output files. As the amount of manipulation involved increases, these intermediary programs become more complicated. When the maintenance of these programs becomes as much trouble as the maintenance of the actual applications program any benefit of computing outside the DBMS disappears.

A general rule then is to use an external program when the application involves mostly computing and an internal program when the application involves mostly data manipulation; all things being equal, the preference should be given to the external program. As the main reason for having external programs is ease of maintenance and portability, the size of the applications program relative to the intermediary DBMS programs is a useful measure of the benefit gained.

6.2 Basic Data Entry

The first contact the user will have with the design system is in entering basic data, building configuration and loading data. The user will more than likely be transferring information to the system from a physical sketch. People interpret information presented in graphical form more easily than either numerical data or text. In order to ensure that the correct data is entered into the design system, it would be a good idea to provide interactive graphics. The graphics provide instant verification of data and the user can immediately check for possible errors.

The database itself is only an information storage device and does not dictate the form of user interaction with the system. The input format can be decided solely in consideration of convenience to the user. The user does not have to enter data directly into the data files. Any suitable input format can be translated to the database by special programs. It would be quite possible to design the system interaction in such a way that the user might never deal with the actual database. The user could be one level removed from even the relational data model.

For reasons of data security and integrity it might not be desirable to permit users direct access to the database without some kind of access program, and were the system to be used in an actual design case, such a program would have to be included in the system.

Another option in data entry is automatic member location. The user would give basic building shape, such as bay spacings and floor level elevations and the system would locate the individual structural members. The great variability in loading and special requirements for open floor areas does not permit such a generalized approach to the description of an industrial structure. Such an automatic procedure would only work if the user could provide floor areas, the open space constraints and the expected loading and have the design system arrange the structural elements in relation to the specific requirements of the individual structure. It would be possible to program this and in so doing to optimize the arrangement of structural elements.

6.3 Geometry from Topology

The absolute locations of structure members in relation to a reference point does not necessarily play a part in the analysis of the structure, but may be needed for other applications. For example, if the analysis were to be done by a matrix analysis program these coordinates would be needed.

The calculation of geometry from topology, as the database stores member locations, is easily accomplished. The only structural "element" that is located in relation to a fixed reference point is the column line, all other locations are given relative to their supporting members. Since a column lies at the intersection of column lines, column location coordinates are easily calculated. A beam is located by its position along a supporting beam or column. If the coordinates of the support are known the coordinates of the supported beam can be calculated. As all beam systems are eventually supported by the columns, beam coordinates can be calculated by a simple procedure of regression.

Member coordinates should not be permanently stored but should be calculated as they are needed. This prevents the possibility of inconsistency that may result from storing redundant data. The related problem of data maintenance is discussed in Chapter 9.

6.4 Analysis

The role of analysis in structural design is to determine the behaviour of the structure under applied load. Once the basic data of building description and loading are in the database analysis can proceed. A structure subjected to a certain loading can behave in only one way and that behaviour can be determined.

The most important information provided by analysis are the locations and magnitudes of the forces that act on individual members; from these, internal forces within the members are calculated. Internal member forces include axial force, shear force and bending moment. Member deflections are also important member behaviour.

The modelling of the structure for analysis is a duty of design, which is also responsible for the selection of load combinations and application of load factors. In the present programming procedures, the model of the structure can be taken direct from the database and

load factors applied within the analysis programs. The difficulties of modelling has been for the most part avoided by taking liberties with the description of floor slabs in the database.

The first step in analysis is to apply the loading that is already stored in the database loading files to the supporting elements. The reactions of these elements are in turn applied to their supporting elements, and so reactions are propagated and the load paths are traced to the base of the columns. Since the structure that is being analysed is determinate, these forces do not depend on the section properties of the members, but only on their location. The individual member loadings are calculated for each loadcase and stored.

The next step is to factor the shear force and bending moment diagrams for various load combinations. Member deflection might also be calculated. From all possible loading combinations the shear force and bending moment envelopes can be determined for use in the selection of member sections. At this point analysis is essentially complete.

Analysis deals with a model of the structure. The creation of this model involves assumption, judgement and decision. Two options that present themselves at this juncture are to either fully automate the process of modelling or allow some measure of human interaction and control of the process. Probably the best system would provide as much automation as possible but leave some provision for the designer to instruct the system if required and assume some overall supervisory control of the process. The design system, after all, does not replace the designer; it is a tool that the designer uses.

6.5 Member Selection

In the structural design system the selection process evaluates criteria. This selection process should be kept separate from the analysis side of the design process. In the selection process the set of possible member sections are checked against the design criteria, and the most suitable section is selected. The criteria that is of greatest importance for the design of beams is the factored moment resistance. Once a section is chosen that satisfies this criteria it is further checked against the other criteria including factored shear resistance and allowable deflection and various geometry constraints.

Design criteria present special problems to the structural design programmer. There is often no clear separation of design and analysis in most structural design programs. In simple cases it is efficient for analysis and design to meet half-way although this is by no means true for the general case.

There should be a clear separation between analysis and design. Analysis deals solely with the calculation of structure and member behaviour. Design has a wider scope of evaluating the suitability of member locations, properties, etc. Much design has nothing to do with analysis. For example, if maintenance of headroom clearances demands a maximum beam depth no analysis is needed to determine if a certain beam section will qualify. Or, if an opening is required in a floor or wall it is design that ensures that no beam is accepted that intrudes on that area. *Design selects member sections, evaluates them against design criteria and then chooses the section that best satisfies the criteria.*

In the design of a simple beam a section is selected and analysed, then evaluated. The design of even a simple beam involves many trials, and analysis must be performed each time. But not all behaviour requires full knowledge of member properties. Shear force and bending moment are calculated from statics alone. If moment and shear resistance are considered member properties it appears that the trial process can be eliminated. This, however, is only an illusion. The process has been simplified by the pre-calculation of some data, but running down the table of moment resistances is itself a process of trial and evaluation, each value must be checked against required value and either accepted or rejected. The basic form remains.

The programming procedures for design should therefore be created to handle the evaluation of known data against known criteria. It should handle any type of evaluation, dealing with structure behaviour or description, and any type of constraint. The criteria should be separate from the applications programs so that criteria can be changed in the database without altering the performance of the design programs.

Chapter 7

The Design System

In order to demonstrate the concept of the building structure database, a body of programs with limited objectives has been created. The goal is to create a system that will analyse a multi-storey diagonally braced steel industrial structure and select appropriate member sections

The resources that I have made use of to achieve this goal were very limited. They were

- 1) Hardware: dy-4 Systems Inc. "ORION V", Z80A based micro-computer system featuring 64K RAM, and two 8 inch floppy drives (single-sided, double density) with 600K bytes per disk (1.2 Megabytes total) and CP/M 2.2 disk operating system
- 2) Software: Ashton-Tate "dBASE II" assembly-language relational database management system

The dBASE II relational database management system provides a true relational model, although the query language is unsophisticated (combined relational operations, for example, must be performed step by step.) The datafiles are implemented as described in Chapter 5, with the addition only of some attributes representing data that would not normally be stored permanently. This temporary data is retained either for purposes of demonstration or because re-calculation on the micro-computer is considered too time consuming.

The system in its present form calculates the external and internal member forces due to vertical loading on floor levels and roofs, and selects beam sections on the basis of factored moment resistance. The system might be easily expanded to select sections for other members, and the database would, in its present state, serve many uses outside of structural design.

In its present context, the database organization puts no limits on the size of the structure or the number of members or connections. Any limits exist only in the data storage

capacity of the commercial DBMS and the micro-computer. The programs also set no intrinsic limits. An overview of the data files, applications programs and the flow of data is presented in Figure 7-1.

7.1 Resource Limitations

Considerable limitations on the development of the building structure database and the design system were imposed by the limited capabilities of the resources used. By narrowing the scope of the description to a relatively simple class of structure, the applications program computations were kept direct and uncomplicated. As computation played a much smaller role in the applications programs than data manipulation, the programs were written in the dBASE II data manipulation language.

dBASE II provides only the four basic mathematical operations (addition, subtraction, multiplication and division) for computing. Functions such as square root and the trigonometric functions are not supplied. Although it is possible to create programs to calculate the values of these functions using the basic four operations, as I have done only with square root, the resulting programs tend to be cumbersome in use. The dBASE II data manipulation language is not suitable for involved calculations.

The major effect of this limitation is in regard to the description of floor slabs. While the obvious manner of defining a continuous floor slab would be to describe the location of the edges of the floor area as a whole, this approach would require an applications program to determine the resulting loading on the supporting beams; this program would have to be external to the DBMS in a standard programming language. It was a matter of practicality to settle for a less than satisfactory floor slab description in order to simplify the beam load calculations.

Another consideration with dBase II is that it allows two data files to be open at the same time. Data manipulation in these two files proceeds at a reasonable pace, but the closing of files and the opening of new files is slow. The more an applications program requires files to be closed and opened, the longer the execution time. It is on this account that in the building structure data files, calculated data is in some files stored along side user supplied

Figure 7-1: Files, programs and data flow.

data rather than in a separate file. In a more powerful DBMS this would be neither necessary nor desirable.

The resources used offer only the most basic and indispensable elements necessary for a design system, dBASE II is a general purpose relational DBMS, not specifically suited to engineering applications

7.2 Data Files

Data files can be separated into the categories of structure description, loading data, structure behaviour and design criteria.¹ Reference databases can be related to these categories. They can also be grouped as:

- 1) Basic Data Files, containing data that must be supplied to the system by the user, such as basic building description and loading;
- 2) Calculated Data Files, containing data that is derived from the basic data;
- 3) Reference Data Files, containing basic data information that is not specific to the one structure and is used selectively.

7.2.1 Basic Data Files

The basic data that must be supplied to the system are the basic building description, the loading data and the design criteria. The basic data files that have been implemented are:

• AXES

This file has been implemented as described in Chapter 5 (cf. Fig. 8-4.) This is a list of directions relative to a reference direction, measured by a clockwise rotation in degrees. Every orientation has a partner orthogonal to itself, that is, lying ninety degrees clockwise. Any number of directions can be listed, each having a two character name. The orientations apply to the directions of column lines and the orientation of the major axes of columns

¹ It may be helpful in the following descriptions to refer to the actual data files relating to the sample design of Chapter 8, Figures 8-4 thru 8-27.

The reference direction would probably, but not necessarily, be true north, relative to which the structure "north" would be defined. Several sets of orthogonal column lines can be defined for non-rectangular structures.

• COLINES

This file contains the list of column lines (cf. Fig. 8-5.) Column lines are actually vertical planes. Each column line has a four character name. Its location is given relative to a fixed point by the right angle distance of the line to the point. As a direction and a distance can define two planes, positive and negative distance values are required. These are determined using the reference direction; if the closest point on the line to the reference point lies to the left (west) of a line in the reference direction running through the reference point, the distance value is negative; if it is to the right (east) then it is positive.

While this is the manner of data storage, it is assumed that in an improved system input of column line locations could be made in terms of a local reference point and direction; the local values then being translated to the reference values for permanent storage.

• COLUMNS

This file contains locations of columns (cf. Fig. 8-7.) A column must lie on the intersection of column lines. Columns are assumed to be vertical members. There is no provision for slanted columns; in most cases this is not a serious restriction. There is also no provision for column offsets. Column major axis orientation is given as pre-defined in the file AXES.

There is as yet no provision for column splices or column sections. At the present level of description the column is assumed to consist of a single member. A more complete description would require some additional relations that would define splice locations and member properties between splices.

• LEVELS

This file is the list of floor level elevations (cf. Fig. 8-6.) By defining floor levels it is not necessary to give a vertical coordinate for the location of beams or floor slab. By giving the horizontal location of these elements within a floor level, their elevation can be calculated. At present, floor levels are described as horizontal planes. However, it is a small step to a description of sloping floors, such as are required at roof level.

• BEAMS&L

This file stores beam data (cf. Fig. 8-26.) The "&L" part of the file name is a two character level name. For example, the file BEAMS01 is the file for beams lying in Level 01. As implemented this file has some additional attributes whose values are supplied by the applications programs. They are included in the BEAMS file to allow more efficient programming in the dBASE II language. The additional attributes are: X1, Y1, X2 and Y2 (beam end coordinates); LENGTH (beam length); and RLENGTH (unrestrained length of the compression flange.) Only simply supported beams are described, and all beams must conform to a support hierarchy; the applications programs cannot handle the case of a beam that indirectly supports itself. Also, no indeterminacy is allowed. The data file is not equipped to describe beams continuous over supports, cantilever beams or moment connections, although all these are common in industrial structures. The datafile can describe only simply supported beam systems.

• PANELS&L

This file describes floor slab locations (cf. Fig. 8-10.) As with the beam files, the panel files are separated into floor levels. This file does not give a true picture of the floor slab. In order to provide an easy calculation of tributary loading areas the floor slabs are described as rectangular panel sections simply supported on a pair of parallel beams. While beams can lie in any direction, the restriction of rectangular floor panels requires that beams run parallel or at right angles (only, of course, if they support floor panels). Dealing with continuous floor slab presents some difficulties in load distribution to the supporting beams that are bypassed by assuming simply supported panels. While it is possible to assume that adjoining slabs are continuous for purposes of slab design and drafting, the description, on the whole is unsatisfactory. The obvious way to handle floor slab description would be to simply define the floor slab area, making use of an applications program to work out the resulting slab forces and beam loading. Unfortunately the computer resources were not suitable for this task and a much less satisfactory but expedient description was used.

• CONCL&L

This file describes the locations and magnitudes of concentrated loads (cf. Fig. 8-14.) Concentrated loads are placed only on beams. Each concentrated load has a named source,

for example, the name of a piece of equipment. This information was found to be useful to the user, although it is not presently used in the programs.

- PLX&L

This file describes the locations and gives magnitudes of uniformly distributed loads (cf. Fig. 8-12.) These loads are placed only on floor panels. The load is uniform over the width of the panel but may be located on any portion of the length of the panel. There is no provision for distributed loading other than uniform. Load sources are not currently named

- CASES

This is the only design criteria file—it gives a list of loading combinations, load cases and load factors (cf. Fig. 8-15.) Any number of load cases and combinations is allowed. Load factors, in the present description, also include the combination factor. For each load combination there is a list of load cases and their factors. Load case is loosely defined, any load or set of loads that the user wishes to keep separate from the other loading can be given a separate load case. There is no limit to the number of load cases allowed.

7.2.2 Calculated Data Files

Calculated files contain data that is derived from the basic data. These files include the member external and internal forces, information necessary for member selection. The files that contain calculated data are:

- BLX&L

This file records beam external forces; forces that come directly from the loading files and also from supported beam reactions (cf. Fig. 8-18.) Each external force has a named source. This can be an applied load or a reaction from a floor panel or beam. Without a named source duplicate record instances could easily occur. For example, two beams connecting to opposite sides of a supporting beam might have the same reaction for the same loadcase, and therefore the external load file would have a duplicate row, which would violate a basic principle of relational databases. The named source prevents this from occurring

- CLX

This file records column external forces (cf Fig. 8-20) The columns do not support applied loads. All loading comes from reactions. As in the beam forces file sources are named

- BMDIAG&L

This file stores factored beam bending moments and shears for a given floor level (cf Fig. 8-21). For any load combination there are recorded the values of beam bending moment at the critical points of the bending moment diagram and shears to the left and right of that point. The attributes of this relation are NAME, LOADCOMB, POSITN, SHEAR1 (to the left), SHEAR2 (to the right) and MOMENT1.

- MOMENT&L

This file stores maximum factored bending moments and shears taken from the relation BMDIAG&L for each beam on a given level for all load cases (cf Fig. 8-23). The attributes are: NAME, LOADCOMB, SHEAR, MOMENTP (maximum positive moment) and MOMENTN (maximum negative moment). The maximum shear and bending moment is needed for beam design. This file contains information already found in the file BMDIAG&L and is really only a temporary file.

7.2.3 Reference Data Files

The only reference datafile that is implemented is an abbreviated beam selection table

- BMSELECT

This file is a beam selection table (cf Fig. 8-25). The attributes are SECTION, LU, VR, L0000, L0015, L0020, L0100 and LOVER. The attribute LU is the maximum unsupported length of the compression flange for which there is no reduction in the factored moment resistance, VR is the factored shear resistance, L0000 to L0100 are the factored moment resistances for unrestrained lengths of zero to ten meters. LOVER is the moment resistance for an unrestrained length of over ten meters and is assumed zero for all sections, thus placing a practical limit on the allowable unrestrained length of beams.

It would be much more satisfactory to have the member sections chosen directly from a

steel sections properties table, but the limited computational capabilities of dBASE II made this inconvenient

7.3 Programs and Transient Data Files

The programming procedures for structural design of the determinate structure can be separated into several discrete functions. The program duties are

- 1) basic data entry,
- 2) calculation of geometry from topology,
- 3) calculation of external member forces,
- 4) calculation of internal member forces,
- 5) selection of member sections

In the present system all programming is done in dBASE II. This is done for a number of reasons. Because the structure to be designed is determinate and relatively simple, the greater part of programming involves data manipulation. As discussed previously the benefits of programming outside the database system diminish as more data manipulation is needed. The available resources also limited the practicability of creating external programs. One drawback is that there is no capability of having the external program run automatically either from a command from the database system or another programmable source. Each external program would have to be run manually. This is clearly not a satisfactory situation. It was preferable to have all programs in the database where they could be run individually from a keyboard command or as a sub-program in a larger routine.

As noted above, the programming does not set any limits on the size or complexity of the structure, given that it can be described in the limited vocabulary of the database. The limits exist in the DBMS and the computer, and are set by the maximum dBASE II data file size and the disk storage capacity.

7.3.1 Basic Data Entry

The preferred manner of data entry for building description and loading would use interactive graphics. Unfortunately, the available resources did not provide the facility for graphics. The dBASE II system does however provide a good system of direct data entry to the data files with a full screen editing feature. No real improvement could be made on the present input format without the inclusion of graphics. No data entry programs are included in the design system, therefore great care must be taken to ensure that the data is correctly transferred to the database, and that the data entered is consistent.

7.3.2 Computing Geometry from Topology

Geometry is calculated from topology by the program XANDY. This program calculates the x and y coordinates of beam ends for a single floor level. The program calls on several sub-programs to perform some repetitive functions.

The only structural member that is fixed in relation to a global reference point is the column line. As columns lie on column line intersections their coordinates can be easily calculated, this is done by the sub-program COLCOORD. The sub-program COLCOORD is not able, at the present time, to handle column line directions other than "NS" and "EW" due to the difficulty with which dBASE II can be made to handle even simple trigonometry.

A beam has two ends, each of which may connect to a column or to another beam. If a beam end connects to a column, the $x - y$ coordinates of the column are the same as those for the beam end. If a beam end connects to another beam (its location along that beam is already stored in the database), and if the end coordinates of the supporting beam are known, the coordinates of any point along that beam can be calculated, this is done by the sub-program BEAMEQ. Beam coordinates are stored along with other beam data in the file BEAMS&L. Beam coordinates are redundant data that would not be permanently stored.

The calculation of beam coordinates requires the previous knowledge of the coordinates of the supporting beam. The order in which beam coordinates are calculated is therefore important. The correct order for calculation is determined by the program BEAMLIST and

the order is stored, for immediate use only, in an ordered file also named BEAMLIST. This program also plays a role in calculation of member forces and is further explained below.

While beam coordinates do not have a direct part in the analysis of the structure they are used to calculate beam lengths which are needed in the analysis computations. The program that calculates beam lengths from beam coordinates is the program LENGTH and it is included in the program XANDY as a matter of convenience; it can also stand alone. As dBASE II does not offer a square root function, this function is performed by the program SQRTX.

The program XANDY goes down the file BEAMLIST and calculates end coordinates for each beam in order. The sub-programs PUTCORD1 and PUTCORD2 insert the coordinates in the file BEAMS&L.

7.3.3 Computing External Member Forces

External member forces are calculated by the program AUTOLOAD and stored in the files BLX&L and CLX. AUTOLOAD runs for a single loadcase for a single level. The file BLX&L contains the beam forces for a given floor level and the file CLX contains all the column forces.

