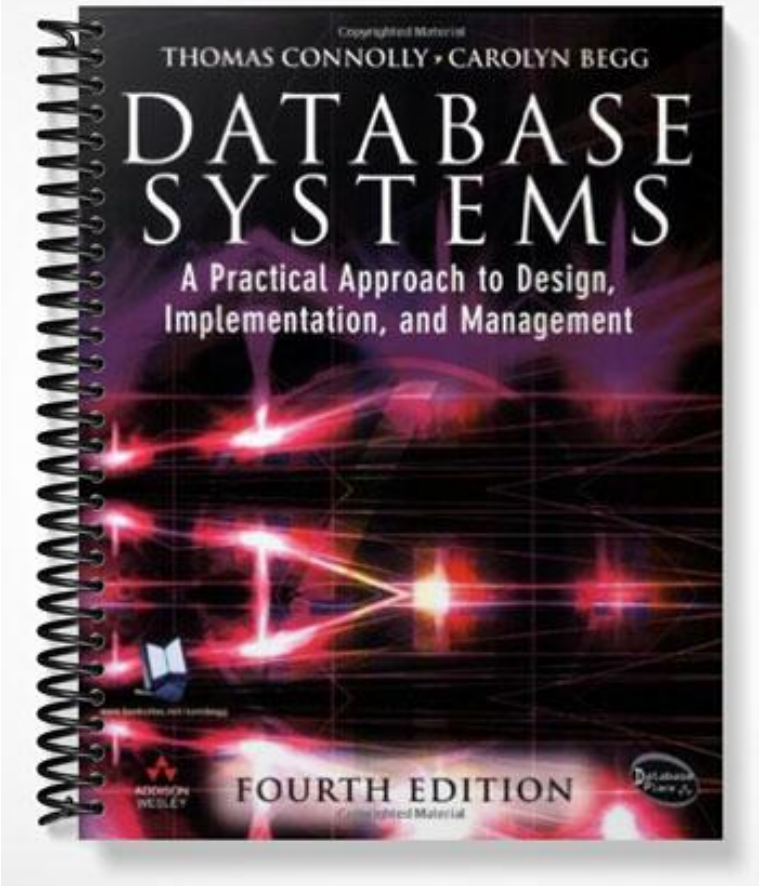
SOLUTIONS MANUAL



PART II

POINTS TO EMPHASIZE AND TEACHING HINTS

Points to Emphasize

Part 1

Background

CHAPTER 1

Introduction to Databases

Aims

The aim of this chapter is to introduce the student to the study of database systems. To help the student appreciate that databases are widespread nowadays the chapter starts by giving some common examples of database systems. The chapter then discusses the development of the database approach, compared to the file-based approach, and introduces the DBMS environment. Finally, the chapter discusses the advantages and disadvantages of DBMSs.

Points to emphasize

- Some common uses of database systems.
- The characteristics of file-based systems.
- The problems with the file-based approach.
- The meaning of the term database.
- The meaning of the term Database Management System (DBMS).
- The typical functions of a DBMS.
- The major components of the DBMS environment.
- The personnel involved in the DBMS environment.
- The history of the development of DBMSs.
- The advantages and disadvantages of DBMSs.

Possible difficulties

The material covers a wide area and it may be advisable to focus material under particular headings. One area that may subsequently confuse students is that of the components of a DBMS environment. Students can interpret the components to be only the actual software, the 3-level architecture, or even the functions of a DBMS. However, ensure that students appreciate the other components, namely hardware, the data itself, procedures, and users.

There is a tendency for students to think that the subject of database systems is easy. Thus, it is worth emphasizing that particular skills are required to use DBMSs effectively, especially in the area of database design.

Teaching hints

When introducing the subject, students could be encouraged to give their own examples of occurrences of databases and database systems.

The problems of the file-based approach can be highlighted by getting the students to use the *Wellmeadows Hospital* case study to produce examples of file systems with details of the individual files. This can then provide a basis for looking at the database approach.

Considering the database approach for the data held in a collection of individual files, should demonstrate the obvious need for a proper design methodology in order to satisfy all user requirements, and provide flexibility for future changes. It should also serve to emphasize that database design is likely to involve many different users owing to the potentially wide scope of the area being considered.

The history can be left as private reading, which could supplement coverage of the file-based and database approaches.

