UNIT 11 ORGANISING DATA

Objectives

After going through this unit, you should be able to:

- identify data as more important than the tools for its handling
- appreciate the limitations of the traditional approach to application system development
- enumerate the considerations that motivate a database approach
- classify data structures into records, files, object- relations etc.
- appreciate the stages of database design, such as logical design and normalization.

Structure

11.1 INTRODUCTION

Since computer professionals have proclaimed database and DBMS technology as revolutionary, many managers may regard the database approach as disruptive and may therefore avoid it; or they may think it must be embraced at any cost for their organization to remain efficient, modern, and competitive. Neither position is entirely acceptable, yet both contain some truth.

To say the database approach is entirely new reflects a lack of historical perspective. Data has always existed in organisations, and accountants and bureaucrats have traditionally controlled it to provide information in support of operations and management. The approach may seem radically new to people in data processing, but it is not. Many electronic data processing (EDF) departments have been so over whelmed with the programming aspects, that they have not applied well-known, sound management practices such as accountability and control. This "radical new concept" is merely a return to good management-of an old resource.

Growing recognition of data as a valuable corporate resource has led to the establishment of organisational units such as "Information Systems and Management Services" or "Information Resource Management" which have responsibility for data processing, management information systems, and the computer. Database technology is forcing a re-examination of the goals and operations of conventional EDP departments; in other words, it is forcing top management and users to change the way they view the EDP department.

Traditionally, EDP/MIS departments have been viewed as. managing the hardware and software systems. Yet these are only the factors of production, that is, the tools. The real product of the EDP/MIS department is data to support organisational operations and management decisions. Viewing the EDP/MIS department as a service function to the organisation focuses attention on the product to be delivered. ` 'he real resource being managed is data.

The database approach is rooted in an attitude of:

- Sharing valued data resources
- Releasing control of those resources to a common responsible authority
- Cooperating in the maintenance of those shared data resourcesThe database approach is:
- More than simply acquiring a DBMS
- More than collecting and storing data in one integrated database
- More than designating someone to centrally control the data

It involves tools to collect and manage the data, a responsible and cooperative attitude among users, and a database administrator.

An organisation must not become overly optimistic. The database approach is not a panacea for information systems and the use of computers. Neither is acquiring a DBMS. After all it is only a tool; it must be made to work in and for an organisation.

11.2 TRADITIONAL APPROACH TO APPLICATION SYSTEM DEVELOPMENT

A strong focus on application programs and processes characterizes the traditional approach to application system development. With a primary focus on processes, application systems often develop separately and operate independently. Figure 11.2 shows a set of application subsystems developed in a petroleum exploration and production organisation.

Level separation can result in a separation of data used for management reporting and analysis from data used in operations. For example, salespersons that enter customer orders into the computer system every day may also be asked to prepare a summary of the number of customers visited during the month in each customer class.

To retrieve data that is stored redundantly, the user must decide which application to use to obtain the data. Data updating, an even larger problem, must be coordinated across application systems to ensure that the same data is updated in the same way and at the same time wherever it exists.

Preparing top management reports is always more difficult. Independent master files with relatively homogeneous data cannot easily represent the data relationships across files. This makes it difficult to prepare top-level management reports containing aggregates of more heterogeneous data,

Developing a new application system presents additional problems. To use the data from an existing application may require adding new items of information or restructuring an existing file. Both alternatives require rewriting existing programs. With programs tied very closely to the physical layout of the data (e.g. the size of data values or the relative position of data items within a record), program modification is very difficult, which explains the great reluctance of most organisations to revise existing files or databases.

With no attempt to use existing data directly from the existing application, developing a new application means solving the data problem all over again - capturing the data, putting it in a form suited for the new application, and writing programs to store, retrieve, and update the data. Entirely new and separate master files are created, often resulting in duplication of already stored data. Data files are designed to suit the individual application. In the resulting collection of application systems, marked differences can occur in data quality. It may also be difficult to answer new ad hoc questions not previously considered in the design of the system.

Sometimes the designers try to minimize data redundancy. The result is often excessive transfer of data between subsystems. Without good multiple processing and reporting

facilities, increased data redundancy may be the best short run solution to the problem of intersystem file transfers.