AUTOLOAD is essentially a joining of three separate programs.

The program CONCLOAD takes the concentrated loads recorded in the file CONCL&L and transfers them to the beam loading file BLX&L.

The program PANEL takes panel distributed loads and calculates the resulting beam loadings. Panels are rectangular floor sections assumed simply supported for the purpose of calculating the resulting beam loadings. The beam loadings are directly stored in the file BLX&L.

The program BEAMEVAL does the most work in taking the existing loading already stored in BLX&L and propagating beam reactions. For each beam BEAMEVAL calculates reactions at the beam ends from the beam loading. If a beam end connects to a column the reaction is stored as a column load in the file CLX; if a beam end connects to another beam the reaction is stored as a beam load in the file BLX&L.

As with calculation of beam coordinates the order of calculating and propagating reactions is important. Before the reactions of any beam are calculated the reactions of the beams that it supports must first be calculated. The correct order is determined by the program BEAMLIST and stored in the file BEAMLIST.²

Sub-programs of the program BEAMEVAL are PUTLOADC and PUTLOAD, these programs perform the actual operation of inserting calculated external member loads into the files CLX and BLX&L respectively.

Factored loads are not stored in the database but factors are applied before analysis. There are several reasons for doing this. Firstly, factors vary with the construction material. Steel design use the factors 1.5 and 1.25 for live load and dead load, for concrete the factors are 1.7 and 1.4, soil bearing is still based on the system of allowable stress. The loading on a steel structure on concrete foundation therefore will have different factors depending on the part of the structure being designed. Other disciplines are also expected to make use of the structure database, and they will be interested not in factored loads but in the actual unfactored loads. As the database should avoid redundant data, either factored loads or unfactored loads should be stored, but not both. It is therefore far more practical to store unfactored loads.

7.3.4 Computing Internal Member Forces

Internal member forces are stored in terms of shear force and bending moment. Cal-

² The file BEAMLIST is initialized as the un-ordered list of all the names of beams on a given level, with a marker added to the bottom of the list. The applications program scans this list of beams and each beam that is known to support another beam in the list. This information is obtained from the file BEAMS&L. is moved below the marker. The list above the marker will now contain only the names of those beams that support no others, the first level of independence. The program now ignores the beams above the marker-- the first level of independence-- and operates on the bottom list. A new marker is put at the bottom and all beams that support another in the same list, i.e., the bottom list are moved below the marker. This creates a second level of independence. The program now ignores the second level and operates on the new bottom list. The process is repeated, creating new levels of independence, until no new levels can be created, at some point the bottom list will contain no beam that supports another in the same list. The result of this program is a list of all the beam names separated by markers into levels of independence (cf. Fig. 8-16). For propagation of beam reactions calculation proceeds from the top of the list (with the independent or supported beams), for beam end coordinates calculation proceeds from the bottom of the list (with the dependant or supporting beams). The markers can be ignored.

ulation of these is performed by the program MOMENTS. This program runs for a single floor level and load combination.

The first step in calculation of internal forces is application of load factors for a certain Loading Combination (design is in accordance with Limits States Design). The load factors from the file CASES are joined to the beam loading file to create a temporary factored loading file called BMLOAD. From the factored loading the shear force and bending moments at critical points along the length of the beam are calculated by the program SHANDBM. This program works on the principle of superposition. The shear force and bending moment for each force are calculated separately, then combined. Because of this, the program will handle any number of concentrated or distributed loads. It could also be easily extended to handle any other type of load, such as an applied moment, for example.

The positions along the beam at which shear force and bending moment are calculated are determined by the sub-program POSITNS. These positions are the beam ends, wherever concentrated loads occur and wherever distributed loads start or end. Another important point occurs where the shear force diagram crosses the zero axis. This point is located by the sub-program ZCROSS in the program SHANDBM.

The program MANDMS takes the values of shear force and bending moment in the file BMDIAG&L for all load combinations and extracts the maximum shear and the maximum positive and negative moments. These are stored in the file MOMENT&L for each level.

7.3.5 Member Selection

The actual selection of beam sections is performed by the program BMSELECT. This program runs for a single level at a time. It selects beam sections from an abbreviated beam selection table on the basis of factored moment resistance.

The first thing this program does is calculate the unrestrained length of the compression flange for each beam. This is accomplished by the sub-program URESTRL. Unrestrained length is stored in the file BEAMS&L under the attribute RLENGTH. It really should not be permanently stored as it is calculated data, but its inclusion in the beams file speeds the applications program execution.

Having the factored moment and the unrestrained length, the program BMSELECT runs through the beam selection table and chooses the lightest member that has an adequate factored moment resistance. The selected beam section is permanently stored in the file BEAMS&L.

7.4 Limits to Usefulness

Within the strict limitations imposed on building configuration, the present design system provides an accurate analysis of the external forces acting on beams and columns. These are not complete as the effects of horizontal loading are not considered. Still, beams and columns that are not part of a braced frame are not affected by horizontal loading and these can be designed.

The weakest part of the design system, as should be expected, is the evaluation of design criteria. In the case of the present system, only two criteria were evaluated. Member selection was based on the criteria that factored moment resistance must exceed the factored moment and that the beam section should be the lightest section possible. The evaluation of these criteria is embedded in the applications program. In a working system this kind of programmed evaluation should not be an acceptable practice, criteria should be separate from the programming.

The present system is a useful tool. It provides an acceptable preliminary design of floor systems in a simple steel industrial structure. It could be expanded to include the design of interior columns with little trouble; the data needed for this is already contained in the database. If greater effort were to be expended on the system, it could include the analysis of horizontal effects and the design of all basic framing elements of the simple structure.

Because the whole design process is programmed, changes to the basic data require little manual work. Even though the present system offers no special updating procedures, re-analysis makes little demand on the users time. For example, major changes in loading data would require that the design be totally re-calculated whether done by computer or by hand, with the computer one needs only input the changes and the computer does the re-calculation. Also, the computer can store more information more accessibly than is usually done manually.

Of course, the present system was created only to demonstrate the use of the building structure database. The database itself has many other uses outside of structural design. The database could be used as the basis of a drafting system. With the database, material-take-off for estimating is a simple matter. The database also serves as a reference for the designer. Should the designer require any building structure information, it can be quickly accessed through the DBMS. Everything that needs to be known about the structure is, or should be, recorded in the database. There are no limits to what can be done with this information.

Chapter 8

Sample Design

The sample design is a two storey industrial building as shown in Figures 8-1 thru 8-3. The length of the building consists of two bays of seven and eight meters, and the width consists of a single bay of seven meters.

The first level supports a distributed dead and live load, and also some concentrated loads imposed by some large fixed mechanical equipment. The arrangement of floor beams is somewhat more complicated than should be necessary, this is to demonstrate the abilities of the system to handle complicated beam arrangements. The floor slab is assumed to consist of independant floor panels spanning between floor beams. Where no floor panels are designated an open floor area is assumed.

The second level is the roof and it supports a distributed dead load and snow load. The arrangement of beams is uncomplicated.

The file listings that follow, Figures 8-4 thru 8-27, have been reproduced from the user console as presented by dBASE II. The command "USE filename" opens a file. The command "LIST STRUCTURE" gets the column positions and descriptions of the relation attributes, including name, type and number of characters. The command "LIST" gets a listing of the file contents without column headings, one must refer to the output of the LIST STRUCTURE command for the heading for a particular column. The first column in each listing is the record number, this is not part of the relation.

8.1 Entry of Basic Data

The first step in using the design system is the entry of basic data into the database. Initially, the datafiles (relations) in the database are defined but empty. The basic description of the building structure and the loading must be entered. The dBASE II DBMS provides a

good full screen editing feature for putting data into the datafiles. No additional data entry programming was created, except the optional program LEVELS; this program automatically initializes the level dependant secondary data files as floor level elevations are input.

All data is in metric units. The basic units are meters (m) and kilo-newtons (kN).

8.1.1 Building Description

The first information that should be given to the system are the possible line directions and coordinate axes; these are entered into the file AXES (Fig. 8-4.) A column line may be oriented either "NS" for north-south or "EW" for east-west. This file consists of the single record NAME="NS", ORTHO="EW"; ROTATION="0.00".

The next required information is the names and locations of the column lines. These are recorded in the file COLINES (Fig. 8-5.) A column line name can be any string of four characters. To avoid the use of blanks, confusing because they are invisible, leading zeroes are used to fill the character places; for example, column line "B" is entered as "000B". The location is the right angle distance from the column line to a fixed reference point in meters. In this sample design, the reference point is chosen arbitrarily as the south-west corner of the building, at the intersection of column lines "000B" and "0001". Each column line also has a direction, either "NS" for north-south or "EW" for east-west; for the time being no other directions are recognized. These directions have already been defined in the file AXES.

The elevations of the floor levels is entered in the file LEVELS (Fig. 8-6.) This file consists of the name of the level (a two character string), and a top of steel elevation.

Column names and locations are stored in the file COLUMNS (Fig. 8-7.) Each column has a unique four character name and a location specified by the names of the two column lines on whose intersection the column lies. The orientation of a column's principle axis may be either "NS" or "EW", as defined for the column line directions. Columns are assumed to be single continuous members.

The beams locations on both levels can now be entered in separate files. These files are BEAMS01 and BEAMS02 (Figs. 8-8 & 8-9.) The data consists of the beam name, the name of the members to which it connects and the positions of the connections along those

members. There are some other attributes in this relation whose values are later supplied by the applications programs (beam end coordinates X1, Y1, X2, Y2, LENGTH, unrestrained length RLENGTH, SECTION), at the time of data entry these are left empty.

The location of floor panels is entered into the data files PANELS01 and PANELS02 (Figs. 8-10 & 8-11). The data consists of the names of the supporting beams (joists) and the location of the corners of the panel on those beams. The width of the panel, that is, the distance separating the supporting joists, is supplied by the user.

At this point the structure is fully described to the present limits of the database. This description consists of the basic framework of columns, beams and floor slab.

The next step is to enter the loading.

8.1.2 Loading Data

Distributed loads are entered in the file PLX01 and PLX02 (Figs. 8-12 & 8-13). Distributed loads are applied across the full width of floor panels. The location of a distributed load is given by the name of the supporting floor panel (NAME), the location of the load on that panel, the magnitude of the load in kN/m^2 (LOAD) and the loadcase. The location of the load is given by two position attributes (POSITN1 and POSITN2). These give the distance of the edges of the loaded zone relative to the edge of the panel.

Concentrated loads are entered in the files CONCL01 (Fig. 8-14) and CONCL02. In the present example there are no concentrated loads on Level 02 so file CONCL02 is left empty. Concentrated loads are applied only on beams and are located by their position along the supporting beam (POSITN1). Each load is given a magnitude in kN (LOAD) and loadcase, but in addition each load is also given two attributes, SOURCET and SOURCEN, representing the type and name of the source of the load. The sources of concentrated loads—the mechanical equipment—in this sample design are given the names "Z0001" and "Z0002". Naming the sources makes it possible to manipulate these loads as groups.

8.1.3 Design Criteria

The only design criteria recorded in the database are the load combinations and load factors. These are recorded in the file CASES (Fig. 8-15.) For every load combination that is to be analysed there is a set of individual load cases and corresponding load factors. In the present sample design, since there is no wind load, there is only one load combination, dead load plus live load. The applications programs can however handle any number of load combinations, load cases and load factors.

All basic data has now been recorded and the design system can be run.

8.2 Running the System

The design system consists of four programs. These programs must be run in order, as the data resulting from one program is needed for the next. The programs are run for one floor level at a time. As the structure is determinate, the design of one floor level may be completed without regard for the progress of design on other floor levels. The programs and their proper order are:

- 1) XANDY,
- 2) AUTOLOAD,
- 3) MOMENTS,
- 4) BMSELECT

The commands required to run these programs are very simple. The user simply types in a DO command with the name of the applications program; for example, "DO AUTOLOAD". If the application program requires user supplied information such as floor level name or load case, the user is prompted on the console and the information is input interactively. To view any data file the user types in "USE filename" to open the file and then "LIST" to view the contents.

8.2.1 Geometry from Topology

To get the coordinates of the beam ends and also the beam lengths that are necessary for analysis, the program XANDY is run. The command for running the program is "DO XANDY". The program will request the name of the floor level. It will also ask "BEAMLIST required? (Y/N)" to which the answer must be "Y". A sub-program BEAMLIST creates an ordered listing—a temporary file also called BEAMLIST—of beam names in order of independence. For demonstration, the contents of this file for Levels 01 and 02 have been copied to the files BMLIST01 and BMLIST02 (Figs 8-16 & 8-17.) The results of the program XANDY, the beam end coordinates and length, are recorded in the files BEAMS01 or BEAMS02 (cf Figs 8-26 & 8-27.)

8.2.2 Analysis

The first part of analysis is performed by the program AUTOLOAD which takes the applied loading, factors it and calculates external member forces. This program records external beam forces in the files BLX01 and BLX02 (Figs 8-18 & 8-19.) Column external forces are recorded in the file CLX (Fig 8-20.) The location, magnitude and source of each external force is given in these files.

Further analysis is performed by the program MOMENTS. Using the beam external forces, beam internal forces, shear forces and moments at key locations, are recorded in the files BMGIAG01 and BMDIAG02 (Figs 8-21 & 8-22), the maximum shear and moment values are stored in the files MOMENT01 and MOMENT02 (Figs 8-23 & 8-24.)

8.2.3 Member Selection

Member selection is the last part of the process. This is performed by the program BMSELECT. In this program the maximum unrestrained length of beams are calculated and stored in the beam file. Beam sections are chosen from an abbreviated beam selection table based on factored moment resistance (Fig 8-25.)¹ The final selection is stored in the beam

¹ The reference data file BMSELECT is an abbreviated beam selection table. Because this is an exceptionally wide table, only the first nine columns—representing moment capacity up to an unrestrained length of 4 meters—are presented. As no beam in this present example has such a long

files, BEAMS01 and BEAMS02 (Figs 8-26 & 8-27) It is important to realize that this selection of beam sections is incomplete, it does not consider all necessary design criteria Other criteria besides factored moment resistance currently would have to be checked manually

unrestrained length the portion listed is sufficient for this illustration The partial table has been copied to the file BMSEL, this table is listed

Figure 8-1: Plan at Level 01.

Figure 8-2: Plan at Level 02.

Figure 8-3: Elevation on column line "B".

```

. USE AXES
.
. LIST STRUCTURE
STRUCTURE FOR FILE: AXES.DBF
NUMBER OF RECORDS: 00001
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      002
002      ORTHO     C      002
003      ROTATION  N      010    005
** TOTAL **                00015
.
. LIST
00001  NS EW      0.00000

```

Figure 8-4: AXES—reference axes systems, column line directions.

```

.
.
. USE COLINES
.
. LIST STRUCTURE
STRUCTURE FOR FILE: COLINES.DBF
NUMBER OF RECORDS: 00005
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      004
002      DIST      N      010    005
003      ORIEN     C      002
** TOTAL **                00017
.
. LIST
00001  0001      0.00000 NS
00002  0002      7.00000 NS
00003  0003     15.00000 NS
00004  000A      7.00000 EW
00005  000B      0.00000 EW

```

Figure 8-5: COLINES—column line locations.

```

. USE LEVELS
.
. LIST STRUCTURE
STRUCTURE FOR FILE: LEVELS.DBF
NUMBER OF RECORDS: 00002
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      002
002      TOS:EL     N      010    003
** TOTAL **                00013
.
. LIST
00001  01      10.000
00002  02      15.000

```

Figure 8-6: LEVELS—floor level elevations.

```

. USE COLUMNS
.
. LIST STRUCTURE
STRUCTURE FOR FILE: COLUMNS.DBF
NUMBER OF RECORDS: 00006
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      004
002      LINE1      C      004
003      LINE2      C      004
004      ORIEN      C      002
** TOTAL **                00015
.
. LIST
00001  0001 0001 000A EW
00002  0002 0002 000A NS
00003  0003 0003 000A EW
00004  0004 0001 000B EW
00005  0005 0002 000B NS
00006  0006 0003 000B EW

```

Figure 8-7: COLUMNS—column locations.

USE BEAMS01

LIST STRUCTURE

STRUCTURE FOR FILE: BEAMS01.DBF

NUMBER OF RECORDS: 00023

DATE OF LAST UPDATE: 00/00/00.

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	004	
002	CONTYPE1	C	001	
003	CONNAME1	C	004	
004	POSITN1	N	010	005
005	X1	N	010	005
006	Y1	N	010	005
007	CONTYPE2	C	001	
008	CONNAME2	C	004	
009	POSITN2	N	010	005
010	X2	N	010	005
011	Y2	N	010	005
012	LENGTH	N	010	005
013	LENGTH	N	010	005
014	SECTION	C	009	
** TOTAL **				00104

LIST

|--|

Figure 8-8: BEAMS01—beams on Level 01, Initial condition.

```

. USE BEAMS02
. LIST STRUCTURE
STRUCTURE FOR FILE: BEAMS02.DRF
NUMBER OF RECORDS: 00013
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD. NAME TYPE WIDTH DEC
001 NAME C 004
002 CONTYPE1 C 001
003 CONNAME1 C 004
004 POSITN1 N 010 005
005 X1 N 010 005
006 Y1 N 010 005
007 CONTYPE2 C 001
008 CONNAME2 C 004
009 POSITN2 N 010 005
010 X2 N 010 005
011 Y2 N 010 005
012 LENGTH N 010 005
013 RLENGTH N 010 005
014 SECTION C 009
** TOTAL ** 00104

. LIST
00001 0001 C 0004 0.00000 0.00000 0.00000 0.00000 0.00000
00002 0002 C 0001 0.00000 0.00000 0.00000 0.00000 0.00000
00003 0003 C 0004 0.00000 0.00000 0.00000 0.00000 0.00000
00004 0004 C 0005 0.00000 0.00000 0.00000 0.00000 0.00000
00005 0005 B 0001 1.75000 0.00000 0.00000 0.00000 0.00000
00006 0006 B 0001 3.50000 0.00000 0.00000 0.00000 0.00000
00007 0007 B 0001 5.25000 0.00000 0.00000 0.00000 0.00000
00008 0008 C 0005 0.00000 0.00000 0.00000 0.00000 0.00000
00009 0009 C 0002 0.00000 0.00000 0.00000 0.00000 0.00000
00010 0010 C 0006 0.00000 0.00000 0.00000 0.00000 0.00000
00011 0011 B 0008 2.00000 0.00000 0.00000 0.00000 0.00000
00012 0012 B 0008 4.00000 0.00000 0.00000 0.00000 0.00000
00013 0013 B 0008 6.00000 0.00000 0.00000 0.00000 0.00000

```

SECTION
COLUMN
BLANK

Figure 8-9: BEAMS02—beams on Level 02, Initial condition.

```

USE PANELS01

. LIST STRUCTURE
STRUCTURE FOR FILE: PANELS01.DBF
NUMBER OF RECORDS: 00016
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD  NAME      TYPE  WIDTH  DEC
001  NAME      C      004
002  JOIST1     C      004
003  POSITN1A   N      010   005
004  POSITN2A   N      010   005
005  JOIST2     C      004
006  POSITN1B   N      010   005
007  POSITN2B   N      010   005
008  WIDTH      N      010   005
** TOTAL **      00063

. LIST
00001 0001 0001 0.00000 0.00000 4.00000 0007 0.00000 2.00000
00002 0002 0007 0.00000 0.00000 2.00000 0008 0.00000 1.66667
00003 0003 0008 0.00000 0.00000 2.00000 0009 2.00000 1.66667
00004 0004 0009 0.00000 0.00000 2.00000 0002 2.00000 1.66667
00005 0005 0001 4.00000 0.00000 7.00000 0010 3.00000 1.75000
00006 0006 0010 0.00000 0.00000 3.00000 0011 3.00000 1.75000
00007 0007 0011 0.00000 0.00000 3.00000 0012 3.00000 1.75000
00008 0008 0012 0.00000 0.00000 3.00000 0002 7.00000 1.75000
00009 0009 0013 0.00000 0.00000 8.00000 0016 8.00000 1.75000
00010 0010 0016 0.00000 0.00000 3.00000 0020 3.00000 1.75000
00011 0011 0016 3.00000 0.00000 5.00000 0022 2.00000 0.75000
00012 0012 0016 5.00000 0.00000 8.00000 0021 3.00000 1.75000
00013 0013 0020 0.00000 0.00000 3.00000 0017 3.00000 1.75000
00014 0014 0023 0.00000 0.00000 2.00000 0017 5.00000 0.75000
00015 0015 0021 0.00000 0.00000 3.00000 0017 8.00000 1.75000
00016 0016 0017 0.00000 0.00000 8.00000 0014 8.00000 1.75000

```

Figure 8-10: PANELS01—floor panels on Level 01.


```

. USE PANELS02
.
. LIST STRUCTURE
STRUCTURE FOR FILE: PANELS02.DBF
NUMBER OF RECORDS: 00008
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD  NAME      TYPE WIDTH DEC
001  NAME      C    004
002  JOIST1    C    004
003  POSITN1A  N    010  005
004  POSITN2A  N    010  005
005  JOIST2    C    004
006  POSITN1B  N    010  005
007  POSITN2B  N    010  005
008  WIDTH     N    010  005
** TOTAL **      00063

. LIST
00001 0001 0003 0.00000 7.00000 0005 0.00000 7.00000 1.75000
00002 0002 0005 0.00000 7.00000 0006 0.00000 7.00000 1.75000
00003 0003 0006 0.00000 7.00000 0007 0.00000 7.00000 1.75000
00004 0004 0007 0.00000 7.00000 0004 0.00000 7.00000 1.75000
00005 0005 0004 0.00000 7.00000 0011 0.00000 7.00000 2.00000
00006 0006 0011 0.00000 7.00000 0012 0.00000 7.00000 2.00000
00007 0007 0012 0.00000 7.00000 0013 0.00000 7.00000 2.00000
00008 0008 0013 0.00000 7.00000 0010 0.00000 7.00000 2.00000

```

Figure 8-11: PANELS02—floor panels on Level 02.