Students tend to view advantages and disadvantages as a list of points to be memorized. Getting them to think **how** file-based systems are used in contrast to a database system can help them to understand how the points arise. It is also worth pointing out that using a DBMS will not compensate for a bad database design, and in the worst cases, DBMSs are used as glorified file-based systems, especially the PC-based DBMSs. Coverage of the advantages/disadvantages is well worth a revisit at a later part of the course, when the student has acquired a deeper understanding of the concepts and the discussion will be less abstract.

CHAPTER 2

Database Environment

Aims

The aim of this chapter is to discuss issues that underpin DBMSs. The chapter starts by presenting the three-level ANSI-SPARC architecture for a database system and examines each level together with the different database languages that can be associated with it. The chapter introduces different data models and the importance of conceptual modeling. It provides an introduction to the functions of a DBMS (that are developed in later chapters in the book) and examines the software components. The chapter ends with a discussion of common multi-user DBMS architectures and structures.

Points to emphasize

- The purpose and origin of the three-level database architecture.
- The contents of the external, conceptual, and internal levels.
- The purpose of the external/conceptual and the conceptual/internal mappings.
- The meaning of logical and physical data independence.
- The distinction between a data definition language (DDL) and a data manipulation language (DML).
- A classification of data models.
- The purpose and importance of conceptual modeling.
- The typical functions and services a DBMS should provide.
- The function and importance of the system catalog.
- The software components of a DBMS.
- The meaning of client–server architecture and the advantages of this type of architecture for a DBMS.
- The function and uses of Transaction Processing (TP) Monitors.

Possible difficulties

For a first reading, parts of this material will be too detailed, such as the software components of a DBMS, client-server architecture, and TP Monitors, and consideration has to be given to what needs to be covered in the course. Students can confuse the ANSI-SPARC 3-level architecture with the software component architecture for a DBMS. Some may also subsequently confuse DBMS services and functions with the software components without necessarily making any connection between them.

Teaching hints

When explaining the three-level architecture, consider describing a retrieval operation to illustrate the need for the DBMS to examine the mapping between the different levels in order to access required data.

Students could be asked to define a conceptual view and some external views (using the example given) on available DBMSs to show how actual systems handle these definitions.

Data independence can be illustrated by seeing the impact on external views of changes made at the conceptual level and the internal level. The students could also try this out on the examples previously defined. This again will demonstrate the limitations or otherwise of actual DBMSs.

It may be useful to ask the students to match up the appropriate software components to the DBMS functions.

Students should be required to re-read individual sections (possibly near the end of the course when other concepts have been covered).

Part 2

The Relational Model and Languages

CHAPTER 3

The Relational Model

Aims

The aim of this chapter is to introduce the terminology and concepts of the relational model, which is now the dominant system for business applications.

Points to emphasize

- The origins of the relational model.
- The terminology of the relational model.
- How tables are used to represent data.
- The connection between mathematical relations and relations in the relational model.
- Properties of database relations.
- How to identify candidate, primary, alternate, and foreign keys.
- The meaning of entity integrity and referential integrity.
- The purpose and advantages of views in relational systems.

Possible difficulties

By using an illustrative example (such as that given in Figure 3.1), many of the concepts and properties of the relational model can be easily explained. The concept of views may appear quite abstract for the student at this point, but many grasp the concept once the SQL CREATE VIEW statement has been covered in Chapter 6.

Teaching hints

Ensure that the students understand the basic concepts of the relational model before continuing with other chapters. To this end, going through the properties of relations in Section 3.2.4 can be very useful. Point out, though, that the perception of a relation as a table applies only to the logical structure of the database, not the physical structure.

Emphasize the importance of entity and referential integrity.

Give a brief introduction to views, but come back to them again in Chapter 6 – SQL: Data Definition.

On completion of this chapter, the material in Appendix D – When is a DBMS Relational? could be covered, although these rules may appear slightly academic.

CHAPTER 4

Relational Algebra and Relational Calculus

Aims

The aim of this chapter is to introduce the theoretical languages of the relational model.

Points to emphasize

- The meaning of the term 'relational completeness'.
- How to form queries in relational algebra.
- How to form queries in tuple relational calculus.
- How to form queries in domain relational calculus.
- The categories of relational Data Manipulation Languages (DMLs).

Possible difficulties

The material on relational algebra and relational calculus may be difficult for certain students to grasp, particularly on a first reading. It may be that this material is not important for your course and can be omitted.

Teaching hints

Usually the unary operations, Selection and Projection, and the set operations (Union, Set difference, and Intersection) are easily understood. Some time should be spent ensuring that the Cartesian product operation is understood before moving on to the Join operations. Usually the best way to explain relational algebra is by illustration. Emphasize the procedural nature of the relational algebra. Use the Exercises at the end of this chapter and the additional questions in Part IV of this Instructor's Guide to give students practice of writing relational algebra operations.