In early computer systems, available storage media constrained the data structure. In any organization, all master files were stored on magnetic tape and the input data transactions on punched cards, Punched cards and magnetic tape (often viewed as a long strip of punched cards) impose a sequential structure on the file and a periodic batching discipline on its processing. In the early 1960s, direct access storage devices and online equipment expanded the possibilities for storing and processing data. Unfortunately, some systems in use today still carry the legacy of this batch processing mentality, During the 1960s and early 1970s, several DBMS emerged. Most systems provided facilities which. augmented a conventional programming language. A host language DBMS was designed to serve application programmers. While such a system may have allowed the integration of multiple, independent data files, it still relied on traditional tools for application systems development - namely, writing programs.

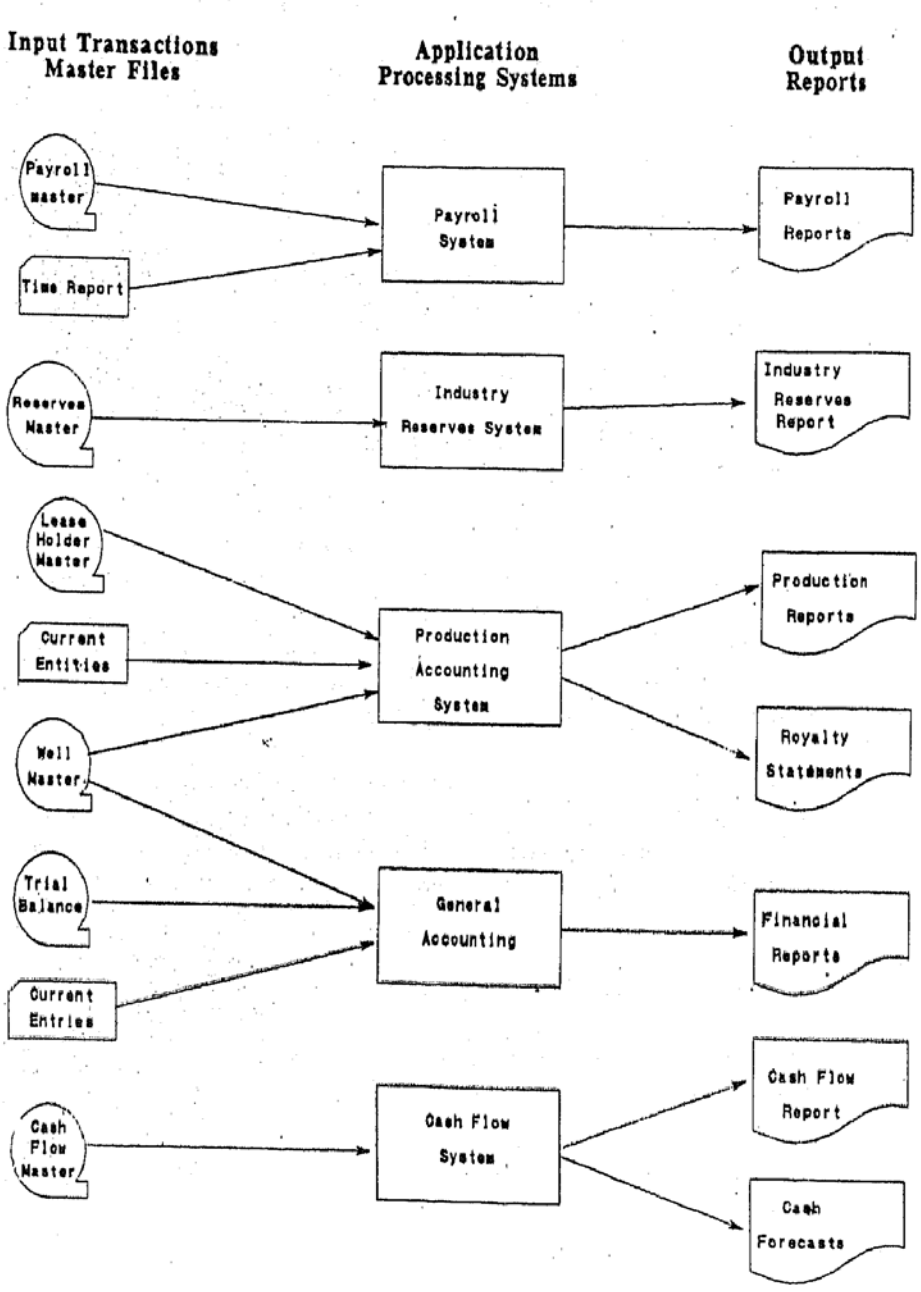


Fig. 11.2. Traditional Application System Development in a Typical Organization, in the Oil & Natural Gas Sector.