. USE PLX01

. LIST STRUCTURE

STRUCTURE FOR FILE: PLX01.DBF

NUMBER OF RECORDS: 00032

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	004	
002	LOADCASE	C	005	
003	LOAD	N	010	005
004	POSITN1	N	010	005
005	POSITN2	N	010	005
** TOTAL **			00040	

. LIST

00001	0001	DEAD1	5.00000	0.00000	4.00000
00002	0002	DEAD1	5.00000	0.00000	2.00000
00003	0003	DEAD1	5.00000	0.00000	2.00000
00004	0004	DEAD1	5.00000	0.00000	2.00000
00005	0005	DEAD1	5.00000	0.00000	3.00000
00006	0006	DEAD1	5.00000	0.00000	3.00000
00007	0007	DEAD1	5.00000	0.00000	3.00000
00008	0008	DEAD1	5.00000	0.00000	3.00000
00009	0009	DEAD1	5.00000	0.00000	8.00000
00010	0010	DEAD1	5.00000	0.00000	3.00000
00011	0011	DEAD1	5.00000	0.00000	2.00000
00012	0012	DEAD1	5.00000	0.00000	3.00000
00013	0013	DEAD1	5.00000	0.00000	3.00000
00014	0014	DEAD1	5.00000	0.00000	2.00000
00015	0015	DEAD1	5.00000	0.00000	3.00000
00016	0016	DEAD1	5.00000	0.00000	8.00000
00017	0001	LIVE1	20.00000	0.00000	4.00000
00018	0002	LIVE1	20.00000	0.00000	2.00000
00019	0003	LIVE1	20.00000	0.00000	2.00000
00020	0004	LIVE1	20.00000	0.00000	2.00000
00021	0005	LIVE1	20.00000	0.00000	3.00000
00022	0006	LIVE1	20.00000	0.00000	3.00000
00023	0007	LIVE1	20.00000	0.00000	3.00000
00024	0008	LIVE1	20.00000	0.00000	3.00000
00025	0009	LIVE1	20.00000	0.00000	8.00000
00026	0010	LIVE1	20.00000	0.00000	3.00000
00027	0011	LIVE1	20.00000	0.00000	2.00000
00028	0012	LIVE1	20.00000	0.00000	3.00000
00029	0013	LIVE1	20.00000	0.00000	3.00000
00030	0014	LIVE1	20.00000	0.00000	2.00000
00031	0015	LIVE1	20.00000	0.00000	3.00000
00032	0016	LIVE1	20.00000	0.00000	8.00000

Figure 8-12: PLX01—distributed loads on Level 01.

. USE PLX02

. LIST

00001	0001	DEAD1	2.50000	0.00000	7.00000
00002	0002	DEAD1	2.50000	0.00000	7.00000
00003	0003	DEAD1	2.50000	0.00000	7.00000
00004	0004	DEAD1	2.50000	0.00000	7.00000
00005	0005	DEAD1	2.50000	0.00000	7.00000
00006	0006	DEAD1	2.50000	0.00000	7.00000
00007	0007	DEAD1	2.50000	0.00000	7.00000
00008	0008	DEAD1	2.50000	0.00000	7.00000
00009	0001	SNOW1	2.50000	0.00000	7.00000
00010	0002	SNOW1	2.50000	0.00000	7.00000
00011	0003	SNOW1	2.50000	0.00000	7.00000
00012	0004	SNOW1	2.50000	0.00000	7.00000
00013	0005	SNOW1	2.50000	0.00000	7.00000
00014	0006	SNOW1	2.50000	0.00000	7.00000
00015	0007	SNOW1	2.50000	0.00000	7.00000
00016	0008	SNOW1	2.50000	0.00000	7.00000

Figure 8-13: PLX02—distributed loads on Level 02.

. USE CONCL01

. LIST STRUCTURE

STRUCTURE FOR FILE: CONCL01.DBF
 NUMBER OF RECORDS: 00008
 DATE OF LAST UPDATE: 00/00/00
 PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	004	
002	SOURCE1	C	001	
003	SOURCE2	C	004	
004	LOADCASE	C	005	
005	LOAD	N	010	005
006	POSITN1	N	010	005
** TOTAL **			00035	

. LIST

00001	0006	Z	0001	LIVE2	75.00000	1.00000
00002	0006	Z	0001	LIVE2	75.00000	4.00000
00003	0005	Z	0001	LIVE2	75.00000	3.00000
00004	0005	Z	0001	LIVE2	75.00000	6.00000
00005	0016	Z	0002	LIVE2	100.00000	3.00000
00006	0016	Z	0002	LIVE2	100.00000	5.00000
00007	0017	Z	0002	LIVE2	100.00000	3.00000
00008	0017	Z	0002	LIVE2	100.00000	5.00000

Figure 8-14: CONCL01—concentrated loads on Level 01.

```

. USE CASES
.
. LIST STRUCTURE
STRUCTURE FOR FILE:  CASES.DBF
NUMBER OF RECORDS:  00004
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      LOADCOMB   C      005
002      LOADCASE   C      005
003      FACTOR     N      010    005
** TOTAL **              00021
.
. LIST
00001 COMB1 DEAD1      1.25000
00002 COMB1 SNOW1      1.50000
00003 COMB1 LIVE1      1.50000
00004 COMB1 LIVE2      1.50000
.
.
.

```

Figure 8-15: CASES—loading combinations, load cases and load factors.

```

. USE BMLIST01
.
. LIST STRUCTURE
STRUCTURE FOR FILE: BMLIST01.DBF
NUMBER OF RECORDS: 00029
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      004
** TOTAL **          00005

```

```

. LIST
00001 0008
00002 0009
00003 0010
00004 0011
00005 0012
00006 0013
00007 0014
00008 0020
00009 0021
00010 0022
00011 0023
00012 XXXX
00013 0006
00014 0018
00015 0019
00016 XXXX
00017 0007
00018 0016
00019 0017
00020 XXXX
00021 0003
00022 0005
00023 0004
00024 0015
00025 XXXX
00026 0001
00027 0002
00028 XXXX
00029 XXXX

```

Figure 8-16: BMLIST01—levels of independence for beams on Level 01.

```

. USE BMLIST02
.
. LIST STRUCTURE
STRUCTURE FOR FILE: BMLIST02.DBF
NUMBER OF RECORDS: 00016
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      004
** TOTAL **              00005

```

```

. LIST
00001 0003
00002 0004
00003 0005
00004 0006
00005 0007
00006 0010
00007 0011
00008 0012
00009 0013
00010 XXXX
00011 0001
00012 0002
00013 0008
00014 0009
00015 XXXX
00016 XXXX

```

Figure 8-17: BMLIST02—levels of independence for beams on Level 02.

. USE BLX01

. LIST STRUCTURE

STRUCTURE FOR FILE: BLX01.DBF

NUMBER OF RECORDS: 00146

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	004	
002	SOURCE	C	001	
003	SOURCE	C	004	
004	LOADCASE	C	005	
005	LOADTYPE	C	001	
006	LOAD	N	010	005
007	POSITN1	N	010	005
008	POSITN2	N	010	005
** TOTAL **			00046	

. LIST

00001	0006	Z	0001	LIVE2	C	75.00000	1.00000	0.00000
00002	0006	Z	0001	LIVE2	C	75.00000	4.00000	0.00000
00003	0005	Z	0001	LIVE2	C	75.00000	3.00000	0.00000
00004	0005	Z	0001	LIVE2	C	75.00000	6.00000	0.00000
00005	0016	Z	0002	LIVE2	C	100.00000	3.00000	0.00000
00006	0016	Z	0002	LIVE2	C	100.00000	5.00000	0.00000
00007	0017	Z	0002	LIVE2	C	100.00000	3.00000	0.00000
00008	0017	Z	0002	LIVE2	C	100.00000	5.00000	0.00000
00009	0007	B	0006	LIVE2	C	75.00000	2.00000	0.00000
00010	0002	B	0006	LIVE2	C	75.00000	2.00000	0.00000
00011	0003	B	0007	LIVE2	C	37.50000	2.00000	0.00000
00012	0005	B	0007	LIVE2	C	37.50000	2.00000	0.00000
00013	0004	B	0016	LIVE2	C	100.00000	1.75000	0.00000
00014	0015	B	0016	LIVE2	C	100.00000	1.75000	0.00000
00015	0004	B	0017	LIVE2	C	100.00000	5.25000	0.00000
00016	0015	B	0017	LIVE2	C	100.00000	5.25000	0.00000
00017	0001	B	0005	LIVE2	C	80.35714	4.00000	0.00000
00018	0002	B	0005	LIVE2	C	107.14285	4.00000	0.00000
00019	0001	P	0001	DEAD1	U	5.00000	0.00000	4.00000
00020	0007	P	0001	DEAD1	U	5.00000	0.00000	4.00000
00021	0007	P	0002	DEAD1	U	4.16667	0.00000	2.00000
00022	0008	P	0002	DEAD1	U	4.16667	0.00000	2.00000
00023	0008	P	0003	DEAD1	U	4.16667	0.00000	2.00000
00024	0009	P	0003	DEAD1	U	4.16667	0.00000	2.00000
00025	0009	P	0004	DEAD1	U	4.16667	0.00000	2.00000
00026	0002	P	0004	DEAD1	U	4.16667	0.00000	2.00000
00027	0001	P	0005	DEAD1	U	4.37500	4.00000	7.00000
00028	0010	P	0005	DEAD1	U	4.37500	0.00000	3.00000
00029	0010	P	0006	DEAD1	U	4.37500	0.00000	3.00000
00030	0011	P	0006	DEAD1	U	4.37500	0.00000	3.00000
00031	0011	P	0007	DEAD1	U	4.37500	0.00000	3.00000
00032	0012	P	0007	DEAD1	U	4.37500	0.00000	3.00000
00033	0012	P	0008	DEAD1	U	4.37500	0.00000	3.00000
00034	0002	P	0008	DEAD1	U	4.37500	4.00000	7.00000

Figure 8-18: BLX01—external beam forces on Level 01.

00035	0013	F	0009	DEAD1	U	4.37500	0.00000	8.00000
00036	0016	F	0009	DEAD1	U	4.37500	0.00000	8.00000
00037	0016	F	0010	DEAD1	U	4.37500	0.00000	3.00000
00038	0020	F	0010	DEAD1	U	4.37500	0.00000	3.00000
00039	0016	F	0011	DEAD1	U	1.87500	3.00000	5.00000
00040	0022	F	0011	DEAD1	U	1.87500	0.00000	2.00000
00041	0016	F	0012	DEAD1	U	4.37500	5.00000	8.00000
00042	0021	F	0012	DEAD1	U	4.37500	0.00000	3.00000
00043	0020	F	0013	DEAD1	U	4.37500	0.00000	3.00000
00044	0017	F	0013	DEAD1	U	4.37500	0.00000	3.00000
00045	0023	F	0014	DEAD1	U	1.87500	0.00000	2.00000
00046	0017	F	0014	DEAD1	U	1.87500	3.00000	5.00000
00047	0021	F	0015	DEAD1	U	4.37500	0.00000	3.00000
00048	0017	F	0015	DEAD1	U	4.37500	5.00000	8.00000
00049	0017	F	0016	DEAD1	U	4.37500	0.00000	8.00000
00050	0014	F	0016	DEAD1	U	4.37500	0.00000	8.00000
00051	0003	B	0008	DEAD1	C	8.33334	3.66667	0.00000
00052	0006	B	0008	DEAD1	C	8.33334	1.66667	0.00000
00053	0003	B	0009	DEAD1	C	8.33334	5.33333	0.00000
00054	0006	B	0009	DEAD1	C	8.33334	3.33333	0.00000
00055	0005	B	0010	DEAD1	C	13.12500	1.75000	0.00000
00056	0004	B	0010	DEAD1	C	13.12500	1.75000	0.00000
00057	0005	B	0011	DEAD1	C	13.12500	3.50000	0.00000
00058	0004	B	0011	DEAD1	C	13.12500	3.50000	0.00000
00059	0005	B	0012	DEAD1	C	13.12500	5.25000	0.00000
00060	0004	B	0012	DEAD1	C	13.12500	5.25000	0.00000
00061	0004	B	0020	DEAD1	C	13.12500	3.50000	0.00000
00062	0018	B	0020	DEAD1	C	13.12500	1.75000	0.00000
00063	0019	B	0021	DEAD1	C	13.12500	1.75000	0.00000
00064	0015	B	0021	DEAD1	C	13.12500	3.50000	0.00000
00065	0018	B	0022	DEAD1	C	1.87500	0.75000	0.00000
00066	0019	B	0022	DEAD1	C	1.87500	0.75000	0.00000
00067	0018	B	0023	DEAD1	C	1.87500	2.75000	0.00000
00068	0019	B	0023	DEAD1	C	1.87500	2.75000	0.00000
00069	0007	B	0006	DEAD1	C	8.33333	2.00000	0.00000
00070	0002	B	0006	DEAD1	C	8.33333	2.00000	0.00000
00071	0016	B	0018	DEAD1	C	8.43750	3.00000	0.00000
00072	0017	B	0018	DEAD1	C	8.43749	3.00000	0.00000
00073	0016	B	0019	DEAD1	C	8.43750	5.00000	0.00000
00074	0017	B	0019	DEAD1	C	8.43749	5.00000	0.00000
00075	0003	B	0007	DEAD1	C	20.41667	2.00000	0.00000
00076	0005	B	0007	DEAD1	C	16.25000	2.00000	0.00000
00077	0004	B	0016	DEAD1	C	40.93750	1.75000	0.00000
00078	0015	B	0016	DEAD1	C	40.93750	1.75000	0.00000
00079	0004	B	0017	DEAD1	C	40.93749	5.25000	0.00000
00080	0015	B	0017	DEAD1	C	40.93749	5.25000	0.00000
00081	0001	B	0005	DEAD1	C	31.29464	4.00000	0.00000
00082	0002	B	0005	DEAD1	C	24.33035	4.00000	0.00000
00083	0001	F	0001	LIVE1	U	20.00000	0.00000	4.00000
00084	0007	F	0001	LIVE1	U	20.00000	0.00000	4.00000
00085	0007	F	0002	LIVE1	U	16.66670	0.00000	2.00000
00086	0008	F	0002	LIVE1	U	16.66670	0.00000	2.00000
00087	0008	F	0003	LIVE1	U	16.66670	0.00000	2.00000

Figure 8-18 (continued.)

00088	0009	F	0003	LIVE1	U	16.66670	0.00000	2.00000
00089	0009	F	0004	LIVE1	U	16.66670	0.00000	2.00000
00090	0002	F	0004	LIVE1	U	16.66670	0.00000	2.00000
00091	0001	F	0005	LIVE1	U	17.50000	4.00000	7.00000
00092	0010	F	0005	LIVE1	U	17.50000	0.00000	3.00000
00093	0010	F	0006	LIVE1	U	17.50000	0.00000	3.00000
00094	0011	F	0006	LIVE1	U	17.50000	0.00000	3.00000
00095	0011	F	0007	LIVE1	U	17.50000	0.00000	3.00000
00096	0012	F	0007	LIVE1	U	17.50000	0.00000	3.00000
00097	0012	F	0008	LIVE1	U	17.50000	0.00000	3.00000
00098	0002	F	0008	LIVE1	U	17.50000	4.00000	7.00000
00099	0013	F	0009	LIVE1	U	17.50000	0.00000	8.00000
00100	0016	F	0009	LIVE1	U	17.50000	0.00000	8.00000
00101	0016	F	0010	LIVE1	U	17.50000	0.00000	3.00000
00102	0020	F	0010	LIVE1	U	17.50000	0.00000	3.00000
00103	0016	F	0011	LIVE1	U	7.50000	3.00000	5.00000
00104	0022	F	0011	LIVE1	U	7.50000	0.00000	2.00000
00105	0016	F	0012	LIVE1	U	17.50000	5.00000	8.00000
00106	0021	F	0012	LIVE1	U	17.50000	0.00000	3.00000
00107	0020	F	0013	LIVE1	U	17.50000	0.00000	3.00000
00108	0017	F	0013	LIVE1	U	17.50000	0.00000	3.00000
00109	0023	F	0014	LIVE1	U	7.50000	0.00000	2.00000
00110	0017	F	0014	LIVE1	U	7.50000	3.00000	5.00000
00111	0021	F	0015	LIVE1	U	17.50000	0.00000	3.00000
00112	0017	F	0015	LIVE1	U	17.50000	5.00000	8.00000
00113	0017	F	0016	LIVE1	U	17.50000	0.00000	8.00000
00114	0014	F	0016	LIVE1	U	17.50000	0.00000	8.00000
00115	0003	B	0008	LIVE1	C	33.33340	3.66667	0.00000
00116	0006	B	0008	LIVE1	C	33.33340	1.66667	0.00000
00117	0003	B	0009	LIVE1	C	33.33340	5.33333	0.00000
00118	0006	B	0009	LIVE1	C	33.33340	3.33333	0.00000
00119	0005	B	0010	LIVE1	C	52.50000	1.75000	0.00000
00120	0004	B	0010	LIVE1	C	52.50000	1.75000	0.00000
00121	0005	B	0011	LIVE1	C	52.50000	3.50000	0.00000
00122	0004	B	0011	LIVE1	C	52.50000	3.50000	0.00000
00123	0005	B	0012	LIVE1	C	52.50000	5.25000	0.00000
00124	0004	B	0012	LIVE1	C	52.50000	5.25000	0.00000
00125	0004	B	0020	LIVE1	C	52.50000	3.50000	0.00000
00126	0018	B	0020	LIVE1	C	52.50000	1.75000	0.00000
00127	0019	B	0021	LIVE1	C	52.50000	1.75000	0.00000
00128	0015	B	0021	LIVE1	C	52.50000	3.50000	0.00000
00129	0018	B	0022	LIVE1	C	7.50000	0.75000	0.00000
00130	0019	B	0022	LIVE1	C	7.50000	0.75000	0.00000
00131	0018	B	0023	LIVE1	C	7.50000	2.75000	0.00000
00132	0019	B	0023	LIVE1	C	7.50000	2.75000	0.00000
00133	0007	B	0006	LIVE1	C	33.33339	2.00000	0.00000
00134	0002	B	0006	LIVE1	C	33.33339	2.00000	0.00000
00135	0016	B	0018	LIVE1	C	33.75000	3.00000	0.00000
00136	0017	B	0018	LIVE1	C	33.74999	3.00000	0.00000
00137	0016	B	0019	LIVE1	C	33.75000	5.00000	0.00000
00138	0017	B	0019	LIVE1	C	33.74999	5.00000	0.00000
00139	0003	B	0007	LIVE1	C	81.66674	2.00000	0.00000
00140	0005	B	0007	LIVE1	C	85.00004	2.00000	0.00000

Figure 8-18 (continued.)

