

# Contents

<i>Preface to Second Edition</i>	v
<b>Part 1: Fundamentals</b>	<b>1</b>
1 Databases as Abstract Machines	3
2 Key Concepts	15
3 Databases and Information Systems	27
4 The Data Management Layer	37
<b>Part 2: Data Models</b>	<b>45</b>
5 The Relational Data Model	47
6 The Classic Data Models	65
7 The Object-Oriented Data Model	72
8 The Deductive Data Model	80
9 The Post-Relational Data Model	92
<b>Part 3: Database Management Systems: Interface and Toolkit</b>	<b>99</b>
10 DBMS Interface: SQL	101
11 DBMS Toolkit – End-User Tools	118
12 DBMS Toolkit – Application Development Tools	124
13 DBMS Toolkit – DBA Tools	132
<b>Part 4: Database Management Systems: Kernel</b>	<b>137</b>
14 Data Organisation	139
15 Access Mechanisms	148
16 Transaction Management	154
17 Other Kernel Functions	166
<b>Part 5: Database Management Systems: Standards and Commercial Systems</b>	<b>173</b>
18 Post-Relational DBMS: SQL3	175
19 Object-Oriented DBMS: ODMG Object Model	181
20 Microsoft Access	187
21 Oracle	195
22 O <sub>2</sub>	203
<b>Part 6: Database Development</b>	<b>209</b>
23 Normalisation	214

24	Entity-Relationship Diagramming	232
25	Object Modelling	254
26	Physical Database Design	273
27	Database Implementation	289
<b>Part 7: The Planning and Administration of Database Systems</b>		<b>297</b>
28	Strategic Data Planning	299
29	Data Administration	308
30	Database Administration	314
<b>Part 8: Trends/Features</b>		<b>323</b>
31	Distribution	325
32	Parallelism	336
33	Multimedia Data	348
<b>Part 9: Applications of Databases</b>		<b>357</b>
34	Data Warehousing	359
35	On-Line Analytical Processing	367
36	Data Mining	372
37	Networked Databases	376
	<i>Sample Solutions</i>	384
	<i>Glossary and index</i>	394

# Preface to the Second Edition

## Mission

The main aim of this work is to provide one readable text of essential core material for further education, higher education and commercial courses on database systems. The current volume is designed to form a consolidated, introductory text on modern database technology and the development of database systems.

It is undoubtedly true that database systems hold a prominent place in most contemporary approaches to the development of information systems. It is this practical emphasis on the use of database systems for information systems work that distinguishes the current volume from other texts on the subject. This work therefore forms a companion volume to *Information Systems Development: an introduction to information systems engineering*, also published by Macmillan.

The text is built from the author's experiences of running a number of academic and commercial courses on database technology and database development for several years.

## Changes to the Second Edition

The second edition has been much expanded from approximately 300 to some 500 pages. This expansion has enabled us to cover particular topics in more depth:

1. Greater coverage has been devoted to the kernel functions of contemporary DBMS: file organisation, access mechanisms, transaction management, etc.
2. Material has been added on contemporary database applications such as data warehousing, data mining and on-line analytical processing.
3. A chapter on Microsoft Access has been included.
4. Material has been provided on extended-relational and object-oriented data management standards.
5. The topic of strategic data planning has been addressed.
6. More detail is provided on the issue of physical database design and database implementation.

We have also updated many chapters with developments since 1996 and re-organised a number of chapters to provide a more coherent presentation of key topics.

## Parts

The text is organised into a number of parts:

1. *Fundamentals*. The introductory chapters set the scene for the core of the text. We begin with a description of the key features of a database system. We then move on to define some key concepts and illustrate the importance of database technology to contemporary information systems. We close this part with an example of some of the key features of the data management layer of an information technology system.
2. *Data models*. Part two explores a number of contemporary architectures for database systems. Because of its current dominance, particular emphasis is made of the relational data model, although developments in object-oriented data management explain the extended coverage offered. We also consider two historically important data models – hierarchical and network – and another data model significant because of its association with ‘intelligent’ applications – the deductive data model. We close this part with a consideration of a hybrid data model – the extended or post-relational data model – which is important because of its popularity among contemporary DBMS.
3. *DBMS interface and toolkit*. Part three addresses the issue of the standard interface to DBMS – SQL. We also consider the toolkit of products that can be used in association with DBMS. The toolkit is divided into end-user tools, database administration tools and application development tools.
4. *DBMS kernel*. Part four considers some of the critical functions of the kernel of a database management system: file organisation, access mechanisms and transaction management.
5. *Systems and standards*. Part five discusses the architectures and facilities available in three contemporary DBMS, two relational (Microsoft Access and Oracle) and one object-oriented (O<sub>2</sub>). We also consider developments among the SQL3 and ODMG standards for DBMS.
6. *Development*. Part six represents a discussion of the major techniques utilised in the design and implementation of database systems. We discuss the techniques of entity-relationship diagramming, normalisation and object modelling, each of which is particularly useful in the early stages of database design. We conclude with a review of a number of issues involved in the physical design of databases and database implementation.
7. *Planning and administration*. The issue of planning and administering data in organisations is considered in this part. We first consider the issue of using corporate data models to plan for database systems. This is followed by a consideration of the importance of administering data within organisations. We conclude with a review of the database administration function.
8. *Trends*. Part eight considers new applications for databases. It consists of three chapters which look at areas having a significant effect on the functionality of database systems: the issue of parallelism, the issue of distribution and the issue of handling complex data.
9. *Applications*. This part discusses four contemporary applications for database systems: data warehousing, OLAP, data mining and networked databases.