In this abbreviated schematic of the system, relatively independent application processes have their own master files. Six master files of data are associated with five application processing subsystems. One of the master files is shared by two applications. Designing relatively independent subsystems results in redundancy and inconsistency. Within the s ~, e organization, different codes represent the geographical location of wells, leases, aid lands. Product accounting and financial accounting each have their own location codes, while the legal department and government reports use a legal location (based on Khasra Khatauni) or a latitude-longitude designation (preferred by the engineering type of staff). Another inconsistency often results because `produce accounting' is done in barrels of oil or cubic meters of gas, while financial accounting is always done in rupees. If the barrels of oil are updated one month in the product accounting system and the corresponding rupees are updated the following month in the financial accounting system, the month-end reports to top management management will be inconsistent, sometimes even bizarre! Management is understandably perturbed when the reported barrels of oil produced and rupees of production from a lease for a given month do not correspond (using an approximate price per barrel to recalculate the revenue.)

In the traditional approach, data files are established as a by-product of application development. If the same items of data are needed in two applications, they are often kept redundantly, that is, duplicated in several application systems. Redundancy also occurs when multiple applications serve different organisational functions or managerial levels.

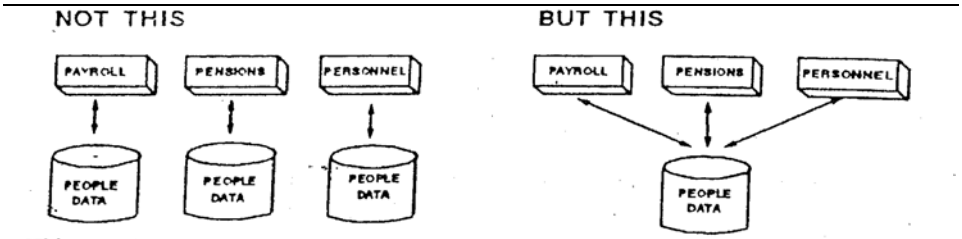
Some times, even with a host-language DBMS, an organisation still follows the traditional approach to application systems development.

Centralized development and operation has also characterized the traditional approach. The high cost of equipment and the scarcity of skilled personnel dictated strong central execution and control of system development. As a result, users often faced unavailable or unreliable data. This led to private, manual collections of data maintained by end users. The emergence of relatively inexpensive Personal Computers (with the power and capacity of a mainframe of 20 years ago) now makes it possible for these local; private collections of data to be mechanized. Local organizational units press for decentralization believing that their information problems would be overcome if the data processing operations were locally controlled. In fact, local users often end up making many of the same mistakes made by central DP a decade earlier, rediscovering the lessons learned from past experience in the computer industry.

The above discussion reveals many symptoms of problems in data processing. Although the example illustrated in Fig. 11.2. may be worse than current practice in some organisations, it does illustrate many of the problems stemming from the traditional approach to application system development. Such problems motivate the database approach.

The essence of the data base concept, can be appreciated from the following schematic diagram:

"Where data can be shared between different systems, it should be"



This Requires:

* Flexible data storage and access methods to meet the different needs of different users
* Absolutely reliable recovery methods

11.3 MOTIVATION FOR DATABASE APPROACH

Several problems may motivate management in an organisation to adopt the database approach- focusing on the problem of data, and perhaps acquiring a DBMS. Difficulties may be manifest in an inability to get something done, or do economically, quickly, and accurately. Though this need may initially justify acquiring a DBMS, solving a particular problem or building a particular system must not be the sole criterion for selecting a DBMS. Management must look at the application environment and the data processing requirements from a global perspective.