00141	0004	B	0016	LIVE1	C	163.75000	1.75000	0.00000
00142	0015	B	0016	LIVE1	C	163.75000	1.75000	0.00000
00143	0004	B	0017	LIVE1	C	163.74998	5.25000	0.00000
00144	0015	B	0017	LIVE1	C	163.74998	5.25000	0.00000
00145	0001	B	0005	LIVE1	C	125.17860	4.00000	0.00000
00146	0002	B	0005	LIVE1	C	97.32143	4.00000	0.00000

Figure 8-18 (continued.)

USE BLX02

. LIST STRUCTURE
 STRUCTURE FOR FILE: BLX02.DBF
 NUMBER OF RECORDS: 00056
 DATE OF LAST UPDATE: 00/00/00
 PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	004	
002	SOURCE	C	001	
003	SOURCE	C	004	
004	LOADCASE	C	005	
005	LOADTYPE	C	001	
006	LOAD	N	010	005
007	POSITN1	N	010	005
008	POSITN2	N	010	005
** TOTAL **			00046	

. LIST

00001	0003	P	0001	SNOW1	U	2.18750	0.00000	7.00000
00002	0005	P	0001	SNOW1	U	2.18750	0.00000	7.00000
00003	0005	P	0002	SNOW1	U	2.18750	0.00000	7.00000
00004	0006	P	0002	SNOW1	U	2.18750	0.00000	7.00000
00005	0006	P	0003	SNOW1	U	2.18750	0.00000	7.00000
00006	0007	P	0003	SNOW1	U	2.18750	0.00000	7.00000
00007	0007	P	0004	SNOW1	U	2.18750	0.00000	7.00000
00008	0004	P	0004	SNOW1	U	2.18750	0.00000	7.00000
00009	0004	P	0005	SNOW1	U	2.50000	0.00000	7.00000
00010	0011	P	0005	SNOW1	U	2.50000	0.00000	7.00000
00011	0011	P	0006	SNOW1	U	2.50000	0.00000	7.00000
00012	0012	P	0006	SNOW1	U	2.50000	0.00000	7.00000
00013	0012	P	0007	SNOW1	U	2.50000	0.00000	7.00000
00014	0013	P	0007	SNOW1	U	2.50000	0.00000	7.00000
00015	0013	P	0008	SNOW1	U	2.50000	0.00000	7.00000
00016	0010	P	0008	SNOW1	U	2.50000	0.00000	7.00000
00017	0001	B	0005	SNOW1	C	15.31250	1.75000	0.00000
00018	0002	B	0005	SNOW1	C	15.31250	1.75000	0.00000
00019	0001	B	0006	SNOW1	C	15.31250	3.50000	0.00000
00020	0002	B	0006	SNOW1	C	15.31250	3.50000	0.00000
00021	0001	B	0007	SNOW1	C	15.31250	5.25000	0.00000
00022	0002	B	0007	SNOW1	C	15.31250	5.25000	0.00000
00023	0008	B	0011	SNOW1	C	17.50000	2.00000	0.00000
00024	0009	B	0011	SNOW1	C	17.50000	2.00000	0.00000
00025	0008	B	0012	SNOW1	C	17.50000	4.00000	0.00000
00026	0009	B	0012	SNOW1	C	17.50000	4.00000	0.00000
00027	0008	B	0013	SNOW1	C	17.50000	6.00000	0.00000
00028	0009	B	0013	SNOW1	C	17.50000	6.00000	0.00000
00029	0003	P	0001	DEAD1	U	2.18750	0.00000	7.00000
00030	0005	P	0001	DEAD1	U	2.18750	0.00000	7.00000
00031	0005	P	0002	DEAD1	U	2.18750	0.00000	7.00000
00032	0006	P	0002	DEAD1	U	2.18750	0.00000	7.00000
00033	0006	P	0003	DEAD1	U	2.18750	0.00000	7.00000
00034	0007	P	0003	DEAD1	U	2.18750	0.00000	7.00000

Figure 8-19: BLX02—external beam forces on Level 02.

USE BLX02

. LIST STRUCTURE
 STRUCTURE FOR FILE: BLX02.DBF
 NUMBER OF RECORDS: 00056
 DATE OF LAST UPDATE: 00/00/00
 PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	004	
002	SOURCET	C	001	
003	SOURCEN	C	004	
004	LOADCASE	C	005	
005	LOADTYPE	C	001	
006	LOAD	N	010	005
007	POSITN1	N	010	005
008	POSITN2	N	010	005
** TOTAL **			00046	

. LIST

00001	0003	P	0001	SNOW1	U	2.18750	0.00000	7.00000
00002	0005	P	0001	SNOW1	U	2.18750	0.00000	7.00000
00003	0005	P	0002	SNOW1	U	2.18750	0.00000	7.00000
00004	0006	P	0002	SNOW1	U	2.18750	0.00000	7.00000
00005	0006	P	0003	SNOW1	U	2.18750	0.00000	7.00000
00006	0007	P	0003	SNOW1	U	2.18750	0.00000	7.00000
00007	0007	P	0004	SNOW1	U	2.18750	0.00000	7.00000
00008	0004	P	0004	SNOW1	U	2.18750	0.00000	7.00000
00009	0004	P	0005	SNOW1	U	2.50000	0.00000	7.00000
00010	0011	P	0005	SNOW1	U	2.50000	0.00000	7.00000
00011	0011	P	0006	SNOW1	U	2.50000	0.00000	7.00000
00012	0012	P	0006	SNOW1	U	2.50000	0.00000	7.00000
00013	0012	P	0007	SNOW1	U	2.50000	0.00000	7.00000
00014	0013	P	0007	SNOW1	U	2.50000	0.00000	7.00000
00015	0013	P	0008	SNOW1	U	2.50000	0.00000	7.00000
00016	0010	P	0008	SNOW1	U	2.50000	0.00000	7.00000
00017	0001	B	0005	SNOW1	C	15.31250	1.75000	0.00000
00018	0002	B	0005	SNOW1	C	15.31250	1.75000	0.00000
00019	0001	B	0006	SNOW1	C	15.31250	3.50000	0.00000
00020	0002	B	0006	SNOW1	C	15.31250	3.50000	0.00000
00021	0001	B	0007	SNOW1	C	15.31250	5.25000	0.00000
00022	0002	B	0007	SNOW1	C	15.31250	5.25000	0.00000
00023	0008	B	0011	SNOW1	C	17.50000	2.00000	0.00000
00024	0009	B	0011	SNOW1	C	17.50000	2.00000	0.00000
00025	0008	B	0012	SNOW1	C	17.50000	4.00000	0.00000
00026	0009	B	0012	SNOW1	C	17.50000	4.00000	0.00000
00027	0008	B	0013	SNOW1	C	17.50000	6.00000	0.00000
00028	0009	B	0013	SNOW1	C	17.50000	6.00000	0.00000
00029	0003	P	0001	DEAD1	U	2.18750	0.00000	7.00000
00030	0005	P	0001	DEAD1	U	2.18750	0.00000	7.00000
00031	0005	P	0002	DEAD1	U	2.18750	0.00000	7.00000
00032	0006	P	0002	DEAD1	U	2.18750	0.00000	7.00000
00033	0006	P	0003	DEAD1	U	2.18750	0.00000	7.00000
00034	0007	P	0003	DEAD1	U	2.18750	0.00000	7.00000

Figure 8-19: BLX02—external beam forces on Level 02.

00035	0007	P	0004	DEAD1	U	2.18750	0.00000	7.00000
00036	0004	P	0004	DEAD1	U	2.18750	0.00000	7.00000
00037	0004	P	0005	DEAD1	U	2.50000	0.00000	7.00000
00038	0011	P	0005	DEAD1	U	2.50000	0.00000	7.00000
00039	0011	P	0006	DEAD1	U	2.50000	0.00000	7.00000
00040	0012	P	0006	DEAD1	U	2.50000	0.00000	7.00000
00041	0012	P	0007	DEAD1	U	2.50000	0.00000	7.00000
00042	0013	P	0007	DEAD1	U	2.50000	0.00000	7.00000
00043	0013	P	0008	DEAD1	U	2.50000	0.00000	7.00000
00044	0010	P	0008	DEAD1	U	2.50000	0.00000	7.00000
00045	0001	B	0005	DEAD1	C	15.31250	1.75000	0.00000
00046	0002	B	0005	DEAD1	C	15.31250	1.75000	0.00000
00047	0001	B	0006	DEAD1	C	15.31250	3.50000	0.00000
00048	0002	B	0006	DEAD1	C	15.31250	3.50000	0.00000
00049	0001	B	0007	DEAD1	C	15.31250	5.25000	0.00000
00050	0002	B	0007	DEAD1	C	15.31250	5.25000	0.00000
00051	0008	B	0011	DEAD1	C	17.50000	2.00000	0.00000
00052	0009	B	0011	DEAD1	C	17.50000	2.00000	0.00000
00053	0008	B	0012	DEAD1	C	17.50000	4.00000	0.00000
00054	0009	B	0012	DEAD1	C	17.50000	4.00000	0.00000
00055	0008	B	0013	DEAD1	C	17.50000	6.00000	0.00000
00056	0009	B	0013	DEAD1	C	17.50000	6.00000	0.00000

Figure 8-19 (continued.)

USE CLX

. LIST STRUCTURE
 STRUCTURE FOR FILE: CLX.DBF
 NUMBER OF RECORDS: 00056
 DATE OF LAST UPDATE: 00/00/00
 PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	004	
002	SOURCET	C	001	
003	SOURCEN	C	004	
004	SOURCEL	C	002	
005	LOADTYPE	C	001	
006	LOADCASE	C	005	
007	LOAD	N	010	005
** TOTAL **			00028	

. LIST

00001	0004	B	0003	02	C	SNOW1	7.65625
00002	0001	B	0003	02	C	SNOW1	7.65625
00003	0005	B	0004	02	C	SNOW1	16.40625
00004	0002	B	0004	02	C	SNOW1	16.40625
00005	0006	B	0010	02	C	SNOW1	8.75000
00006	0003	B	0010	02	C	SNOW1	8.75000
00007	0004	B	0001	02	C	SNOW1	22.96875
00008	0005	B	0001	02	C	SNOW1	22.96875
00009	0001	B	0002	02	C	SNOW1	22.96875
00010	0002	B	0002	02	C	SNOW1	22.96875
00011	0005	B	0008	02	C	SNOW1	26.25000
00012	0006	B	0008	02	C	SNOW1	26.25000
00013	0002	B	0009	02	C	SNOW1	26.25000
00014	0003	B	0009	02	C	SNOW1	26.25000
00015	0004	B	0003	02	C	DEAD1	7.65625
00016	0001	B	0003	02	C	DEAD1	7.65625
00017	0005	B	0004	02	C	DEAD1	16.40625
00018	0002	B	0004	02	C	DEAD1	16.40625
00019	0006	B	0010	02	C	DEAD1	8.75000
00020	0003	B	0010	02	C	DEAD1	8.75000
00021	0004	B	0001	02	C	DEAD1	22.96875
00022	0005	B	0001	02	C	DEAD1	22.96875
00023	0001	B	0002	02	C	DEAD1	22.96875
00024	0002	B	0002	02	C	DEAD1	22.96875
00025	0005	B	0008	02	C	DEAD1	26.25000
00026	0006	B	0008	02	C	DEAD1	26.25000
00027	0002	B	0009	02	C	DEAD1	26.25000
00028	0003	B	0009	02	C	DEAD1	26.25000
00029	0005	B	0013	01	C	DEAD1	17.50000
00030	0006	B	0013	01	C	DEAD1	17.50000
00031	0002	B	0014	01	C	DEAD1	17.50000
00032	0003	B	0014	01	C	DEAD1	17.50000
00033	0004	B	0003	01	C	DEAD1	20.53572
00034	0001	B	0003	01	C	DEAD1	16.54762
00035	0005	B	0004	01	C	DEAD1	67.18749

Figure 8-20: CLX—external column forces.

00036	0002	B	0004	01	C	DEAD1	67.18749
00037	0006	B	0015	01	C	DEAD1	47.49999
00038	0003	B	0015	01	C	DEAD1	47.49999
00039	0004	B	0001	01	C	DEAD1	30.51020
00040	0005	B	0001	01	C	DEAD1	33.90943
00041	0001	B	0002	01	C	DEAD1	26.33503
00042	0002	B	0002	01	C	DEAD1	27.78698
00043	0005	B	0013	01	C	LIVE1	70.00000
00044	0006	B	0013	01	C	LIVE1	70.00000
00045	0002	B	0014	01	C	LIVE1	70.00000
00046	0003	B	0014	01	C	LIVE1	70.00000
00047	0004	B	0003	01	C	LIVE1	82.14295
00048	0001	B	0003	01	C	LIVE1	66.19058
00049	0005	B	0004	01	C	LIVE1	268.74999
00050	0002	B	0004	01	C	LIVE1	268.74998
00051	0006	B	0015	01	C	LIVE1	189.99999
00052	0003	B	0015	01	C	LIVE1	189.99998
00053	0004	B	0001	01	C	LIVE1	122.04082
00054	0005	B	0001	01	C	LIVE1	135.63777
00055	0001	B	0002	01	C	LIVE1	105.34023
00056	0002	B	0002	01	C	LIVE1	111.14798

Figure 8-20 (continued.)

USE BMDIAG01

. LIST STRUCTURE

STRUCTURE FOR FILE: BMDIAG01.DBF

NUMBER OF RECORDS: 00092

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	004	
002	LOADCOMB	C	005	
003	POSITN	N	010	005
004	SHEAR1	N	010	005
005	SHEAR2	N	010	005
006	MOMENT1	N	010	005
** TOTAL **			00050	

. LIST

00001	0001	COMB1	0.00000	272.85715	272.85715	0.00000
00002	0001	COMB1	4.00000	127.85715	-219.56475	801.42863
00003	0001	COMB1	7.00000	-423.47100	-423.47100	0.00000
00004	0002	COMB1	0.00000	340.16380	340.16380	0.00000
00005	0002	COMB1	2.00000	279.74704	106.83030	619.91085
00006	0002	COMB1	4.00000	46.41354	-290.69579	833.57147
00007	0002	COMB1	7.00000	-476.47718	-476.47718	0.00000
00008	0003	COMB1	0.00000	189.06265	189.06265	0.00000
00009	0003	COMB1	2.00000	189.06265	-15.20829	378.12530
00010	0003	COMB1	3.66667	-15.20829	-75.62506	352.77809
00011	0003	COMB1	5.33333	-75.62506	-136.04183	226.73683
00012	0003	COMB1	7.00000	-136.04183	-136.04183	0.00000
00013	0004	COMB1	0.00000	637.10936	637.10936	0.00000
00014	0004	COMB1	1.75000	637.10936	95.15624	1114.94138
00015	0004	COMB1	3.50000	95.15624	-95.15626	1281.46480
00016	0004	COMB1	5.25000	-95.15626	-637.10934	1114.94134
00017	0004	COMB1	7.00000	-637.10934	-637.10934	0.00000
00018	0005	COMB1	0.00000	347.42191	347.42191	0.00000
00019	0005	COMB1	1.75000	347.42191	252.26566	607.98835
00020	0005	COMB1	2.00000	252.26566	78.20310	671.05477
00021	0005	COMB1	3.00000	78.20310	-34.29689	749.25788
00022	0005	COMB1	3.50000	-34.29689	-129.45314	732.10943
00023	0005	COMB1	5.25000	-129.45314	-224.60939	505.56643
00024	0005	COMB1	6.00000	-224.60939	-337.10939	337.10939
00025	0005	COMB1	7.00000	-337.10939	-337.10939	0.00000
00026	0006	COMB1	0.00000	172.91676	172.91676	0.00000
00027	0006	COMB1	1.00000	172.91676	60.41676	172.91676
00028	0006	COMB1	1.66667	60.41676	0.00000	213.19481
00029	0006	COMB1	3.33333	0.00000	-60.41677	213.19481
00030	0006	COMB1	4.00000	-60.41677	-172.91677	172.91676
00031	0006	COMB1	5.00000	-172.91677	-172.91677	0.00000
00032	0007	COMB1	0.00000	204.27094	204.27094	0.00000
00033	0007	COMB1	2.00000	71.35418	-101.56256	275.62512
00034	0007	COMB1	4.00000	-174.06256	-174.06256	0.00000
00035	0008	COMB1	0.00000	60.41676	60.41676	0.00000
00036	0008	COMB1	1.00000	0.00000	0.00000	30.20838

Figure 8-31: BMDIAG01—beam shears and bending moments on Level 01.

00037	0008	COMB1	2.00000	-60.41676	-60.41676	0.00000
00038	0009	COMB1	0.00000	60.41676	60.41676	0.00000
00039	0009	COMB1	1.00000	0.00000	0.00000	30.20838
00040	0009	COMB1	2.00000	-60.41676	-60.41676	0.00000
00041	0010	COMB1	0.00000	95.15625	95.15625	0.00000
00042	0010	COMB1	1.50000	0.00000	0.00000	71.36718
00043	0010	COMB1	3.00000	-95.15625	-95.15625	0.00000
00044	0011	COMB1	0.00000	95.15625	95.15625	0.00000
00045	0011	COMB1	1.50000	0.00000	0.00000	71.36718
00046	0011	COMB1	3.00000	-95.15625	-95.15625	0.00000
00047	0012	COMB1	0.00000	95.15625	95.15625	0.00000
00048	0012	COMB1	1.50000	0.00000	0.00000	71.36718
00049	0012	COMB1	3.00000	-95.15625	-95.15625	0.00000
00050	0013	COMB1	0.00000	126.87500	126.87500	0.00000
00051	0013	COMB1	4.00000	0.00000	0.00000	253.75000
00052	0013	COMB1	8.00000	-126.87500	-126.87500	0.00000
00053	0014	COMB1	0.00000	126.87500	126.87500	0.00000
00054	0014	COMB1	4.00000	0.00000	0.00000	253.75000
00055	0014	COMB1	8.00000	-126.87500	-126.87500	0.00000
00056	0015	COMB1	0.00000	494.37498	494.37498	0.00000
00057	0015	COMB1	1.75000	494.37498	47.57811	865.15622
00058	0015	COMB1	3.50000	47.57811	-47.57813	948.41792
00059	0015	COMB1	5.25000	-47.57813	-494.37496	865.15618
00060	0015	COMB1	7.00000	-494.37496	-494.37496	0.00000
00061	0016	COMB1	0.00000	446.79686	446.79686	0.00000
00062	0016	COMB1	1.00000	256.48436	45.31249	1054.92185
00063	0016	COMB1	3.99999	0.00045	0.00045	1077.57811
00064	0016	COMB1	5.00000	-45.31250	-256.48437	1054.92185
00065	0016	COMB1	8.00000	-446.79687	-446.79687	0.00000
00066	0017	COMB1	0.00000	446.79683	446.79683	0.00000
00067	0017	COMB1	3.00000	256.48433	45.31249	1054.92176
00068	0017	COMB1	3.99999	0.00045	0.00045	1077.57802
00069	0017	COMB1	5.00000	-45.31250	-256.48434	1054.92176
00070	0017	COMB1	8.00000	-446.79684	-446.79684	0.00000
00071	0018	COMB1	0.00000	61.17187	61.17187	0.00000
00072	0018	COMB1	0.75000	61.17187	47.57812	45.87890
00073	0018	COMB1	1.75000	47.57812	-47.57812	93.45703
00074	0018	COMB1	2.75000	-47.57812	-61.17187	45.87890
00075	0018	COMB1	3.50000	-61.17187	-61.17187	0.00000
00076	0019	COMB1	0.00000	61.17187	61.17187	0.00000
00077	0019	COMB1	0.75000	61.17187	47.57812	45.87890
00078	0019	COMB1	1.75000	47.57812	-47.57812	93.45703
00079	0019	COMB1	2.75000	-47.57812	-61.17187	45.87890
00080	0019	COMB1	3.50000	-61.17187	-61.17187	0.00000
00081	0020	COMB1	0.00000	95.15625	95.15625	0.00000
00082	0020	COMB1	1.50000	0.00000	0.00000	71.36718
00083	0020	COMB1	3.00000	-95.15625	-95.15625	0.00000
00084	0021	COMB1	0.00000	95.15625	95.15625	0.00000
00085	0021	COMB1	1.50000	0.00000	0.00000	71.36718
00086	0021	COMB1	3.00000	-95.15625	-95.15625	0.00000
00087	0022	COMB1	0.00000	13.59375	13.59375	0.00000
00088	0022	COMB1	1.00000	0.00000	0.00000	6.79687
00089	0022	COMB1	2.00000	-13.59375	-13.59375	0.00000

Figure 8-21 (continued.)