## Feedback

The author is keen to receive feedback on the current text. Any comments or suggestions should be addressed to the author (pbeynon@glamorgan.ac.uk). A teaching pack is available from the author. My thanks to staff and students at the University of Glamorgan and to Professor Frank Sumner for writing chapter 32.

## Suggested Routes through the Material

Below we indicate some ways in which the material may be used in educational modules and courses. This is meant to be purely suggestive.

### INTRODUCTORY MODULE IN DATABASE SYSTEMS

An introductory module in database systems should provide a basic understanding of database concepts and an appreciation of database development issues. For this purpose the following chapters are suggestive: *three chapters in fundamentals; chapter 5: The Relational Data Model; chapter 10: DBMS Interface – SQL; chapter 20: Microsoft Access; chapter 23: Normalisation; chapter 24: Entity-Relationship Diagramming.*

### INTERMEDIATE MODULE IN DATABASE SYSTEMS

An intermediate module should impart an understanding of the architecture of a DBMS and issues pertaining to physical design and database administration. For this purpose the following chapters are suggestive: *chapters 11–13; chapters 14–17; chapter 21: Oracle; chapter 26: Physical Database Design; chapter 27: Database Implementation; chapter 30: Database Administration.*

### ADVANCED MODULE IN DATABASE SYSTEMS

Advanced modules in database systems should provide an awareness of alternative approaches to handling data and developing database systems over and above that provided by the relational data model and relational databases. For this purpose the following chapters are suggestive: *chapter 6: The Classic Data Models; chapter 7: The Object-Oriented Data Model; chapter 8: The Deductive Data Model; chapter 9: The Post-Relational Data Model; chapter 22: O<sub>2</sub>; chapter 25: Object Modelling; chapter 28: Strategic Data Planning; chapter 29: Data Administration; chapters 31–33.*

### TRADEMARK NOTICE

Access™ and SQL Server™ are trademarks of Microsoft.

O<sub>2</sub>™ is a trademark of O<sub>2</sub> Technology.

Oracle™ is a trademark of the Oracle corporation.



# Classic Data Models

## OBJECTIVES

---

1. To describe the features of two traditional data models, the hierarchical data model and the network data model
  2. To compare the hierarchical and network data models with the relational data model
- 

## 6.1 Introduction

In this chapter we discuss the two so-called classic data models: the hierarchical data model and the network data model. In chapter 2 we defined the idea of a classic data model in terms of record-orientation. In this sense, we might include the relational data model in among the classic data models in the sense that rows correspond closely to the idea of records. Here, the network and hierarchical data models are also described as being classic in the sense that they formed the basis for the first commercially available DBMS. Although currently relational systems have tended to outstrip network or hierarchical DBMS, a large number of legacy systems still exist running under these data models.

## 6.2 The Hierarchical Data Model

In chapter 5 we discussed how the relational data model was originally specified in a series of papers by E. F. Codd. The hierarchical data model does not have the same unified, theoretical foundation. This data model can generally be said to have developed after the fact from an examination of existing implementations. Probably the most prominent of DBMS adhering to a hierarchical approach is IBM's IMS (Information Management System). However, rather than concentrating on any one DBMS, in this section we provide a brief description of a number of key features which characterise all hierarchical DBMS.

### 6.2.1 DATA DEFINITION

The hierarchical data model uses two data structures: record types and parent-child relationships. A record type is a named data structure composed of a collection of named fields such as courseCode and courseName. Each field is used to store a simple attribute and is given a data type such as integer, character, etc. A parent-child link is a one-to-many relationship between two record types. The record type at the one end of a relationship is said to be the parent record type,

such as course; that at the many end the child record type, such as module. Hence, a hierarchical schema is made up of a number of record types related by parent-child links.