Five major problems motivate a database approach:

* Inability to get quick answers to "simple" ad hoc requests.
* High development costs.
* Low responsiveness to change.
* Low data integrity and quality.
* Inadequate data model of the real world.

Four major trends in the evolution of DBMS especially those based on a relational approach have led to an increased interest on the part of managers to adopt the new technology.

1) increase in commercially available products
2) recognition of relational databases as practical and desirable approach to cope with the information management needs of organisation.
3) emergence of relational DBMS products that satisfy a broad range of business Requirements.
4) increasing possibility of associating people with limited database background in the design of relational databases.

11.4 TAXONOMY OF DATA STRUCTURES

In current DBMS literature, systems are most often classified according to the logical structures of the underlying data "model",* that is, the class of data structures which can be defined in and processed by a given DBMS. A database is formally defined to a DBMS by writing statements in its Data Definition Language (DDL).

The following paragraphs introduce the taxonomy of data structures by naming and briefly describing various types of structures. At this point the reader should visualize the taxonomy as presented in Figure 11.4. This brief overview provides an initial understanding by placing the pieces of the taxonomy into a consistent, overall picture.

A database is a collection of information about entities (People, organisations, positions, policies, orders, parts, projects, events, etc.). Attributes describe entities (e.g., age of person, budget of organisation). A particular entity ("instance") is described by the values of a set of attributes. (Age of "Rakesh Kumar" is 52). The database designer selects some set of attributes to describe entities in the database.

The first division in the taxonomy of data structures is based upon whether or not the attributes or data items are grouped into records. (Records represent entities). Historically, most data structures have been record-based. Most people in data processing are familiar' with collections of data consisting of files of records. Assuming a grouping of data items at the outset of database design leads to some difficulties in the resulting structures, particulaly.when they expand to encompass more of the data in an organisation. Recognizing these limitations, several authors proposed a class of data structures which omit the dual notion of entity-attribute, replacing it with the single

* The term "model" is widely used although it is somewhat misleading: The phrase "data structure class" is more accurate. Something is a "model" if it bears a likeness to or is an imitation of something else. A defined database is a data model of reality to the users; a DBMS is a data modeling system.