In contrast, the non-procedural nature of the relational calculus will seem obscure to the students. Again, illustration is usually the best way to explain both tuple and domain relational calculus. It is worthwhile coming back to relational calculus once SQL and QBE have been covered, and the language will appear more meaningful to the student. Again, use the Exercises at the end of this chapter and the additional questions in Part IV of this Instructor's Guide to give students practice of writing relational calculus operations.

Note that relational algebra is needed to understand query processing in Chapter 21 and fragmentation in Chapter 22 on distributed databases. If preferred, you could delay coverage of the algebra until then. However, the comparative aspects of the algebra and calculus act as a useful precursor for the study of SQL in Chapters 5, 6, and 7, although not essential.

CHAPTER 5

SQL: Data Manipulation

Aims

The aim of this chapter is to introduce the data manipulation facilities in the SQL standard. The chapter is presented as a tutorial, giving a series of worked examples that demonstrate the main concepts of SQL. In particular, it concentrates on the data manipulation statements: SELECT, INSERT, UPDATE, and DELETE.

Points to emphasize

- The purpose and importance of the Structured Query Language (SQL).
- The history and development of SQL.
- How to write an SQL command.
- How to retrieve data from the database using the SELECT statement.
- How to build SQL statements that:
  - use the WHERE clause to retrieve rows that satisfy various conditions;
  - sort query results using ORDER BY;
  - use the aggregate functions of SQL;
  - group data using GROUP BY;
  - use subqueries;
  - join tables together;
  - perform set operations (UNION, INTERSECT, EXCEPT).
- How to perform database updates using INSERT, UPDATE, and DELETE.

Possible difficulties

The SQL language is now very complex. In particular, the SELECT statement has many variations. If your time on SQL is limited, cover the main parts of the SELECT statement and omit many of the complexities.

A common mistake is for students to try to use an aggregate function in the WHERE clause and when using the GROUP BY statement to not appreciate that each item in the SELECT list must be single-valued per group.

Due to the wide variations in SQL dialects, it is worthwhile checking that the SQL of your RDBMS performs the statements covered in this chapter. For example, your RDBMS's dialect of SQL may not support outer joins. Let the students know in advance which features are not supported. Also point out that the order of rows returned in queries that do not use an ORDER BY clause may not be different between the examples in the

chapter and your RDBMS and that this is not an error but simply an implementation-defined feature of the system.

Teaching hints

Perhaps the best way to learn SQL is by illustration and practice. The transparencies for this chapter contain all the worked examples. You will most likely want to omit some of these due to time considerations. It would be useful to give the students a handout containing the base tables for the case study (Figure 3.3). This will make it easier for them to see how the result of each SELECT statement has been derived.

We demonstrate how to use the facilities of SQL using simple examples presented as a tutorial. As such, students can work through this chapter with Microsoft Office Access or an equivalent RDBMS to gain 'hands-on' experience of SQL. The SQL operations access the tables of the *DreamHome* case study. Therefore, before using this chapter, create these tables and populate them with the data shown in Figure 3.3.

In addition, use the Exercises at the end of this chapter and the additional questions in Part IV of this Instructor's Guide to give students practice of writing SQL statements.

An alternative or complementary approach to teach SQL is to use QBE to visually depict the query and then show the resulting SQL statement. This approach is used in Chapter 7 and you may prefer to cover Chapter 7 first and then return to Chapters 5 and 6.

CHAPTER 6

SQL: Data Definition

Aims

The aim of this chapter is to introduce the main data definition facilities in the SQL standard. As with the previous chapter, the information is presented as a worked tutorial. This chapter covers the SQL data types, examines the SQL Integrity Enhancement Feature (IEF), and shows the main data definition statements. This chapter also looks at views and the access control statements GRANT and REVOKE.

Points to emphasize

- The data types supported by the SQL standard.
- The purpose of the Integrity Enhancement Feature of SQL.
- How to define integrity constraints using SQL including:
  - required data;
  - domain constraints;
  - entity integrity;
  - referential integrity;
  - general constraints.
- How to use the Integrity Enhancement Feature in the CREATE and ALTER TABLE statements.
- The purpose of views.
- How to create and delete views using SQL.
- How the DBMS performs operations on views.
- Under what conditions views are updatable.
- The advantages and disadvantages of views.
- How the ISO transaction model works.
- How to use the GRANT and REVOKE statements as a level of security.