Consider, for instance, the relationship between courses, modules and students. Courses can be considered the parent record type of modules in the sense that there are many modules making up one course. The record-type modules can in turn be considered the parent record type of students as there are many students taking each module. Using these data structures we might specify such a schema as below:

**SCHEMA:** University

**RECORD:** Courses

PARENT: none

FIELDS (

courseCode: CHARACTER(6),

courseName: CHARACTER(20),

validationYear: DATE)

KEY: courseCode

ORDER BY courseCode

**RECORD:** Modules

PARENT: Course

FIELDS (

moduleName: CHARACTER(20),

staffNo: INTEGER(6),

level: INTEGER(1))

KEY: moduleName

ORDER BY moduleName

**RECORD:** Students

PARENT: Module

FIELDS (

studentNo: INTEGER(6),

studentName: CHARACTER(20),

termAddress: CHARACTER(30))

KEY: studentNo

ORDER BY studentName

Note that this schema has many similarities with the relational schema described in section 5.2. The key differences are:

1. That the data structures are different. In the hierarchical data model we have the record type, while in the relational data model we have the relation.
2. Relationships are implemented differently. In the hierarchical data model relationships are implemented via parent-child links. In the relational data model relationships are implemented via foreign keys.

### 6.2.2 DATA MANIPULATION

In the hierarchical data model, data manipulation is accomplished by embedding

database access functions within some standard programming language (a host language). The program segment below illustrates some of the key features of this approach:

```
WHILE DB_STATUS = 0 DO
  BEGIN
    WRITELN(module.moduleName);
    GET NEXT module WHERE level = 1;
  END;
```

The statement is made up of a series of read commands encased in a loop with an appropriate terminating condition. `DB_STATUS` is a system variable which is set to 1 when the end of file is reached. In this sense, the statement is a procedure – a set of statements expressed in some sequence. It is analogous to procedures expressed in such high-level languages as C. For this reason hierarchical query languages are generally described as being procedural query languages.

### 6.2.3 DATA INTEGRITY

There are a number of inherent integrity constraints in the hierarchical model which come into being whenever we express a hierarchical schema. Two examples of such constraints are given below:

1. No record occurrence, except a root record, can exist without being linked to a parent-record occurrence. This means that a child record cannot be inserted unless it is linked to a parent record and also that deletion of a parent record causes automatic deletion of all linked child records.
2. If a child record type has two or more parent record types, then a child record must be duplicated once for each parent record.

## 6.3 The Network Data Model

In historical terms the network data model is seen by many as the successor to the hierarchical data model. The dominant influence in the development of the network data model was a series of proposals put forward by the Database Task Group (DBTG) of the Conference on Data Systems and Languages (CODASYL) in 1971 (DBTG 1971). During the 1970s the majority of commercial DBMS adhered, albeit loosely, to a network data model. More recently, ANSI made a recommendation for a network definition language (NDL) in 1986.

### 6.3.1 DATA DEFINITION

Like the hierarchical model, the network data model has two data structures: record types and set types. A record type is similar in concept to the record type of the hierarchical model except that fields may be used to store multiple values or represent a composite of values which repeat. For example, a record type students may have the following fields: `studentNo`, `studentName` and `studentProfile`. `StudentProfile` could be considered a composite in that it clusters together a repeating group made up of `courseName`, `year` and `grade`. A set type is a description of a one-to-many relationship between two record types.

It is useful to distinguish between record types and record occurrences and set types and set occurrences. A record type is part of a schema definition. a record

occurrence is part of the extension of a database. The same applies to set types and set occurrences. Each set type is made up of a name for the set type, an owner record type and a member record type. A set occurrence is a group of one owner record and one or more member record occurrences.

A network schema for our academic database might look as follows:

**SCHEMA:** University

**RECORD:** Courses

FIELDS (  
courseCode: CHARACTER(6),  
courseName: CHARACTER(20),  
validationYear: DATE)  
ACCESS: HASH USING courseCode  
DUPLICATES ARE NOT ALLOWED FOR courseCode

**RECORD:** Modules

FIELDS (  
moduleName: CHARACTER(20),  
staffNo: INTEGER(6),  
level: INTEGER(1))  
ACCESS: VIA SET  
DUPLICATES ARE NOT ALLOWED FOR moduleName

**RECORD:** Students

FIELDS (  
studentNo: INTEGER(6),  
studentName: CHARACTER(20),  
termAddress: CHARACTER(30),)  
ACCESS: VIA SET  
DUPLICATES ARE NOT ALLOWED FOR studentNo

**SET:** Runs

OWNER: Course  
MEMBER: Module  
ORDER BY moduleName ASCENDING

**SET:** Has

OWNER: Module  
MEMBER: Student  
ORDER BY studentNo ASCENDING

In the schema above three record types are declared and two set types. Each record type defines the field names and associated data types. The manner in which the record occurrences are to be accessed is also specified. For instance, Courses records are to be accessed using a hashing procedure. Modules and Students records are to be accessed with regard to record membership of a specified set. This means that record occurrences are physically located close to the owner of the occurrence in a specified set.