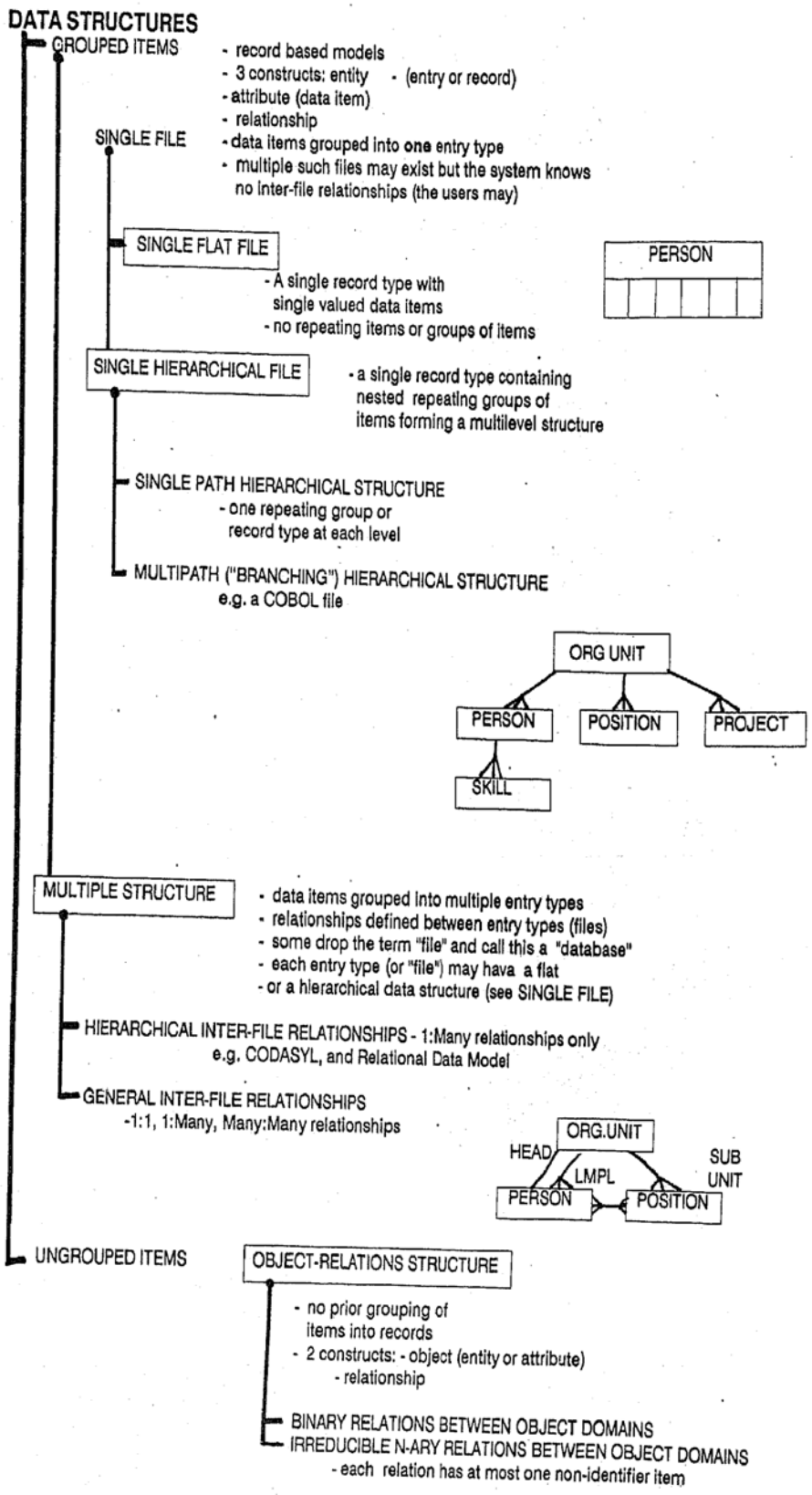


Fig. 11.4 A Taxanomy of Data Structures (Taken from Everest Data Base Management)

motion of "object". These are called the **object-relation** class of data structurers.

Data structures with grouped items can first be divided into single-file structures, and multifile structures. In a single-file structure, all information in the database can be grouped according to *a* single entry type. In other words, there is one primary entity connotation for the entire database. Single-file structures are further divided into flat file structures and hierarchical file structures. In a single hierarchical file structure, each primary entity may have subentities (that *is,* the primary record, corresponding to the primary entity, contains nested repeating groups of data items representing attributes of the subentities). Single hierarchical structures are further divided into single path and multipath ("branching") hierarchical file structures or data structurers.

A multifle data structure consists of multiple, related files. All the data items **in** the database are grouped into multiple record types. There is no longer one primary entity or record type in the database. Entities may exist **independently** of each other and may be related to each other. Multifile data structures offer greater flexibility than single-file structures, therefore, enabling greater fidelity in modelling the world of interest to the user(s).

Multifile data structures are further divided into those which only permit a hierarchical relationship between entity types and those which permit a general relationship between entity types. (Multifile data structures are also further divided depending on whether each file must be flat or may be a hierarchitcal structure.)

11.5 DATA BASE DESIGNING',

There are two aspects to the database design. First, the data items, entities, relationships, and other concepts need to be translated into their equivalent logical database structures of tables and columns. Next, the logical database design needs to be turned into physical database structures such as indices, permits, and integrities.

Logical Design

The process of database logical design entails translating entities and relationships into database tables. *For* example, consider a one-to-one relationship between two entities. This could entail the fact that each employee has a single office and each office has a single employee in it. In this case, the translation from the entity-relationship models to the relational database model is fairly simple. An employee table is constructed and that table has columns for both employee and location.

For one-to-many relationships, the process is slightly more complicated. If a sales-person has several sales outstanding, this *is* a one-to-many relationship. In this case, two tables are constructed. One table is for the entity salesperson. This table has a unique key for the salesperson, say the last name.