Possible difficulties

As with the last chapter, due to the wide variations in SQL dialects, it is worthwhile checking that the SQL of your RDBMS performs the statements covered in the chapter. For example, your RDBMS's dialect of SQL may not support the same data types as the SQL standard. Further, it may not support some of the advanced data definition facilities, such as the FOREIGN KEY clause or the CONSTRAINT clause.

Teaching hints

Again, perhaps the best way to learn SQL is by illustration and practice. The transparencies for this chapter contain all the worked examples, some of which you may want to omit due to time considerations. Again, use the Exercises at the end of this chapter and the additional questions in Part IV of this Instructor's Guide to give students practice of writing SQL statements.

The CREATE/DROP INDEX statements are not part of the SQL standard. As a result, you may wish not to cover this material. If you do, it would be worthwhile stressing this fact.

CHAPTER 7

Query-By-Example

Aims

To provide an overview of the major features of the query-by-example (QBE) facility of the Microsoft Office Access database management system (DBMS). To describe how QBE represents a visual approach for accessing information in a database through the use of query templates. To demonstrate how to use QBE by entering example values directly into the query template to represent what the access to the database is to achieve, such as the answer to a query.

Points to emphasize

- The main features of Query-By-Example (QBE).
- The types of query provided by the Microsoft Office Access DBMS QBE facility.
- How to use QBE to build queries to select fields and records.
- How to use QBE to target single or multiple tables.
- How to perform calculations using QBE.
- How to use advanced QBE facilities including parameter, find matched, find unmatched, crosstab, and autolookup queries.
- How to use QBE action queries to change the content of tables.

Possible difficulties

This chapter attempts to describe and demonstrate the power of QBE. However, to really appreciate the range of possible operations available using the QBE facility requires 'hands-on' experience in a laboratory environment. Therefore, it is important that students are using a DBMS that provides a QBE or equivalent facility.

Although this chapter is written in a tutorial-like style, it is not intended to be a Microsoft Access software manual. This chapter cannot possibly compete with a software manual but simply aims to introduce the range of facilities available using QBE to students with some prior knowledge of the DBMS software.

Teaching hints

In this chapter, we aim to give a general overview of the QBE facility provided by Microsoft Access. This is achieved using simple examples to demonstrate a range of possible operations using QBE such as inserting and deleting records, modifying the values of fields, or creating new fields. The screen dumps shown are available as transparencies. We do appreciate that the screen dumps will most likely be superseded by new versions of Microsoft Access and also that some readers of the accompanying textbook will be using different

DBMS software. However, the transparencies will be useful to introduce QBE. In addition, we recommend that (if possible) QBE is best demonstrated in action, using a PC equipped with an overhead projector. With this approach students very quickly see the potential of the QBE facility.

We demonstrate how to use the facilities of QBE using simple examples presented as a tutorial. As such, students can work through this chapter with Microsoft Access or an equivalent DBMS to gain 'hands-on' experience of QBE. The QBE operations access the tables of the *DreamHome* case study. Therefore, before using this chapter, create these tables and populate them with the data shown in Figure 3.3.

CHAPTER 8

Commercial RDBMSs: Office Access and Oracle

Aims

The aim of this chapter is to provide an overview of both the Microsoft Office Access and Oracle RDBMSs.

Points to emphasize

- About Microsoft Office Access:
  - the DBMS architecture;
  - how to create base tables and relationships;
  - how to create general constraints;
  - how to use forms and reports;
  - how to use macros.
- About Oracle9*i*:
  - the DBMS architecture;
  - how to create base tables and relationships;
  - how to create general constraints;
  - how to use PL/SQL;
  - how to create and use stored procedures and functions;
  - how to create and use triggers;
  - support for grid computing.

Possible difficulties

This chapter attempts to describe and demonstrate the power of two of the most popular RDBMSs on the market: Microsoft Office Access and Oracle. However, to really appreciate these systems requires 'hands-on' experience in a laboratory environment. Therefore, it would be extremely useful if the students had access to one (or both) of these systems for experimentation.

Although this chapter is written in a tutorial-like style, it is not intended to be a Microsoft Office Access or Oracle software manual. This chapter cannot possibly compete with a software manual but simply aims to introduce the range of facilities available using these systems to students with some prior knowledge of the DBMS software. The chapter also aims to show the difference between a PC-based DBMS and a large multi-user DBMS. The web site for this book provides a lab manual for both Microsoft Office Access and Oracle, and you may wish to use these to supplement this chapter.