00090	0023	COMB1	0.00000	13.59375	13.59375	0.00000
00091	0023	COMB1	1.00000	0.00000	0.00000	6.79687
00092	0023	COMB1	2.00000	-13.59375	-13.59375	0.00000

Figure 8-21 (continued.)

USE BMDIAG02

. LIST STRUCTURE

STRUCTURE FOR FILE: BMDIAG02.DBF

NUMBER OF RECORDS: 00047

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	004	
002	LOADCOMB	C	005	
003	POSITN	N	010	005
004	SHEAR1	N	010	005
005	SHEAR2	N	010	005
006	MOMENT1	N	010	005
** TOTAL **			00050	

. LIST

00001	0001	COMB1	0.00000	63.16405	63.16405	0.00000
00002	0001	COMB1	1.75000	63.16405	21.05468	110.53709
00003	0001	COMB1	3.50000	21.05468	-21.05468	147.38279
00004	0001	COMB1	5.25000	-21.05468	-63.16405	110.53709
00005	0001	COMB1	7.00000	-63.16405	-63.16405	0.00000
00006	0002	COMB1	0.00000	63.16405	63.16405	0.00000
00007	0002	COMB1	1.75000	63.16405	21.05468	110.53709
00008	0002	COMB1	3.50000	21.05468	-21.05468	147.38279
00009	0002	COMB1	5.25000	-21.05468	-63.16405	110.53709
00010	0002	COMB1	7.00000	-63.16405	-63.16405	0.00000
00011	0003	COMB1	0.00000	21.05467	21.05467	0.00000
00012	0003	COMB1	3.50000	0.00000	0.00000	36.84567
00013	0003	COMB1	7.00000	-21.05467	-21.05467	0.00000
00014	0004	COMB1	0.00000	45.11717	45.11717	0.00000
00015	0004	COMB1	3.50000	0.00000	0.00000	78.95504
00016	0004	COMB1	7.00000	-45.11717	-45.11717	0.00000
00017	0005	COMB1	0.00000	42.10934	42.10934	0.00000
00018	0005	COMB1	3.50000	0.00000	0.00000	73.69134
00019	0005	COMB1	7.00000	-42.10934	-42.10934	0.00000
00020	0006	COMB1	0.00000	42.10934	42.10934	0.00000
00021	0006	COMB1	3.50000	0.00000	0.00000	73.69134
00022	0006	COMB1	7.00000	-42.10934	-42.10934	0.00000
00023	0007	COMB1	0.00000	42.10934	42.10934	0.00000
00024	0007	COMB1	3.50000	0.00000	0.00000	73.69134
00025	0007	COMB1	7.00000	-42.10934	-42.10934	0.00000
00026	0008	COMB1	0.00000	72.18750	72.18750	0.00000
00027	0008	COMB1	2.00000	72.18750	24.06250	144.37500
00028	0008	COMB1	4.00000	24.06250	-24.06250	192.50000
00029	0008	COMB1	6.00000	-24.06250	-72.18750	144.37500
00030	0008	COMB1	8.00000	-72.18750	-72.18750	0.00000
00031	0009	COMB1	0.00000	72.18750	72.18750	0.00000
00032	0009	COMB1	2.00000	72.18750	24.06250	144.37500
00033	0009	COMB1	4.00000	24.06250	-24.06250	192.50000
00034	0009	COMB1	6.00000	-24.06250	-72.18750	144.37500
00035	0009	COMB1	8.00000	-72.18750	-72.18750	0.00000
00036	0010	COMB1	0.00000	24.06250	24.06250	0.00000

Figure 8-22: BMDIAG02—beam shears and bending moments on Level 02.

00037	0010	COMB1	3.50000	0.00000	0.00000	42.10937
00038	0010	COMB1	7.00000	-24.06250	-24.06250	0.00000
00039	0011	COMB1	0.00000	48.12500	48.12500	0.00000
00040	0011	COMB1	3.50000	0.00000	0.00000	84.21875
00041	0011	COMB1	7.00000	-48.12500	-48.12500	0.00000
00042	0012	COMB1	0.00000	48.12500	48.12500	0.00000
00043	0012	COMB1	3.50000	0.00000	0.00000	84.21875
00044	0012	COMB1	7.00000	-48.12500	-48.12500	0.00000
00045	0013	COMB1	0.00000	48.12500	48.12500	0.00000
00046	0013	COMB1	3.50000	0.00000	0.00000	84.21875
00047	0013	COMB1	7.00000	-48.12500	-48.12500	0.00000

Figure 8-22 (continued.)

```

USE MOMENT01
. LIST STRUCTURE
STRUCTURE FOR FILE: MOMENT01.DBF
NUMBER OF RECORDS: 00023
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      004
002      LOADCOMB   C      005
003      SHEAR      N      010    005
004      MOMENTP    N      010    005
005      MOMENTN    N      010    005
** TOTAL **          00040
. LIST
00001 0001 COMB1 423.47100 801.42863 0.00000
00002 0002 COMB1 476.47718 833.57147 0.00000
00003 0003 COMB1 189.06265 378.12530 0.00000
00004 0004 COMB1 637.10936 1281.46480 0.00000
00005 0005 COMB1 347.42191 749.25788 0.00000
00006 0006 COMB1 172.91677 213.19481 0.00000
00007 0007 COMB1 234.47932 275.62512 0.00000
00008 0008 COMB1 60.41676 30.20838 0.00000
00009 0009 COMB1 60.41676 30.20838 0.00000
00010 0010 COMB1 95.15625 71.36718 0.00000
00011 0011 COMB1 95.15625 71.36718 0.00000
00012 0012 COMB1 95.15625 71.36718 0.00000
00013 0013 COMB1 126.87500 253.75000 0.00000
00014 0014 COMB1 126.87500 253.75000 0.00000
00015 0015 COMB1 494.37498 948.41792 0.00000
00016 0016 COMB1 646.17187 1077.57811 0.00000
00017 0017 COMB1 646.17184 1077.57802 0.00000
00018 0018 COMB1 61.17187 93.45703 0.00000
00019 0019 COMB1 61.17187 93.45703 0.00000
00020 0020 COMB1 95.15625 71.36718 0.00000
00021 0021 COMB1 95.15625 71.36718 0.00000
00022 0022 COMB1 13.59375 6.79687 0.00000
00023 0023 COMB1 13.59375 6.79687 0.00000

```

Figure 8-23: MOMENT01—maximum beam shears and bending moments on Level 01.

USE MOMENT02

. LIST STRUCTURE
 STRUCTURE FOR FILE: MOMENT02.DBF
 NUMBER OF RECORDS: 00013
 DATE OF LAST UPDATE: 00/00/00
 PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	004	
002	LOADCOMB	C	005	
003	SHEAR	N	010	005
004	MOMENTP	N	010	005
005	MOMENTN	N	010	005
** TOTAL **			00040	

. LIST

00001	0001	COMB1	63.16405	147.38279	0.00000
00002	0002	COMB1	63.16405	147.38279	0.00000
00003	0003	COMB1	21.05467	36.84567	0.00000
00004	0004	COMB1	45.11717	78.95504	0.00000
00005	0005	COMB1	42.10934	73.69134	0.00000
00006	0006	COMB1	42.10934	73.69134	0.00000
00007	0007	COMB1	42.10934	73.69134	0.00000
00008	0008	COMB1	72.18750	192.50000	0.00000
00009	0009	COMB1	72.18750	192.50000	0.00000
00010	0010	COMB1	24.06250	42.10937	0.00000
00011	0011	COMB1	48.12500	84.21875	0.00000
00012	0012	COMB1	48.12500	84.21875	0.00000
00013	0013	COMB1	48.12500	84.21875	0.00000

Figure 8-24: MOMENT02—maximum beam shears and bending moments on Level 02.

```

. USE BMSEL
. LIST STRUCTURE
STRUCTURE FOR FILE: BMSEL.DBF
NUMBER OF RECORDS: 00021
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD NAME TYPE WIDTH DEC
001 SECTION C 009
002 LU M 010 005
003 VR M 010 005
004 L0000 M 010 005
005 L0015 M 010 005
006 L0020 M 010 005
007 L0025 M 010 005
008 L0030 M 010 005
009 L0040 M 010 005
** TOTAL ** 00090

. LIST
00001 C200X1700 1.21000 203.00000 35.90000 33.80000 30.10000 26.30000 22.30000 18.00000
00002 C250X2300 1.34000 276.00000 59.10000 57.30000 51.40000 45.20000 38.90000 32.40000
00003 W200X2700 2.05000 214.00000 75.30000 75.30000 75.30000 70.80000 65.40000 54.00000
00004 W360X3300 1.72000 361.00000 146.00000 146.00000 139.00000 124.00000 108.00000 70.30000
00005 W250X3300 2.18000 280.00000 114.00000 114.00000 114.00000 110.00000 102.00000 84.00000
00006 W410X3900 1.86000 448.00000 197.00000 197.00000 193.00000 175.00000 155.00000 105.00000
00007 W360X5100 2.50000 455.00000 241.00000 241.00000 241.00000 241.00000 227.00000 195.00000
00008 W410X5400 2.48000 539.00000 283.00000 283.00000 283.00000 283.00000 266.00000 225.00000
00009 W410X6000 2.58000 558.00000 321.00000 321.00000 321.00000 321.00000 306.00000 264.00000
00010 W460X6100 2.58000 650.00000 354.00000 354.00000 354.00000 354.00000 336.00000 288.00000
00011 W330X8200 2.73000 733.00000 445.00000 445.00000 445.00000 445.00000 433.00000 380.00000
00012 W330X8200 2.84000 894.00000 559.00000 559.00000 559.00000 559.00000 551.00000 487.00000
00013 W330X9200 2.93000 969.00000 637.00000 637.00000 637.00000 637.00000 633.00000 545.00000
00014 W610X1010 3.11000 1130.00000 783.00000 783.00000 783.00000 783.00000 783.00000 713.00000
00015 W610X1130 3.18000 1210.00000 888.00000 888.00000 888.00000 888.00000 888.00000 818.00000
00016 W610X1250 3.25000 1300.00000 991.00000 991.00000 991.00000 991.00000 991.00000 923.00000
00017 W610X1400 3.32000 1440.00000 1120.00000 1120.00000 1120.00000 1120.00000 1120.00000 1050.00000
00018 W610X1550 4.74000 1380.00000 1280.00000 1280.00000 1280.00000 1280.00000 1280.00000 1280.00000
00019 W610X1740 4.83000 1540.00000 1450.00000 1450.00000 1450.00000 1450.00000 1450.00000 1450.00000
00020 WF800X164 4.35000 1040.00000 1780.00000 1780.00000 1780.00000 1780.00000 1780.00000 1780.00000
00021 WF800X198 5.97000 1040.00000 2250.00000 2250.00000 2250.00000 2250.00000 2250.00000 2250.00000

```

Figure 8-25: BMSEL—an abbreviated beam selection table.

USE BEAMSOL

```

. LIST STRUCTURE
STRUCTURE FOR FILE: BEAMS01.DBF
NUMBER OF RECORDS: 00023
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE

```

NAME	FIELD	NAME	TYPE	WIDTH	DEC
CONNAME1	0001	NAME	C	004	
	0002	CONTYPE1	C	001	
	0003	CONNAME1	C	004	
	0004	POSITN1	N	010	005
	0005	POSITN1	N	010	005
	0006	X1	N	010	005
	0007	CONTYPE2	C	001	
	0008	CONNAME2	C	004	
CONNAME2	0009	POSITN2	N	010	005
	0010	X2	N	010	005
	0011	Y2	N	010	005
	0012	LENGTH	N	010	005
	0013	RLNGTH	N	010	005
	0014	SECTION	C	009	
	** TOTAL **			00104	

LIST

[illegible]

Figure 8-26: BEAMS01—beams on Level 01, final condition.

Chapter 9

Further Work

The design system presented in this thesis illustrates some of the power of the independent database as represented in the relational data model. It was not possible, however, given the limits in time and resources, to implement features that would necessarily be part of a commercial design system, or to fully demonstrate the uses of the building structure database. The concept of the integrated design system built around an independent database has great potential for assisting in all aspects of engineering practice; the present system can show only a small portion of its powers.

9.1 Potential Uses of the Design Database

The introduction of the integrated design system will produce great change in the practice of designing structures, bringing to structural engineering the speed and economy of the machine. But if one looks only at the benefits of computerization in terms of cost efficiency or speedier design one misses perhaps the most important aspect of computerized design—that given the necessarily comprehensive and careful applications programming, the computer can provide greater assurance of safety and the greater efficiency of the structure itself.

The first step in safe design is to ensure that the basic data is consistent, and then to ensure that the derived data is accurate. The computer can perform both these functions.

The building structure database gives central control of all building structure data. The design system maintains data consistency and integrity, all redundant and dependant data are kept correct and up to date. Because the data is accessed universally from the one source, all users work with the same accurate data. This data is also immediately accessible by all users. If the data is required and it exists it is to be found in the database.

With the design system it would also be possible to review the structure design as changes are made to the basic data or as assumed values become known. At any time in the design it should be possible to recheck all design criteria and constraints. Because of the thoroughness of the computer, insufficiencies in the design can be spotted and corrected.

An efficient structure presents economies in the usage of engineering materials, or in economies in construction. As the engineer becomes more confident of his knowledge of structure behaviour, there is less need to allow for uncertainty. The structure data is kept up to date with the latest revisions. The design system may also offer more options to the designer in selecting structural systems. The computer can evaluate several design options and determine the most efficient. A design system should also evaluate hypothetical changes to a structure being designed so that the effects of changes can be studied before they are made.

The computer provides an accuracy and thoroughness of calculation that cannot be matched by hand. Because the computer system can perform calculations quickly and accurately it becomes possible to maintain loading data accurate at all times while allowing for some uncertain future loading. The computer can keep track of all loading, both known and estimated. As changes to the structure or loading are made, the computer can recheck the design of all members affected by the changes. If required the computer can go through the whole design process—something that could not be done manually because of the pressures of time. The computer can monitor changes up to and beyond the "as-built" stage. Future renovations can be checked on the re-activated database.

In the structural engineer, conservatism is a virtue; such are the uncertainties of structural design and the terrible consequences of structural failure. The computer is accurate in its calculations, but these calculations can only approximate the actual behaviour of the real structure. The calculations do not necessarily recognize the uncertainties inherent in the practice of structural design; the assumptions and approximations involved in creating a mathematical idealization of the structure, the uncertainty of loading, of materials, of construction practices. The structural designer respects the fact that at best he can achieve a good approximation of the behaviour of the actual structure, and that any error should be on the side of safety. However, the clear understanding of the structure behaviour is the only firm basis for creating a safe design.

The computer gives the designer more options and the resources for evaluating these options, taking much of the guess work out of structural design. The time element is important. With the computer more time can be spent in preparation and consideration of alternatives. In manual design time is saved by taking short-cuts; making logical assumptions that reduce the amount of manual calculation that must be done. The computer can examine every member for every criteria for every load combination. A detail that might escape the notice of the engineer can be picked out by the computer. Chances of error are reduced. The computer doesn't cover up short-comings in design.

9.2 A Scheme for Data Maintenance

Data maintenance consists of maintaining data consistency and data integrity. Consistency is the agreement of redundant data. Integrity is the agreement of functionally dependant data with basic data [1]. Maintenance of consistency and integrity must be performed in any computer-aided design environment.

In an operating integrated design system the user should have direct access to the data files for reference only. The user should not be permitted to make changes directly to the datafiles as any change imperils the consistency and integrity of the database; rather, there should be in place suitable programmed procedures that will make the changes desired by the designer and in so doing ensure that consistency and integrity are maintained. As the database becomes more complicated the greater is the damage that can be done by improper changes. Access to the database should be made only through a program or package of programs that accepts requests for changes to the database, checks if the change is practicable and then makes the changes to the data. It must also make all other changes required for data maintenance, informing the user what these changes are and how they affect the design.

One means towards managing integrity and consistency is to require the database to be a true representation of an actual structure at all stages of progress. This concept treats the database as if it were a physical model of the structure. Changes to the database would have to make sense in the same way as changes to the actual structure.

For example, the input of data would have to proceed in the same order as building a physical model of the structure. Beams could only be input after the columns to which they

attach are input. Similarly, input of floor slabs would follow input of beams. In each case of input the computer would check to make sure that the specified supporting members exist. If a supporting member were mentioned that the computer could not find, the input would not be accepted. Each new member must be consistent with the existing structure. All new members would be checked for geometry constraints—interference with existing members or designated open spaces. Any other set of constraints for members could be set up against which every new member could also be checked. If consistency is thus ensured at the input stage, the only other concern is data maintenance during and after data changes.

Changes in the database can be accomplished in a manner similar to data input. If the only changes permitted are member addition and member removal data maintenance becomes a manageable problem. Member addition is the same as data input and the same programming can be used, the same checks made. Member removal is member addition in reverse. When a member is removed no elements can be left unsupported, and all references made to a removed member must be corrected.

Regarding the database as a model of the structure, it is obvious that no member can be removed unless it is free; that is, it is not supporting any other member or any load. The removal of a member therefore cannot proceed before the removal of all the elements it supports. For example, moving a beam a small distance horizontally would consist of first removing the beam and then replacing it by a beam in the new location. Removal of the beam cannot proceed until all supported beams are first removed. These beams cannot be removed until the slabs that they support are removed. The slabs cannot be removed until the loads that they support are removed.

The simple procedure of moving a beam even a small distance, following this scheme, entails considerable work: removing all supported elements; removing the beam; placing the new beam; and then replacing all the supported elements. As this procedure can be programmed, the work would not necessarily have to be performed manually. The procedure, however, scrupulously maintains data consistency; the structure described in the database remains physically realizable at all times.

Of course, when changes are made to the basic data, the derived data can no longer be considered valid and must be re-calculated. Whether the whole structure must be re-analysed,

or only a portion of the structure, will depend on the flexibility of the analysis programs and the revision procedures. It is always necessary to know the status of the calculated data; whether or not all revisions have been carried through. It may not be possible to re-calculate dependant data for every revision. Revisions can possibly be allowed to pile up until there are enough to merit new calculations.

Because the whole business of analysis and member selection is programmed, the entire structure can be re-analysed and re-designed as often as desired, making minimum demands on the time of the human designer, providing accurate structure data at any stage of design and maintaining all data accurately up to the as-built stage and, if necessary, beyond.

9.3 Data Security

Another duty of the design system would be to restrict access to the database. Many persons in various disciplines will require access to information stored in the building structure database. Most of these persons should not be permitted to make changes to the database. Full access to the system will be limited to a small number of users. Ideally, access should be limited to the single user who is responsible for the structural design, the structural designer. Access to the database should be permitted only under his authority.

Because so few people should be permitted to make changes to the database, security is not a difficult problem. There is still the usual requirement to have backup copies of data and to keep it secure from deliberate or accidental tampering.

9.4 Further Uses of the Database

Structural design is an important use of the building structure database, but it is only one use.