The sales table needs to have one line for each sales order. It also needs to have the name of the salesperson. The salesperson name is the same name as is *in* the other table and is known as a foreign key. By joining the two tables together, the user can find out information about the relationship. Some information in the entity-relationship model cannot be reflected in a relational database. For example, the cardinality of a relationship cannot be directly reflected in a design consisting of a series of tables. Instead the cardinality aspect of a relationship would have to be implemented as an integrity constraint that regulates the input and update of information in one or more tables.

Translating an entity-relationship model into a relational database logical design usually consists of two steps. First, the formal method of normalization is applied to the entities and relationships to break them down into a series of tables.

Normalization ensures that updates to the database can be performed in a consistent fashion and that data are not overly redundant.

The next step is to change the logical design to reflect the native Of the applications that will use that data. Normalization, byredttaing redundancy, also makes retrieval of data more difficult because many St .alt tables May have to be joined together. Based on the types of applications that will' use the data, the designer may add redundancy or otherwise violate the rules bf normalization.

Normalization

Normalization is a formal method that attempts to solve the problem of update anomalies. An anomaly remits when data are stored twice and are only updated once. It should be stressed that normalization is only one of several database design techniques. Although normalization reduces update anomalies to data, it does often make retrievals more inefficient by breaking data up into many different tables.

The process of normalization consists of breaking tables down into smaller tables, There are various levels of normalization. This unit contains a brief discussion of the first three normal forms. There are also fourth and fifth normal forms available and research continues on further extensions of this technique. The reader is referred to C.J. Date, `An Introduction to Database Systems', Volume 1 (4th ed., Addison Wesley, 1986, Reading, Mass.) for a more formal treatment of this technique.

First normal form consists of eliminating repeating groups. A repeating group means that a table has several columns, each of which represents the same piece of data. For example, an employee might have multiple phone numbers. First normal form is violated when there are two columns, one for each phone number. The update anomaly results when a user attempts to update a phone number. The application program would have to search both columns to find the relevant phone number.. In a design conforming to first normal employee information would be broken into two tables, The employee table would have one row for each employee. The phone table would consist of two columns, one for employee name and the other for the phone number.

A relation is in second normal form if every monkey column is dependent, directly or indirectly, on the key for that table. For example, a projects table could have as the primary key the columns employee and project name. Storing a department's budget in the same table would be a violation of second normal form because the budget is not dependent on either portion of the key.

Third normal forms states that any monkey column in a table must be dependent on the whole key for the table. Using the same projects table, storing employee office locations would be a violation of third normal form because the office location is only dependent on a portkn of the key- the employee name and not the project name.

To achieve normalization, the table would be broken up into two tables. One would have project-specific information, such as the number of hours worked by the. employee. The second table would contain only employee-specific information such as office location. The update anomaly in a violation of third normal form exists because the employee office location exists once for each project that the employee participates in. If the user updates the office location, he would have to ensure that every instance of the phone number is updated.

As an be seen, the process of normalization consists of breaking a table down into a series of tables, each containing a well-defined group of information. The rigorous process of normalization helps the database designer identify potential problems in the design and some ways of curing those problems.

11.6 SUMMARY

In this unit, we have seen the need for an approaeh to Data Base R anagement as an alternative to the traditional approach of establishing data files as a by-product of application development. The motivation for data base approach arises from being able to provide the users direct access to his data and to be able to respond to ad-hoc queries. In order to understand the concept pf the data base, a brief taxonomy of data structures was discussed.

In order that the real world's needs of information *for the* organization are met, some effort is required for data base designing. The concepts of logical design and normalisation were discussed briefly.

11.7 SELF ASSESSMENT EXERCISE

1) In what sense does the database constitute the `imago' of an organization?
2) What are the main disadvantages of separately developing application systems and their associated data files?
3) List the five problems which may motivate an organization to move toward the database approach, explain the significance of each one to a manager to an organization.
4) What is the effect of normalization on database storage and database performance?

11.8 FURTHER READINGS

1. Atre S. *Database Structural Techniques for Design, Ferformance & Management,* John Wiley & Sons, 1980

2. Date C.J. *A Guide to DB2* Addison-Wesley. 1984

3. Everest, G.C., *Database Management: Objectives System functions & Administration, McGraw* Hill, 1986.

4. Ven Halle Fleming, *Handbook of Relational Database Design,* Wesley, 1990