Each set type specifies the names of owner and member record types. An order

by clause is used to specify the logical position of member records in each set occurrence. Both sets in the schema above sort records on some field.

### 6.3.2 DATA MANIPULATION

The functionality of a network data manipulation language (DML) is essentially similar to that of a hierarchical DML. A series of database-specific functions will be embedded in some host language. Such functions can be divided into three main groups: data navigation commands which are used to set currency indicators to specific records and set occurrences; retrieval commands for accessing the contents of current records; update commands for changing the contents of record and set occurrences.

It is important to remember that the host programming language and database system are two separate software systems. They are connected together by a common interface. Since DML commands are essentially record-at-a-time, an important concept in network data manipulation is the idea of currency. That is, a method of determining which is the current record occurrence being processed. A number of currency indicators are usually maintained: the most recently accessed record in a program or terminal session; the most recently accessed record of a record type; the most recently accessed record of a set type.

The host programming language needs a series of local program variables which can hold the contents extracted from database records. This set of local program variables is normally referred to as the user's work area.

### 6.3.3 DATA INTEGRITY

Integrity in the network model mainly concerns determining set membership and insertion mode. A set's membership may be defined as mandatory or optional. If the status is mandatory then the system will force every member record occurrence to belong to a given set occurrence. An insertion mode is also specified for set members. If the insertion mode is manual then application programs must explicitly insert member records into an explicit set occurrence. If the insertion mode is automatic then when a member record is created it is automatically inserted into the current set occurrence. Hence we may add a clause to the Has set type:

```

SET: Has
OWNER: Module
MEMBER: Student
ORDER BY studentNo ASCENDING
INSERTION IS MANDATORY, AUTOMATIC

```

The schema in section 6.3.1 illustrates how a range of integrity constraints can be added to a database specification. In this example we have prohibited duplicates against key fields. Integrity maintenance may also be enhanced by specifying ranges for data items and by incorporating procedural checking before insertion or deletion of data.

## 6.4 Comparison of Classic Data Models with the Relational Data Model

In the table below we compare the classic data models with the relational data model against the three component parts of a data model:

	<b>Structure</b>	<b>Manipulation</b>	<b>Integrity</b>
<b>Hierarchical</b>	Tree	Navigational record at a time	Automatic support for certain forms of referential integrity
<b>Network</b>	Network	Navigational record at a time	Automatic support for certain forms of referential integrity
<b>Relational</b>	Table	Non-procedural	Support varies among products

The main difference between the approaches is that the structure of classic databases encourages navigational or record-at-a-time processing. This means that DBMS based on such data models generally maintain procedural DMLs. In contrast, relational systems work with entire tables and encourage the production of non-procedural interfaces. Note, however, that until quite recently relational systems were poor at directly supporting entity and referential integrity. In contrast, because of the inherent pointer structures in hierarchical systems certain forms of referential integrity must be automatically supported.

## 6.5 SUMMARY

- 1. The hierarchical data model uses two data structures: record types and parent-child relationships.**
- 2. A record type is a named data structure composed of a collection of named fields.**
- 3. A parent-child link is a one-to-many relationship between two record types.**
- 4. In both the the hierarchical and the network data models, data manipulation is accomplished by embedding database access functions within some standard programming language.**
- 5. A number of inherent integrity constraints exist in both the hierarchical and network data models including a form of referential integrity.**

**It is undoubtedly true that DBMS adhering to the relational data model have become much more dominant than those adhering to the hierarchical or network data models in recent times. Having said this, there are still a large number of 'legacy' systems built around hierarchical and network databases. Also, many of the features of the classic data models have reappeared recently under a new guise – the object-oriented data model. This is the topic of the next chapter.**

## 6.6 EXERCISES

1. Add a lecturer record type to the schema in section 6.2.1.
2. Does lecturer have a parent record type?
3. Add a lecturer record type to the schema in section 6.3.
4. What sets does the lecturer record type participate in?
5. Discuss why hierarchical and network databases are described as navigational databases.
6. What do you think are the advantages and disadvantages of procedural DMLs as compared to non-procedural DMLs?

## 6.7 References

DBTG (1971). Report of the CODASYL Database Task Group, ACM.