The tangible product of the structural design office are the structural drawings. The building structure database can be used in computer-aided-drafting. The information that is to be contained in the structural drawing is already stored in the database. All that is required is a translation of data from numerical to graphical form. The organisation of data in the relational database makes for an easy translation. Drafting then is the logical next

process to be programmed after design, in fact the database might well be created for the purpose of drafting with design an afterthought.

Material-take-off becomes a trivial exercise with the building structure database. Using the DBMS directly without any Data Manipulation Language programs the quantities of any type of member can be itemized in any desired way. Estimation is essentially the joining of unit prices to material-take-off data. This is achieved with ease using the powers of the relational database.

Given a supported and a supporting member, knowing their section properties and the load transfer, a connection can be designed, and a description of the connection can be maintained in the database. The location of bolt holes or cut-aways on members can be similarly recorded. From this information fabrication shop drawings can be made, or, as this information is in computer-accessible form in the database, it can be used directly in the computer-aided fabrication of structural members.

This is really just the start of computer applications in building design and construction. Ideally, the system will cross the boundaries of different disciplines. Communication between systems operating in Architectural, Mechanical, Electrical, Civil and other departments will be loosely joined to create a single design and information system. There will also be communication between these disciplines and Project Management and Construction.

What this involves is simply the electronic transfer of data, which is no new thing; the innovation is to store and work with the data in electronically transferable form. The data itself is the same whether written with a pencil on paper or digitally encoded in a computer memory; anything that can be done with the one can be done equally well with the other. The difference is of course that the computer can operate only on specially encoded data and the storage of this encoded data allows better use of the computer. Without permanent storage of data in a computer database the full potential of the computer can not be realised.

9.5 Expanding the Present System

The present system, although it demonstrates the uses of the building structure database and provides a useful design function, is very limited in its powers. Owing to the flexibility

and adaptability of the central relational database the present system could be expanded and so become a much more useful design tool.

Addition to the present system would require more powerful computer resources. The present system approaches the limits of the resources used. Because the data representation is in the form of relations and not specific to the relational DBMS used, the same relations that have been implemented in the present system could be used in another relational DBMS.

Without making much change to the database the design system could be expanded to design all the member types permitted in the database. The inclusion of horizontal load analysis would permit the design of columns and diagonal bracing and wall cladding. A more sophisticated treatment of floor slabs would allow their design. Introduction of new member types, such as continuous beams, posts, hangers and stairways would make the system very practical.

This expansion could be made on the foundation of the present system, which deals with a limited class of structures. A more general system, in which fewer assumptions would be made about the form of the building structure, would need a more detailed description of the structural elements and their positions. If moment connections were to be permitted in the database description, this would require added description data, but would permit analysis of rigid frame structures. Further expansion to include reinforced concrete or composite members would introduce further complications in the structure description. Obviously, as the system handles a greater variety of elements, the description must be more complicated; fewer convenient assumptions can be made. The ideal structural design system would be completely general, accepting any type of structure, so the database must also be completely general.

Getting back to the present system, it is possible to use the existing database to create a drawing of the structure. This would be useful not only for drafting but also for interactive graphics. Other tasks such as material take off are quite simple to accomplish.

The major work that needs to be done is the creation of the necessary means for handling the control and evaluation aspects of design. Analysis of structural behaviour is the easy part of the design system to implement. The difficult part is the programs for the creation

of the structural model for analysis and the evaluation of design criteria. The design of a structure involves many decisions; these have to be made by the design system.

In the present system all design decisions are left in the hands of the designer. The design system only performs some member selection evaluations. All evaluations involving geometry constraints are done manually. A well developed system must be able to perform these evaluations automatically.

The database itself is only a description of the structure. The interpretation of this description in structural engineering terms is performed by applications programs outside the database. There are no limits to the uses that are to be made of the computer database, as there are no limits to the use the information in any other form. The database is simply information in computer-accessible form. Many programs already exist that can make use of this information. The existing programs are, however, very much limited in that data must be manually input. Use of the database eliminates the need for manual input and allows a much wider and more complete use of the computer in structural engineering.

Chapter 10

Conclusion

It is clear that the future will bring much more extensive use of the computer to meet the demands of efficiency and economy in a competitive market. Effective use of the computer in structural engineering will require the development of an integrated design system—consisting of an independent central database and a body of many individual applications programs—that will assist in all aspects of structural design office practice. The use of a central database permits the applications to be independent of the organization of data in the database. Changes can be made to the database organization or the applications programs without affecting the design system as a whole; the system therefore can be easily maintained and improved.

The database for structural design—the building structure database—will be expected to store a great variety of data. The most important data to be stored is the description of the building structure itself. The building description that would be most widely useful would be a general building description, describing the actual physical structure of the building without regard for any specific application. How this data is to be used is the proper concern of the applications program and should not dictate the organization of the database. Interpretation of the physical description in terms useful to a specific application would be the role of other applications programs. The building structure database should be a complete building description requiring no additional or alternate data sources.

A suitable organization for the storage of building structure data is provided by the relational data model. The ability of this data model to logically and concisely express data and their relationships is combined with an easy and powerful means of data manipulation, based on the mathematics of relations. The relational model provides adequate flexibility and adaptability for engineering use.

Any manner of describing the structure completely and accurately would make possible the development of an integrated design system. The manner of representation that has been presented records the locations of structural components relative to their supports, implicitly recognizing the network of component relationships. This method avoids the redundancy that would be introduced by storing the absolute geometry of components, but does not sacrifice any accuracy of description. This representation is logical and concise, and while only the description of a simple class of structure has been attempted, the extension of the representation to include a wider range of structures would involve the same basic principles. The description, as it stands, is accurate within a limited vocabulary. The nature of the relational database allows for additions to the present vocabulary and inclusion of greater detail of description. The database can be expanded to permit an unlimited field of applications.

The database is not linked to any specific applications. The real power of the database is that the building structure data is available in computer accessible form for any application that can be devised. While the present system is demonstrated only for structural design, the same basic data can be used as-is for other applications such as, for example, material take off or member detailing.

The role of the building structure database is to store building structure data in an organized and accessible manner. While certain functions such as data security are necessarily part of the database system, other database related functions such as data integrity maintenance can be the role of a database specific applications program. The database must store data that is dependant on other data; the correspondence of these data must be maintained. Such maintenance requires the knowledge of the functional relationships among the data, and such knowledge is not an essential part of the database itself which stores physical relationships. Basically, the database can be viewed as a filing cabinet; data management must be performed by a special data manager that can operate outside the database management system.

A major difficulty in using a general building description, is that some applications can only deal with a mathematical idealization of the structure. Such an idealization would not be stored in the database but must somehow be derived from the general description. It is not inconsistent with the principles of the integrated design system for this to be performed

manually or by a program relying on interactive assistance from the design engineer. Creation of programs that would model a structure for applications programs is an important area to be developed if databases are to represent actual buildings rather than structural models. This points to the development of an expert system for structural design.

Building structure data can be stored in a computer database with great benefit to the efficiency of the computer. The elimination of manual input of data permits the computer to perform tasks that would otherwise result in no time savings because of time spent on data entry. The computer can assist in a great many tasks if information is kept in computer accessible form. The more general the database—that is, the more data that is stored—the greater are the potential uses. Data should be stored in the manner that is most useful; with greater reliance on computer aids, the most useful manner of storing data is in a computer database.

Appendix A

Data File Structures & Program Listings

```

USE AXES
. LIST STRUCTURE
STRUCTURE FOR FILE: AXES.DBF
NUMBER OF RECORDS: 00001
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      002
002      ORTHO     C      002
003      ROTATION  N      010    005
** TOTAL **                00015

```

```

.
.
.
.
.
USE COLINES
. LIST STRUCTURE
STRUCTURE FOR FILE: COLINES.DBF
NUMBER OF RECORDS: 00005
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      004
002      DIST      N      010    005
003      ORIEN     C      002
** TOTAL **                00017

```

```

.
.
.
.
.
USE LEVELS
. LIST STRUCTURE
STRUCTURE FOR FILE: LEVELS.DBF
NUMBER OF RECORDS: 00003
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      002
002      TOS:EL    N      010    003
** TOTAL **                00013

```

```

USE COLUMNS
. LIST STRUCTURE
STRUCTURE FOR FILE: COLUMNS.DBF
NUMBER OF RECORDS: 00006
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      004
002      LINE1     C      004
003      LINE2     C      004
004      ORIEN     C      002
** TOTAL **          00015

```

```

. USE BEAMS01
. LIST STRUCTURE
STRUCTURE FOR FILE: BEAMS01.DBF
NUMBER OF RECORDS: 00023
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      004
002      CONTYPE1   C      001
003      CONNAME1   C      004
004      POSITN1    N      010    005
005      X1         N      010    005
006      Y1         N      010    005
007      CONTYPE2   C      001
008      CONNAME2   C      004
009      POSITN2    N      010    005
010      X2         N      010    005
011      Y2         N      010    005
012      LENGTH     N      010    005
013      RLENGTH    N      010    005
014      SECTION    C      009
** TOTAL **          00104

```

```

. USE PANELS01
. LIST STRUCTURE
STRUCTURE FOR FILE: PANELS01.DBF
NUMBER OF RECORDS: 00016
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      004
002      JOIST1     C      004
003      POSITN1A   N      010   005
004      POSITN2A   N      010   005
005      JOIST2     C      004
006      POSITN1B   N      010   005
007      POSITN2B   N      010   005
008      WIDTH      N      010   005
** TOTAL **                00063

```

```

. USE CONCL01
. LIST STRUCTURE
STRUCTURE FOR FILE: CONCL01.DBF
NUMBER OF RECORDS: 00008
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      004
002      SOURCET    C      001
003      SOURCEN    C      004
004      LOADCASE   C      005
005      LOAD       N      010   005
006      POSITN1    N      010   005
** TOTAL **                00035

```

```

. USE PLX01
. LIST STRUCTURE
STRUCTURE FOR FILE: PLX01.DBF
NUMBER OF RECORDS: 00032
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      004
002      LOADCASE   C      005
003      LOAD       N      010   005
004      POSITN1    N      010   005
005      POSITN2    N      010   005
** TOTAL **                00040

```


. USE BLX01

LIST STRUCTURE

STRUCTURE FOR FILE: BLX01.DBF

NUMBER OF RECORDS: 00146

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	004	
002	SOURCET	C	001	
003	SOURCEN	C	004	
004	LOADCASE	C	005	
005	LOADTYPE	C	001	
006	LOAD	N	010	005
007	POSITN1	N	010	005
008	POSITN2	N	010	005
** TOTAL **			00046	

. USE BMDIAG01

LIST STRUCTURE

STRUCTURE FOR FILE: BMDIAG01.DBF

NUMBER OF RECORDS: 00092

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	004	
002	LOADCOMB	C	005	
003	POSITN	N	010	005
004	SHEAR1	N	010	005
005	SHEAR2	N	010	005
006	MOMENT1	N	010	005
** TOTAL **			00050	

. USE MOMENT01

LIST STRUCTURE

STRUCTURE FOR FILE: MOMENT01.DBF

NUMBER OF RECORDS: 00023

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	004	
002	LOADCOMB	C	005	
003	SHEAR	N	010	005
004	MOMENTP	N	010	005
005	MOMENTN	N	010	005
** TOTAL **			00040	

```

. USE BLX
. LIST STRUCTURE
STRUCTURE FOR FILE: BLX.DBF
NUMBER OF RECORDS: 00000
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      NAME      C      004
002      SOURCET    C      001
003      SOURCEN    C      004
004      LOADCASE   C      005
005      LOADTYPE   C      001
006      LOAD       N      010    005
007      POSITN1    N      010    005
008      POSITN2    N      010    005
** TOTAL **              00046

```

```

. USE POSITNS
. LIST STRUCTURE
STRUCTURE FOR FILE: POSITNS.DBF
NUMBER OF RECORDS: 00003
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      POSITN     N      010    005
002      SHEAR1     N      010    005
003      SHEAR2     N      010    005
004      MOMENT1    N      010    005
** TOTAL **              00041

```

```

. USE RESTRS
. LIST STRUCTURE
STRUCTURE FOR FILE: RESTRS.DBF
NUMBER OF RECORDS: 00002
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE WIDTH  DEC
001      RTYPE      C      001
002      POSITN1    N      010    005
003      POSITN2    N      010    005
** TOTAL **              00022

```

```

* XANDY *
*
* This program calculates beam end co-ordinates and beam lengths
* and stores them in the data file BEAMS&L.
*
* Sub-programs: BEAMLIST; COLCOORD; PUTCORD1; PUTCORD2; BEAMEQ; LENGTH.
*
*
* 'Welcome to XANDY'
*
ACCEPT 'LEVEL' TO L
*
? 'BEAMLIST required? (Y/N)'
ACCEPT TO YESNO
*
IF YESNO < 'N'
    RELEASE CONT
    DO BEAMLIST
ENDIF
*
SELECT PRIMARY
USE BEAMLIST
*
COUNT TO END
GOTO BOTTOM
STORE 0 TO HERE
*
DO WHILE HERE<END
    STORE HERE+1 TO HERE
*
    IF NAME='XXXX'
        SKIP -1
        LOOP
    ENDIF
*
    STORE # TO NBMLIST
    STORE NAME TO BMNAME
*
    USE BEAMS&L
    LOCATE FOR NAME=BMNAME
*
    IF CONTYPE1='C'
        STORE CONNAME1 TO COLNAME
        DO COLCOORD
        DO PUTCORD1
    ENDIF
*
    IF CONTYPE2='C'
        STORE CONNAME2 TO COLNAME
        DO COLCOORD
        DO PUTCORD2
    ENDIF
*

```

```
IF CONTYPE1='B'
  STORE CONNAME1 TO CBEAM
  STORE LENGTH TO LEN
  STORE POSITN1 TO POS
  DO BEAMEQ
  DO PUTCORD1
ENDIF
*
IF CONTYPE2='B'
  STORE CONNAME2 TO CBEAM
  STORE LENGTH TO LEN
  STORE POSITN2 TO POS
  DO BEAMEQ
  DO PUTCORD2
ENDIF
*
DO LENGTH
*
USE BEAMLIST
GOTO NBMLIST
SKIP -1
*
ENDDO
*
RELEASE NBMLIST,BMNAME,COLNAME,CBEAM,LEN, POS,X,Y
*
RETURN
```

```

      * BEAMLIST *
*
* This is a sub-program for programs XANDY and AUTOLoad.
*
* This program creates an ordered list of beam names in levels of
*   independance, the data file BEAMLIST.
*
* Sub-programs: BMLIST2; BEAMDEP; NODUPS.
*
DO BMLIST2
STORE T TO CONT
*
DO WHILE CONT
  DO BEAMDEP
  DO NODUPS
ENDDO
*
RELEASE CONT
*
RETURN

```

```

      * BMLIST2 *
*
* This is a sub-program of BEAMLIST.
*
* This program initializes the data file BEAMLIST.
*
USE BEAMLIST
DELETE ALL
PACK
APPEND FROM BEAMS&L
APPEND BLANK
REPLACE NAME WITH 'XXXX'
USE
RETURN

```

```

      * BEAMDEP *
*
* This is a sub-program of BEAMLIST.
*
* This program determines beam dependance and moves dependant beams
*   below the last 'XXXX' record in the file BEAMLIST and appends
*   another 'XXXX' record.
*
SELECT PRIMARY
USE BEAMLIST
*
GOTO BOTTOM
SKIP -1

```

```

*
IF NAME='XXXX'
  STORE F TO CONT
  RETURN
ENDIF
*
GOTO TOP
COUNT FOR NAME='XXXX' TO XES
*
IF XES=1
  GOTO TOP
ENDIF
*
IF XES>1
  LOCATE FOR NAME='XXXX'
  DO WHILE XES>2
    STORE XES-1 TO XES
    CONTINUE
  ENDDO
  SKIP
ENDIF
*
SELECT SECONDARY
USE BEAMS&L
SELECT PRIMARY
*
DO WHILE NAME <> 'XXXX'
*
  SELECT SECONDARY
  LOCATE FOR S.NAME = P.NAME
*
  IF CONTYPE1='B'
    SELECT PRIMARY
    STORE # TO PLACE
    GOTO BOTTOM
    APPEND BLANK
    REPLACE P.NAME WITH CONNAME1
    GOTO PLACE
    SELECT SECONDARY
  ENDIF
*
  IF CONTYPE2='B'
    SELECT PRIMARY
    STORE # TO PLACE
    GOTO BOTTOM
    APPEND BLANK
    REPLACE P.NAME WITH CONNAME2
    GOTO PLACE
    SELECT SECONDARY
  ENDIF
*
  SELECT PRIMARY
  SKIP

```

```

*
ENDDO
*
GOTO BOTTOM
APPEND BLANK
REPLACE P.NAME WITH 'XXXX'
*
RELEASE PLACE,XES
RETURN

```

```

* NODUPS *
*
* This is a sub-program of BEAMLIST.
*
* This program takes the data file BEAMLIST and removes duplicate beam names.
*
SELECT PRIMARY
USE BEAMLIST
GOTO TOP
COUNT TO END
*
GOTO TOP
COUNT FOR NAME = 'XXXX' TO XES
*
GOTO TOP
LOCATE FOR NAME = 'XXXX'
  DO WHILE XES > 2
    STORE XES-1 TO XES
    CONTINUE
  ENDDO
  SKIP
*
DO WHILE # < END
  STORE # TO NRECORD
  STORE NAME TO SNAME
  DELETE ALL FOR NAME = SNAME
  GOTO NRECORD
  INSERT BLANK
  REPLACE NAME WITH SNAME
  STORE END+1 TO END
  SKIP
ENDDO
*
PACK
RELEASE SNAME, NRECORD,XES,END
RETURN

```



```

* COLCOORD *
*
* This is a sub-program of XANDY.
*
* This program determines the co-ordinates of a column and stores
*   them in the variables X and Y.
*
SELECT SECONDARY
USE COLUMNS
*
LOCATE FOR NAME=COLNAME
STORE LINE1 TO MLINE1
STORE LINE2 TO MLINE2
*
USE COLINES
*
LOCATE FOR NAME=MLINE1
IF ORIEN='NS'
  STORE DIST TO X
ELSE
  STORE DIST TO Y
ENDIF
LOCATE FOR NAME=MLINE2
IF ORIEN='NS'
  STORE DIST TO X
ELSE
  STORE DIST TO Y
ENDIF
RELEASE MLINE1,MLINE2
RETURN

* PUTCORD1 *
*
* This is a sub-program of XANDY.
*
* This program inserts end-1 co-ordinate values in the data file BEAMS&L.
*
USE BEAMS&L
LOCATE FOR NAME=BMNAME
REPLACE X1 WITH X
REPLACE Y1 WITH Y
RETURN

```

```

* PUTCORD2 *
*
* This is a sub-program of XANDY.
*
* This program inserts end-2 co-ordinate values in the data file BEAMS&L.

```

```

*
USE BEAMS&L
LOCATE FOR NAME=BMNAME
REPLACE X2 WITH X
REPLACE Y2 WITH Y
RETURN

```

```

* BEAMEQ *
*
* This is a sub-program of XANDY.
*
* This program determines the co-ordinates of any point along a beam
*   given the distance POS from end-1.
*
USE BEAMS&L
LOCATE FOR NAME=CBEAM
*
IF X1=X2
  STORE X1 TO X
  STORE Y1+POS TO Y
*
ELSE
*
  IF Y1=Y2
    STORE Y1 TO Y
    STORE X1+POS TO X
  *
  ELSE
  *
    STORE X2-X1 TO RUN
    STORE Y2-Y1 TO RISE
    STORE POS/LEN TO RATIO
    STORE X1+RUN*RATIO TO X
    STORE Y1+RISE*RATIO TO Y
  *
  ENDIF
*
ENDIF
*
RELEASE RUN,RISE,RATIO
*
RETURN

```

```

* LENGTH *
*
* This is a sub-program of XANDY.
*
* This program calculates beam length and files it in the
*   beam file BEAMS&L.
*
* Sub-programs:  SQRX.
*
USE BEAMS&L
LOCATE FOR NAME=BMNAME
*
STORE X2-X1 TO RUN
STORE Y2-Y1 TO RISE
STORE (RUN*RUN)+(RISE*RISE) TO X
*
DO SQRX
*
REPLACE LENGTH WITH SQRX
*
RELEASE RUN,RISE,X,SQRX
*
RETURN

* SQRX *
*
* This command file accepts a number X and gives its square root
*   to five decimals in the variable SQRX.
*
STORE T TO SQRCONT
STORE 0 TO NSQR
STORE 1.000000 TO XN1
*
DO WHILE SQRCONT
*
  STORE NSQR+1 TO NSQR
*
  STORE 0.500000*(XN1+(X/XN1)) TO XN2
*
  STORE (XN1-XN2) TO DX
  IF DX<0
    STORE 0.00000-DX TO DX
  ENDIF
*
  IF DX<0.00001 .OR. NSQR>20
    STORE F TO SQRCONT
  ENDIF
*
STORE XN2 TO XN1
ENDDO

```

*
STORE XN2 TO SQRTX
*
RELEASE SQRCNT, XN1, XN2, NSQR, DX
RETURN

```

* AUTOLOAD *
*
* This program takes applied loading from the and transfers
* the load through the floor system to the beams and
* columns, for floor level 'L', and loadcase 'LCASE'.
*
* Sub-programs: CONCLOAD; PANEL; BEAMEVAL.
*
SET TALK OFF
SAVE TO MEMFILE
CLEAR
*
ACCEPT 'Level (2 Chars)' TO L
*
ACCEPT 'DO BEAMLIST? (Y/N)' TO GO
*
IF GO = 'N'
    DO BEAMLIST
ENDIF
*
RELEASE GO
*
STORE 'Y' TO NEXT
DO WHILE NEXT='Y'
*
RELEASE NEXT
*
ACCEPT 'Loadcase (5 Chars)' TO LCASE
*
USE BLX&L
DELETE FOR LOADCASE=LCASE
PACK
*
USE CLX
DELETE FOR LOADCASE=LCASE .AND. SOURCE=L
PACK
*
? 'CONCLOAD in progress'
*
DO CONCLOAD
*
*
? 'PANEL in progress'
*
DO PANEL
*
*
? 'BEAMEVAL in progress'
*
DO BEAMEVAL
*
*
ACCEPT 'DO YOU WANT ANOTHER LOADCASE? (Y/N)' TO NEXT
ENDDO

```

```

*
CLEAN
RESTORE FROM MEMFILE
*
SET TALK ON
RETURN

```

```

* COLCOORD *
*
* This is a sub-program of XANDY.
*
* This program determines the co-ordinates of a column and stores
* them in the variables X and Y.
*
SELECT SECONDARY
USE COLUMNS
*
LOCATE FOR NAME=COLNAME
STORE LINE1 TO MLINE1
STORE LINE2 TO MLINE2
*
USE COLINES
*
LOCATE FOR NAME=MLINE1
IF ORIEN='NS'
    STORE DIST TO X
ELSE
    STORE DIST TO Y
ENDIF
LOCATE FOR NAME=MLINE2
IF ORIEN='NS'
    STORE DIST TO X
ELSE
    STORE DIST TO Y
ENDIF
RELEASE MLINE1,MLINE2
RETURN

```

```

* PANEL *
*
* This is a sub-program of AUTOLOAD.
*
* This command file distributes loads on panels to the supporting Joists.
*
* Sub-programs: PANEL2.
*
SELECT PRIMARY

```

```
USE PANELS&L
GOTO TOP
*
DO WHILE .NOT. EOF
  STORE NAME TO XP
  STORE WIDTH TO MWIDTH
*
  SELECT SECONDARY
  USE PLX&L
  COPY TO PLXTEMP FOR NAME=XP
  USE PLXTEMP
*
  COUNT FOR LOADCASE=LCASE TO N
*
  IF N=0
    SELECT PRIMARY
    SKIP
    LOOP
  ENDIF
*
  SELECT PRIMARY
  STORE JOIST1 TO XB
  STORE POSITN1A TO POS1
  STORE POSITN2A TO POS2
*
  DO PANEL2
*
  SELECT PRIMARY
  USE PANELS&L
  LOCATE FOR NAME=XP
  STORE JOIST2 TO XB
  STORE POSITN1B TO POS1
  STORE POSITN2B TO POS2
*
  DO PANEL2
*
  SELECT PRIMARY
  USE PANELS&L
  LOCATE FOR NAME=XP
  SKIP
ENDDO
*
DO CLOSE
*
DELETE FILE PLXTEMP
RELEASE N,XB,XP,POS1,POS2,MWIDTH
RETURN
.
```

```

* PANEL2 *
*
* This is a sub-program of PANEL.
*
* This program inserts load data into the data file BLX&L.
*
SELECT SECONDARY
USE PLXTEMP
*
GOTO TOP
LOCATE FOR LOADCASE=LCASE
STORE N TO N2
*
DO WHILE N2>0
  STORE N2-1 TO N2
  SELECT PRIMARY
  USE BLX&L
*
  GOTO BOTTOM
  APPEND BLANK
  REPLACE NAME WITH XB
  REPLACE SOURCET WITH 'P'
  REPLACE SOURCEN WITH XP
  REPLACE LOADTYPE WITH 'U'
  REPLACE P.LOAD WITH (S.LOAD * MWIDTH)/2.0
  REPLACE P.POSITN1 WITH (S.POSITN1 + POS1)
  REPLACE P.POSITN2 WITH (S.POSITN2 + POS1)
  REPLACE P.LOADCASE WITH LCASE
*
  SELECT SECONDARY
  IF N2>0
    CONTINUE
  ENDIF
ENDDO
*
RELEASE N2
*
RETURN

```



```

* BEAMEVAL *
* This is a sub-program of AUTOLOAD.
*
* This program calculates beam end reactions and stores them as
* loads in the data file BLX&L.
*
* Sub-programs: PUTLOADC; PUTLOAD.
*
SELECT PRIMARY
USE BEAMLIST
GOTO TOP
*
DO WHILE .NOT. EOF
*
  IF NAME = 'XXXX'
    SKIP
    LOOP
  ENDIF
*
  SELECT SECONDARY
  USE BEAMS&L
  LOCATE FOR S.NAME=P.NAME
*
  STORE NAME TO XB
  STORE LENGTH TO LEN
  SELECT SECONDARY
  USE BLX&L
  COPY TO BLXTEMP FOR NAME=XB
  USE BLXTEMP
*
  COUNT FOR LOADCASE=LCASE TO NC
*
  IF NC=0
    SELECT PRIMARY
    SKIP
    LOOP
  ENDIF
*
  GOTO TOP
  LOCATE FOR LOADCASE=LCASE
*
  STORE 0.000 TO R1
  STORE 0.000 TO R2
*
  DO WHILE NC > 0
*
    IF LOADTYPE='U'
      STORE (POSITN1 + POSITN2)/2.00000 TO EQDIST
      STORE (POSITN2 - POSITN1)* LOAD TO EQLOAD
      STORE R1 + (1.0000-EQDIST/LEN)*EQLOAD TO R1
      STORE R2 + (EQDIST/LEN)*EQLOAD TO R2
    ENDIF
*

```

```

      IF LOADTYPE='C'
        STORE R1 + (1.0000-POSITN1/LEN)*LOAD TO R1
        STORE R2 + (POSITN1/LEN)*LOAD TO R2
      ENDIF
*
      STORE NC-1 TO NC
      IF NC <> 0
        CONTINUE
      ENDIF
*
      ENDDO
*
      USE BEAMS&L
      LOCATE FOR NAME=XB
*
      IF CONTYPE1='C'
        STORE R1 TO REAC
        STORE # TO NBEAM
        STORE CONNAME1 TO XC
*
        DO PUTLOADC
*
      ENDIF
*
      IF CONTYPE2='C'
        STORE R2 TO REAC
        STORE # TO NBEAM
        STORE CONNAME2 TO XC
*
        DO PUTLOADC
*
      ENDIF
*
      IF CONTYPE1='B'
        STORE # TO NBEAM
        STORE CONNAME1 TO CON
        STORE POSITN1 TO POS
        STORE R1 TO REAC
*
        DO PUTLOAD
*
        USE BEAMS&L
        GOTO NBEAM
      ENDIF
*
      IF CONTYPE2='B'
        STORE # TO NBEAM
        STORE CONNAME2 TO CON
        STORE POSITN2 TO POS
        STORE R2 TO REAC
*
        DO PUTLOAD
*

```

```
USE BEAMS&L  
GOTO NBEAM  
ENDIF  
*  
SELECT PRIMARY  
SKIP  
ENDDO  
*  
DELETE FILE BLXTEMP  
*  
RELEASE XB,R1,R2,CON,POS,NBEAM,REAC  
*  
RETURN
```

```

* PUTLOADC *
*
* This is a sub-program of BEAMEVAL.
*
* This program inserts loading data into file CLX.
*
*
USE CLX
GOTO BOTTOM
APPEND BLANK
*
REPLACE NAME WITH XC
REPLACE SOURCET WITH 'B'
REPLACE SOURCEN WITH XB
REPLACE SOURCEC WITH L
REPLACE LOADTYPE WITH 'C'
REPLACE LOADCASE WITH LCASE
REPLACE LOAD WITH REAC
*
USE BEAMS&L
GOTO NBEAM
*
RETURN

```

```

* PUTLOAD *
*
* This is a sub-program of BEAMEVAL.
*
* This program inserts loading data into file BLX&L.
*
*
LOCATE FOR NAME=CON
*
USE BLX&L
*
APPEND BLANK
REPLACE NAME WITH CON
REPLACE SOURCET WITH 'B'
REPLACE SOURCEN WITH XB
REPLACE LOADTYPE WITH 'C'
REPLACE LOADCASE WITH LCASE
REPLACE LOAD WITH REAC
REPLACE POSITN1 WITH POS
*
RETURN

```

```

* MOMENTS *
*
* This program calculates bending moment and shear diagrams for the
*   data file BMDIAG&L and also separates the maximum values into
*   the data file MOMENT&L.
*
* Sub-programs: FACTLOAD; POSITNS; NODUPS2; SHANDBM.
*
SET TALK OFF
CLEAR
*
*
ACCEPT 'LEVEL' TO L
ACCEPT 'LOAD COMBINATION' TO LCOMB
*
DO CLOSE
*
USE BMDIAG&L
DELETE ALL FOR LOADCOMB=LCOMB
PACK
*
USE MOMENT&L
DELETE ALL FOR LOADCOMB=LCOMB
PACK
*
DO FACTLOAD
*
SELECT PRIMARY
USE BEAMS&L
*
GOTO TOP
DO WHILE .NOT. EOF
*
    STORE NAME TO BMNAME
    STORE # TO NBEAMS
    STORE LENGTH TO LEN
*
    USE BMLOAD
    COPY TO BEAMLOAD FOR NAME=BMNAME
*
    DO POSITNS
*
    DO NODUPS2
*
    DO SHANDBM
*
    ?
    ?
    ?
    ? 'BEAM', BMNAME
    ?
    USE POSITNS
    SORT ON POSITN TO TEMP

```

```

USE TEMP
LIST
*
USE BMDIAG&L
COPY STRUCTURE TO TEMP2
USE TEMP2
APPEND FROM TEMP
REPLACE ALL NAME WITH BMNAME
REPLACE ALL LOADCOMB WITH LCOMB
USE BMDIAG&L
APPEND FROM TEMP2
DELETE FILE TEMP
DELETE FILE TEMP2
*
SELECT PRIMARY
USE BEAMS&L
GOTO NBEAMS
SKIP
*
ENDDO
*
?
?
? 'MAXIMUM SHEARS AND MOMENTS FOR BEAMS ON LEVEL',L,'.'
?
USE MOMENT&L
REPORT FORM MOMENTS
*
SET TALK ON
RETURN

```

```

* FACTLOAD *
*
* This is a sub-program of MOMENTS.
*
* This program joins load factors from datafiles CASES to the loading
*   in file BLX&L, to create a factored loading file BMLOAD.
*
*
DO CLOSE
*
SELECT SECONDARY
USE CASES
COPY TO TCASES FOR LOADCOMB=LCOMB
USE TCASES
*
SELECT PRIMARY
USE BLX&L
*
JOIN TO BMLOAD FOR S.LOADCASE=P.LOADCASE FIELD NAME, LOADTYPE,LOAD,;

```

```

    POSITN1,POSITN2,FACTOR
*
USE BMLOAD
REPLACE ALL LOAD WITH LOAD*FACTOR
*
DO CLOSE
*
RETURN

```

```

* POSITNS *
*
* This is a sub-program of MOMENTS.
*
* This program determines the positions at which beam shear force and bending
*   moment are to be calculated for each beam. The critical positions are
* 1) The beam ends;
* 2) Points of concentrated loading;
* 3) Wherever a distributed load begins or ends.
*
*
DO CLOSE
*
SELECT PRIMARY
USE BEAMLOAD
*
SELECT SECONDARY
USE POSITNS
DELETE ALL
PACK
*
SELECT PRIMARY
GOTO TOP
*
DO WHILE .NOT. EOF
*
    SELECT SECONDARY
    APPEND BLANK
    REPLACE POSITN WITH POSITN1
    APPEND BLANK
    REPLACE POSITN WITH POSITN2
*
    SELECT PRIMARY
    SKIP
ENDDO
*
SELECT SECONDARY
APPEND BLANK
REPLACE POSITN WITH LEN
APPEND BLANK

```

REPLACE POSITN WITH 0.00000

*
RETURN

* NODUPS2 *

*

* This is a sub-program of MOMENTS.

*

* This program eliminates duplicate positions in the file POSITNS.

*

*

DO CLOSE

*

SELECT PRIMARY

USE POSITNS

*

GOTO BOTTOM

STORE # TO END

GOTO TOP

*

DO WHILE #<END

*

STORE POSITN TO KEEP

STORE # TO NPOSITNS

DELETE ALL FOR POSITN=KEEP

APPEND BLANK

REPLACE POSITN WITH KEEP

GOTO NPOSITNS

SKIP

*

ENDDO

*

PACK

SORT ON POSITN TO POSITNS2

USE POSITNS2

COPY TO POSITNS

USE

DELETE FILE POSITNS2

*

DO CLOSE

*

RETURN


```

* CHANDEM *
*
* This is a sub program of MOMENTS.
*
* This program calculates beam bending moment and shear force at
* points along a beam given by data file POSITNS.
*
* Sub-programs: ZCROSS; MANIMS.
*
*
DO CLOSE
*
*
SELECT PRIMARY
USE BEAMLOAD
*
SELECT SECONDARY
USE POSITNS
*
SELECT PRIMARY
  STORE 0 TO REAC
*
DO WHILE .NOT. EOF
*
  IF LOADTYPE='U'
    STORE (POSITN1+POSITN2)/2.00000 TO EQDIST
    STORE (POSITN2-POSITN1)*LOAD TO EQLOAD
    STORE REAC+(1.00000-EQDIST/LEN)*EQLOAD TO REAC
  ENDIF
*
  IF LOADTYPE='C'
    STORE REAC+(1.00000-POSITN1/LEN)*LOAD TO REAC
  ENDIF
*
  SKIP
ENDDO
*
*
STORE 0 TO OLDPOS
STORE 0 TO OLDSHEAR
*
SELECT SECONDARY
GOTO TOP
*
DO WHILE .NOT. EOF
  STORE POSITN TO P
*
  SELECT PRIMARY
  COUNT FOR POSITN1<=P TO NPOS
*
  IF NPOS=0
    SELECT SECONDARY

```

```

        REPLACE SHEAR1 WITH REAC
        REPLACE SHEAR2 WITH REAC
        REPLACE MOMENT1 WITH REAC*P
    SKIP
    LOOP
ENDIF
*
GOTO TOP
*
STORE 0 TO N
STORE 0 TO ISHEAR
STORE 0 TO IMOMENT
STORE 0 TO MSHEAR
*
LOCATE FOR POSITN1 =P
*
DO WHILE N<NPOS
    STORE N+1 TO N
    *
    IF LOADTYPE='C'
        IF POSITN1<P
            STORE LOAD+ISHEAR TO ISHEAR
            STORE LOAD*(P-POSITN1)+IMOMENT TO IMOMENT
        ELSE
            STORE LOAD+MSHEAR TO MSHEAR
        ENDIF
    ENDIF
    *
    IF LOADTYPE='U'
        IF POSITN2<=P
            STORE LOAD*(POSITN2-POSITN1)+ISHEAR TO ISHEAR
            STORE (POSITN1+POSITN2)/2.00000 TO POSITNAV
            STORE LOAD*(POSITN2-POSITN1)*(P-POSITNAV)+IMOMENT TO IMOMENT
        ELSE
            STORE LOAD*(P-POSITN1)+ISHEAR TO ISHEAR
            STORE (LOAD*(P-POSITN1)*(P-POSITN1)/2)+IMOMENT TO IMOMENT
        ENDIF
    ENDIF
    *
    IF N<NPOS
        CONTINUE
    ENDIF
    *
    *
ENDDO
*
SELECT SECONDARY
REPLACE SHEAR1 WITH REAC-ISHEAR
REPLACE SHEAR2 WITH REAC-ISHEAR-MSHEAR
REPLACE MOMENT1 WITH REAC*P-IMOMENT
*
DO ZCROSS

```

```

*
* SHIP
*
ENDDO
*
*
DO MANDMS
*
DO CLOSE
*
RETURN

```

```

* ZCROSS *
*
* This is a sub-program of SHANDBM.
*
* This program determines locations of zero shear along a beam.
*
STORE POSITN TO NEWPOS
STORE SHEAR1 TO NEWSHEAR
*
IF NEWSHEAR*OLDSHEAR<0 .AND. NEWPOS OLDPOS
*
  SELECT PRIMARY
  *
  SUM LOAD TO ZLOAD FOR POSITN1<=OLDPOS .AND. POSITN2>=NEWPOS .AND.;
  LOADTYPE='U'
  STORE OLDSHEAR/ZLOAD TO DELTAPOS
  *
  IF DELTAPOS < 0
    STORE DELTAPOS*(-1.00000) TO DELTAPOS
  ENDIF
  *
  STORE OLDPOS+DELTAPOS TO ZPOS
  *
  SELECT SECONDARY
  STORE # TO NPOSITNS
  APPEND BLANK
  REPLACE POSITN WITH ZPOS
  GOTO NPOSITNS
  *
ENDIF
*
*
*
*
SELECT SECONDARY
*
STORE SHEAR2 TO OLDSHEAR
STORE POSITN TO OLDPOS
*
RETURN

```

```

* SHANDMS *
*
* This is a sub-program of SHANDBM.
*
* This program determines maximum shears and moments and stores them
*   in the data file MOMENT&L.
*
*
DO CLOSE
*
SELECT PRIMARY
USE POSITNS
GOTO TOP
*
STORE 0.00000 TO VMAX
STORE 0.00000 TO VMIN
STORE 0.00000 TO MMAX
STORE 0.00000 TO MMIN
*
DO WHILE .NOT. EOF
*
*
    IF SHEAR1>VMAX
        STORE SHEAR1 TO VMAX
    ENDIF
*
    IF SHEAR2>VMAX
        STORE SHEAR2 TO VMAX
    ENDIF
*
    IF SHEAR1<VMIN
        STORE SHEAR1 TO VMIN
    ENDIF
*
    IF SHEAR2<VMIN
        STORE SHEAR2 TO VMIN
    ENDIF
*
    IF MOMENT1>MMAX
        STORE MOMENT1 TO MMAX
    ENDIF
*
    IF MOMENT1<MMIN
        STORE MOMENT1 TO MMIN
    ENDIF
*
*
    SKIP
ENDDO
*
*
    IF VMIN*(-1.00000)>VMAX
        STORE VMIN*(-1.00000) TO VMAX

```

```
ENDIF
*
*
  SELECT SECONDARY
  USE MOMENT&L
  APPEND BLANK
*
  REPLACE NAME WITH BMNAME
  REPLACE LOADCOMB WITH LCOMB
  REPLACE SHEAR WITH VMAX
  REPLACE MOMENTP WITH MMAX
  REPLACE MOMENTN WITH MMIN
*
  RELEASE VMAX,VMIN,MMAX,MMIN
*
  RETURN
```

```

* BMSELECT *
*
* This program selects beam sections based on factored
*   moment resistance.
*
* Sub-programs:  URESTRL; PICKSECT.
*
ACCEPT 'LEVEL' TO L
ACCEPT 'DO URESRTL (unrestrained length program) (Y/N)?' TO GO
*
IF GO='Y'
  'DO URESTRL
ENDIF
*
DO CLOSE
*
SELECT PRIMARY
USE BEAMS&L
GOTO TOP
*
DO WHILE .NOT. EOF
  STORE NAME TO BMNAME
  DO PICKSECT
  STORE 'L'+RLEN TO RLEN
*
  SELECT SECONDARY
  USE MOMENT&L
  COPY TO TEMP FOR NAME=BMNAME
  USE TEMP
  GOTO TOP
  STORE 0.00000 TO MOM
*
  DO WHILE .NOT. EOF
    IF MOMENTP>MOM
      STORE MOMENTP TO MOM
    ENDIF
    SKIP
  ENDDO
*
  USE BMSELECT
  GOTO TOP
  COUNT FOR &RLEN.>=MOM TO NUM
  IF NUM>0
    LOCATE FOR &RLEN.>=MOM
    SELECT PRIMARY
    REPLACE P.SECTION WITH S.SECTION
  ELSE
    SELECT PRIMARY
    REPLACE P.SECTION WITH 'NOSECTION'
  ENDIF
*
  SELECT PRIMARY
  SKIP

```

```

ENDDO
*
RETURN

```

```

* URESTRL *
*
* This is a sub-program of BMSELECT.
*
* This program calculates beam unrestrained length and stores the values
*   in the data file BEAMS&L.
*
* Sub-programs:  RESTRS; ELAR.
*
DO CLOSE
*
USE BEAMS&L
GOTO TOP
*
DO WHILE .NOT. EOF
  STORE # TO NBEAMS
  STORE NAME TO BMNAME
  STORE LENGTH TO LEN
*
  DO RESTRS
  DO ELAR
*
  USE BEAMS&L
  GOTO NBEAMS
  REPLACE RLENGTH WITH LR
  SKIP
ENDDO
*
RETURN

```

```

* RESTRS *
*
* This is a sub-program of URESTRL.
*
* This program creates a file of beam restraints RESTRS.
*
DO CLOSE
*
SELECT SECONDARY
USE RESTRS
DELETE ALL
PACK
*

```

```

SELECT PRIMARY
USE BEAMS&L
*
COPY TO TEMP FOR CONTYPE1='B' .AND. CONNAME1=BMNAME
USE TEMP
COUNT TO NUM
*
IF NUM>0
  GOTO TOP
*
  DO WHILE .NOT. EOF
    SELECT SECONDARY
    APPEND BLANK
    REPLACE S.POSITN1 WITH P.POSITN1
    REPLACE RTYPE WITH 'B'
  *
    SELECT PRIMARY
    SKIP
  ENDDO
ENDIF
**
**
USE BEAMS&L
COPY TO TEMP FOR CONTYPE2='B' .AND. CONNAME2=BMNAME
USE TEMP
COUNT TO NUM
*
IF NUM>0
  GOTO TOP
*
  DO WHILE .NOT. EOF
    SELECT SECONDARY
    APPEND BLANK
    REPLACE S.POSITN1 WITH P.POSITN2
    REPLACE RTYPE WITH 'B'
  *
    SELECT PRIMARY
    SKIP
  ENDDO
ENDIF
**
**
USE PANELS&L
COPY TO TEMP FOR JOIST1=BMNAME
USE TEMP
COUNT TO NUM
*
IF NUM>0
  GOTO TOP
*
  DO WHILE .NOT. EOF
    SELECT SECONDARY
    APPEND BLANK

```



```

        REPLACE RTYPE WITH 'P'
        REPLACE POSITN1 WITH POSITN1A
        REPLACE POSITN2 WITH POSITN2A
    *
        SELECT PRIMARY
        SKIP
    ENDDO
ENDIF
**
**
USE PANELS&L
COPY TO TEMP FOR JOIST2=BMNAME
USE TEMP
COUNT TO NUM
*
IF NUM=0
    GOTO TOP
*
    DO WHILE .NOT. EOF
        SELECT SECONDARY
        APPEND BLANK
        REPLACE RTYPE WITH 'P'
        REPLACE S.POSITN1 WITH P.POSITN1B
        REPLACE S.POSITN2 WITH P.POSITN2B
    *
        SELECT PRIMARY
        SKIP
    ENDDO
ENDIF
**
**
RETURN

```

```

* ELAR *
*
* This is a sub-program of URESTRL.
*
* This program calculates maximum unrestrained length of a beam from
* the restraint file RESTRS.
*
DO CLOSE
*
USE RESTRS
SORT ON POSITN1 TO TEMP
USE TEMP
*
STORE 0.00000 TO LR
STORE 0.00000 TO OLDPOS
STORE T TO CONT
*

```

```

DO WHILE CONT
*
  COUNT FOR RTYPE='P' .AND. POSITN1=OLDPOS .AND. POSITN2 OLDPOS TO NUM
*
  IF NUM>0
    GOTO TOP
    LOCATE FOR RTYPE='P' .AND. POSITN1=OLDPOS .AND. POSITN2 OLDPOS
    STORE POSITN2 TO OLDPOS
  *
  ELSE
    COUNT FOR POSITN1 OLDPOS TO NUM
  *
    IF NUM>0
      GOTO TOP
      LOCATE FOR POSITN1 OLDPOS
    *
    IF POSITN1-OLDPOS>LR
      STORE POSITN1-OLDPOS TO LR
    ENDIF
  *
    IF RTYPE='P'
      STORE POSITN2 TO OLDPOS
    ENDIF
  *
    IF RTYPE='B'
      STORE POSITN1 TO OLDPOS
    ENDIF
  *
  ELSE
    STORE F TO CONT
  *
    IF LEN-OLDPOS>LR
      STORE LEN-OLDPOS TO LR
    ENDIF
  *
  ENDIF
*
ENDIF
*
ENDDO
*
RETURN

```

```

* PICKSECT *
*
* This is a sub-program of BMSELECT.
*
* This program performs the actual beam section selection.
*
IF RLENGTH<1.25

```

```

    STORE '0000' TO RLEN
ENDIF
*
IF RLENGTH =1.25 .AND. RLENGTH =6.0
    STORE INT((RLENGTH*2.0)+0.75)*5 TO INC
    DO FOURZ
    STORE INCS TO RLEN
ENDIF
*
IF RLENGTH>6.0 .AND. RLENGTH<=10.0
    STORE INT(RLENGTH+0.75000)*10 TO INC
ENDIF
*
IF RLENGTH=10.0
    STORE 'OVER' TO RLEN
ENDIF
*
RETURN

```

```

    * FOURZ *
*
*This is a general subroutine
*It creates a (C,4) string INCS from the integer variable INC :10000
*   and fills in leading 0's.
*
*
*
IF INC<10
    STORE STR(INC,1) TO INCS
    STORE '000'+INCS TO INCS
ENDIF
*
IF INC<100 .AND. INC>=10
    STORE STR(INC,2) TO INCS
    STORE '00'+INCS TO INCS
ENDIF
*
IF INC<1000 .AND. INC>=100
    STORE STR(INC,3) TO INCS
    STORE '0'+INCS TO INCS
ENDIF
*
IF INC<10000 .AND. INC>=1000
    STORE STR(INC,4) TO INCS
ENDIF
*
IF INC>=10000
    ? 'Error in value'
    ? 'INC >= 10000'
    ? 'Please press 'ESC' to abort run'

```

```

? 'Thankyou'
ENDIF
*
RETURN

```

```

* CLOSE *
*
*This Commandfile closes secondary and primary files
*
*
SELECT SECONDARY
USE
SELECT PRIMARY
USE
*
RETURN

```

```

* LEVELS *
*
*This commandfile fills in the file LEVELS and also initializes the
* files BLX&L, MOMENT&L and BMDIAG&L.
*
DO CLOSE
*
SELECT PRIMARY
USE LEVELS
*
? 'This commandfile is used to fill in the relation LEVELS and also'
? ' to initialize the relations BLX&L, MOMENT&L and BMDIAG&L.'
?
?
*
ACCEPT 'LEVEL (2 chars or N to quit)' TO L
*
DO WHILE L<>'N' .AND. L<>' '
*
INPUT 'TOP OF STEEL ELEVATION' TO ELEV
*
APPEND BLANK
REPLACE NAME WITH L
REPLACE TOS:EL WITH ELEV
*
SELECT SECONDARY
*
USE BLX
COPY STRUCTURE TO BLX&L
*

```

```

USE MOMENT
COPY STRUCTURE TO MOMENT&L
*
USE BMDIAG
COPY STRUCTURE TO BMDIAG&L
*
SELECT PRIMARY
*
ACCEPT 'LEVEL (2 chars or N to quit)' TO L
ENDDO
*
CLEAR
RETURN

```

* HELP *

```

*
* This file lists the available programs and datafiles.
*
?
? 'Basic data files:'
?
? '    AXES : column line directions'
? '    LEVELS : floor level elevations'
? '    COLINES : column line locations'
? '    COLUMNS : column locations'
? '    BEAMS&L : beam locations'
? '    PANELS&L : floor panel locations'
? '    CONCL&L : beam concentrated loads'
? '    PLX&L : panel distributed loads'
? '    CASES : load combinations, load cases and factors'
?
?
? 'Secondary data files:'
?
? '    CLX : external column forces'
? '    BLX&L : external beam forces'
? '    BMDIAG&L : beam shear forces and bending moments'
? '    MOMENT&L : maximum beam shear forces and bending moments'
?
WAIT
?
? 'Applications programs in order of use:'
? '    XANDY : computes beam X-Y coordinates and beam lengths'
? '    AUTOLOAD : computes beam loads to file BLX&L and column loads to'
? '                file CLX'
? '    MOMENTS : computes beam shear and moments to files BMDIAG&L and MOMENT&L'
? '    BMSELECT : selects beams sections based on moment resistance'
?
? 'The program LEVELS can be used to input data into the LEVELS datafile: this'
? '    program will also initialize the level-dependant secondary data files.'
?
RETURN

```

Appendix B

dBASE II Commands

The building structure database has been implemented on the commercial relational database management system dBASE II, running on dy-4 Systems Inc. ORION micro-computer. Given the limitations of running on a micro-computer, the DBMS performs satisfactorily for demonstration purposes.

dBASE II provides all the basic relational operations for manipulating relational data files. It is limited in the mathematical operations it can perform; the only operations supplied are addition, subtraction, multiplication and division. For engineering application, this is hardly sufficient. Although a graphics capability would have been desirable the resources for implementing this were absent. The resources at my disposal did provide the basic facilities for creating a simple structural design system using the building structure database.

All programming for the design system is in dBASE II data manipulation language. Programs in this dBASE II language (command files) were created and edited with an external file editor. The DBMS responds almost identically to commands originating from the keyboard and commands in a command file.

To use the design system requires the knowledge of only a very few commands; those to actually run the programs and those to access the data files. The major dBase commands are as follows:

DO. The command "DO *commandfile*" executes a command file (program). This command is used within programs to execute sub-programs.

CREATE. This command is used to define the attributes of a new data file (relation). The attributes are also known as field variables. Each attribute is given a name, a data type and a length. Names are alpha-numeric up to eight characters long; there are some restrictions. The data type can be character string, "C", numerical value, "N", or logical value, "L". The length of the character string is specified; numerical values are given an overall length and a length of the decimal portion.

APPEND Once a data file has been created data can be put directly into the file by the command **APPEND**. dBASE II offers a screen editing feature to make data entry easy. This command will also take data from another file and append it to the file in use. "**APPEND FROM filename**". Only the attributes that match the file in use are appended, the attribute name and type must match. This command can also append a blank record to the data file in use, "**APPEND BLANK**". New data can then be entered with the **REPLACE** command.

USE To open a file in either mode, requires the use of the command **USE**. The syntax is "**USE filename**". The use of the command without a filename closes the file in use without opening a new one.

DISPLAY To view the data file in use the **DISPLAY** command is used. To see all records the command is "**DISPLAY ALL**". To see a certain number of records the command is "**DISPLAY NEXT n**", where "**n**" is a number. To see only certain desired records use "**DISPLAY FOR condition**", the condition is an expression that evaluates to true or false for each record. Only those records for which the condition is satisfied are displayed. To get a list of the data file attributes and their description use "**DISPLAY STRUCTURE**".

LIST. The command "**LIST**" is the same as "**DISPLAY ALL**" and the command "**LIST STRUCTURE**" is the same as "**DISPLAY STRUCTURE**".

COPY. Both complete data files or only their structures may be copied. The commands are "**COPY TO filename**" or "**COPY STRUCTURE TO filename**". The file copied from is the file in use and is not named.

SELECT. In dBASE II two data files may be open at the same time. To select the file in use the **SELECT** command is used. The options are "**SELECT PRIMARY**" and "**SELECT SECONDARY**". Although the field variables (attribute values) in the current records of both open files are available, the data file commands apply only to the data file in use. Therefore, if in **PRIMARY** mode the command "**LIST**" were given, the primary use data file would be listed; if in **SECONDARY** mode, the secondary file. The field variables in both primary and secondary files, which may have the same attribute names, are differentiated by pre-fixing the field variable name with "**P.**" or "**S.**".

STORE The only way to give a variable a value is with the STORE command. The command is "STORE *expression* TO *variable*". The value of the expression may be a character string, a logical or a numerical value.

REPLACE. To enter or change a value of a field variable the command REPLACE is used. The form is "REPLACE *field variable* WITH *expression*". The expression can be anything; a mathematical expression, a character, another field variable, etc. To operate on the whole file other forms of the expression are "REPLACE *field variable* WITH *expression* FOR *condition*" or "REPLACE ALL *field variable* WITH *expression*".

INSERT BLANK. This command inserts a blank record at the current position in the data file.

INPUT ... ACCEPT These are the two input commands. They both prompt the user with a character expression and a question mark and wait for a response. For example "INPUT *character expression* TO *variable*". The INPUT command can be used for any data. The ACCEPT command is similar to INPUT but accepts character string input without the need for quotes.

JOIN. This command joins specified portions of the primary and secondary data files. No common attribute is needed. The form is "JOIN TO *filename* FOR *condition* FIELD *field variables*".

GOTO. The data files in the relational database can be viewed as tables consisting of rows (records) and columns (attributes, field variables). Some relational operations deal with relations as a whole, but in many cases the user will want to deal with individual records and so must be able to advance up or down the table to the desired record. Only the field variables in the current record are accessible. One command for moving through a data file table is the GOTO command. The forms of this command are "GOTO TOP", "GOTO BOTTOM" and "GOTO RECORD *n*" where "*n*" is a record number.

SKIP. To advance to the next record in the table the command "SKIP" is used. The form "SKIP $\pm n$ " allows the user to move up or down the table the indicated number of records.

LOCATE **CONTINUE** Another means of advancing to a desired record is by use of a key attribute and the LOCATE command. The command "LOCATE FOR *condition*" examines each record and stops at the first record for which the condition is satisfied. To locate the next record following that also satisfies the condition the command CONTINUE is used.

DELETE **PACK** To delete the current record in the file in type "DELETE", To delete a specific record, "DELETE RECORD *n*", to delete several records, "DELETE FOR *condition*", to delete all records, "DELETE ALL". The delete command does not immediately destroy the deleted records. The records are marked for deletion and are ignored by the DBMS, but they are not destroyed until the command "PACK" is used. Deleted records can be recalled by the command "RECALL".

?. This command gives the values of the expression that follow it. "*variable*" gives the value of the variable. "*expression*" gives the value of the expression. "*character string*" simply gives the character string, and so is a useful print command for interactive programming.

COUNT. This command counts records. The form is "COUNT FOR *condition* TO *variable*". Or for the whole file: "COUNT TO *variable*".

Special variables. Two important special variables are "**#**" and "**EOF**". "**#**" is the record number of the current record of the file in use. "**EOF**" is a logical value that is true when the current record is at the end of file.

DO WHILE *condition* ... **LOOP** ... **ENDDO**. This is a program control command and is used only in the command files. All program steps between the DO WHILE statement and the ENDDO statement are repeated as long as the condition evaluates as true. The LOOP command short circuits the do-loop and returns control back to the DO WHILE.

IF ... ELSE ... ENDIF. The only other program control is the IF-ELSE- ENDIF command and it works in the usual manner.

RETURN. This is the last statement in any commandfile. RETURN returns control to the calling statement of a sub-program.

References

- [1] Date, C. J., *An Introduction to Database Systems*, Addison-Wesley, 1975.
- [2] Sangakarra, S. R., West, D. J., Haslett, T. G., "Development of CAD in Civil Engineering Reviewed", *Computer-Aided Design*, 16:3:141-147, May 1984.
- [3] Fenves, S. J. and Rasdorf, W. J., *Role of Database Management Systems in Structural Engineering*, Technical Report DRC-12-13-82, Design Research Center, Carnegie-Mellon University, Pittsburgh, PA, December 1982.
- [4] Holtz, N. M., Fenves, S. J., "Symbolic Manipulation of Design Constraints", *Engineering Software II*, ed. Adey, pp. 58-72, CML Publications, 1981.
- [5] Buchmann, A. P., "Current Trends in CAD", *Computer-Aided Design*, 16:3:123-126, May 1984.
- [6] Sandberg, G., "A Primer on Relational Database Concepts", *IBM Systems Journal*, 20:1:23-40, 1981.
- [7] Schaefer, M. J., Rebak, D. R., Fenves, S. J., *A Tutorial on Relational Databases Using Structural Engineering Examples*, Technical Report DRC-12-09-82, Design Research Center, Carnegie-Mellon University, Pittsburgh, PA, December 1982.
- [8] Kent, W., *Data and Reality*, North-Holland, 1981.
- [9] Eastman, C. M., "The Representation of Design Problems and Maintenance of their Structure", *Artificial Intelligence and Pattern Recognition in Computer Aided Design*, ed. Latombe, North-Holland, 1978.
- [10] Eastman, C. M., "Database Facilities for Engineering Design", *Proceedings of the IEEE*, 69:10:1249-1263, 1981.
- [11] Eastman, C. M., "General Purpose Building Description Systems", *Computer-Aided Design*, 8:1:17-26, January 1976.

- [12] Lafue, G. M. E., "Integrating Language and Database for CAD Applications", *Computer-Aided Design*, 11:3:127-130, May 1979
- [13] Sidle, T. W., "Weaknesses of Commercial Data Base Management Systems in Engineering Applications", *Proc. 17th ACM Design Automation Conf.*, pp. 57-61, June 1980
- [14] *dBASE II User's Manual*, Ashton-Tate, Los Angeles, California.
- [15] Vernadat, F. B., *A Commented and Indexed Bibliography on Data Structuring and Data Management in CAD/CAM 1970 to Mid-1983*, ERB-956, NRCC No. 2733, National Research Council of Canada, Division of Electrical Engineering, March 1984
- [16] Thompson, B. G. J., Young, J. S., "CEDAR3 in Practice: Using a Large, Integrated Computer-aided Building Design System in Government Design Offices", *Computer-Aided Design*, 12:3:139-148, May 1980
- [17] Hoskins, E. M., "Design Development and Description Using 3-D Box Geometries", *Computer-Aided Design*, 11:6:329-336, November 1979
- [18] Spoonamore, J. H., *CAEADS—Computer-Aided Engineering and Architectural Design System*, Technical Report, U.S. Army, Construction Engineering Research Ltd., Champaign, IL, August 1982
- [19] Eastman, C. M., "Prototype Integrated Building Model", *Computer-Aided Design*, 12:3:115-119, May 1980
- [20] Yasky, Y., "Transforming a Set of Building Drawings into a Consistent Database", *Proc. CAD80 4th Int. Conf. and Exhib. on Computers in Design Engineering*, Brighton, UK, pp. 101-108, March 1980
- [21] Grimbale, L., "Taskmaster: A Tool for Facilities Management" *Proc. Annual Conf. of Canadian Society for Civil Engineering, Computer Applications Division*, pp. 77-87, June 1983

END

11-106-186

FIN