Teaching hints

In this chapter we give a general overview of the facilities provided by Microsoft Office Access and Oracle. This is achieved using simple examples to demonstrate some of the functionality. We recommend that (if possible) the systems are best demonstrated in action, using a PC equipped with an overhead projector.

If covering both systems, highlight the significant difference between the administration and management of database structures of Office Access and Oracle – an Access database essentially consists of a single file and a new database can be easily created by a user. On the other hand, the administration and management of an Oracle database is a complex task, and in a large organization requires a dedicated team to maintain.

Part 3

Database Analysis and Design Techniques

CHAPTER 9

Database Planning, Design, and Administration

Aims

To discuss the purpose of the Information Systems (IS) lifecycle and how this cycle relates to the database system development lifecycle. To describe the activities associated with each stage of the database system development lifecycle. To focus on and discuss in more detail, particular stages in the lifecycle such as database design and DBMS selection. To describe and contrast the activities associated with data administration (DA) and database administration (DBA) and to discuss the relationship of these functions with the database system development lifecycle.

Points to emphasize

- The main components of an information system.
- The main stages of the database system development lifecycle.
- The main phases of database design: conceptual, logical, and physical design.
- The benefits of Computer-Aided Software Engineering (CASE) tools.
- The types of criteria used to evaluate a DBMS.
- How to evaluate and select a DBMS.
- The distinction between data administration and database administration.
- The purpose and tasks associated with data administration and database administration.

Possible difficulties

Students may have some difficulties distinguishing between the characteristics and purpose of the three phases of database design, namely conceptual, logical, and physical database design.

Also students may have difficulties appreciating the difference between local and global data models and the different approaches that can be used to create a global model.

The discussion on DBMS selection may prove difficult for students new to the subject as they will not be familiar with the criteria used in the selection. However, even new students should gain some appreciation of the process.

Teaching hints

Case studies can be used to reinforce and clarify the various stages of the database system development lifecycle. For example, students can undertake database planning, system definition, and requirements collection for a particular case study. The results would then form the basis for later work.

Students can be introduced to the concepts of views by using a complex case study that includes several views of the same situation such as the *DreamHome* or *Wellmeadows Hospital* case studies.

If students have access to appropriate CASE tools, these can be used for the activities previously mentioned. They could also be used after the activities have been undertaken manually, to highlight the difference in working.

CHAPTER 10

Fact-Finding Techniques

Aims

To discuss when a database developer might use fact-finding techniques and what types of facts should be captured. To show how these facts are used to generate the main types of documentation used throughout the database system development lifecycle. To describe the most commonly used fact-finding techniques and identify the advantages and disadvantages of each. Finally, to demonstrate how some of these techniques may be used during the early stages of the database system development lifecycle.

Points to emphasize

- When fact-finding techniques are used in the database system development lifecycle.
- The types of facts collected in each stage of the database system development lifecycle.
- The types of documentation produced in each stage of the database system development lifecycle.
- The most commonly used fact-finding techniques.
- How to use each fact-finding technique and the advantages and disadvantages of each.
- About a property rental company called *DreamHome.*
- How to apply fact-finding techniques to the early stages of the database system development life-cycle.

Possible difficulties

If the database course is delivered early in the student's education, some of the research techniques may be easy for the student to understand but difficult to put into practice, particularly their ability to interview and develop appropriately structured questionnaires.

Teaching hints

Case studies can be used to reinforce and clarify this chapter. For example, students can undertake database planning, system definition, and requirements collection for a particular case study. The results would then form the basis for later work.

Students can be introduced to the concepts by using the case studies given in Appendix B, which become increasingly more complex. In many ways, practice is the best method for the students to learn and become comfortable with these concepts.

CHAPTER 11

Entity–Relationship Modeling

Aims

To introduce the Entity–Relationship (ER) model, a high-level conceptual data model developed by Chen (1976) and to discuss the usefulness of this modeling technique in database design. To describe the basic concepts associated with the ER model and to illustrate how these concepts can be used to describe the structure of a database and the associated retrieval and update transactions on the database. To identify potential problems associated with the development of an ER model called connection traps.

Points to emphasize

- How to use Entity–Relationship (ER) modeling in database design.
- The basic concepts associated with the Entity–Relationship (ER) model, namely entities, relationships, and attributes.
- A diagrammatic technique for displaying an ER model using the Unified Modeling Language (UML).
- How to identify and resolve problems with ER models called connection traps.

Possible difficulties

Students are sometimes confused by the fact that there may be a range of possible ER models that represent the same situation. This illustrates the subjective nature of the interpretation of user requirement specifications. Building an ER model from a textual description allows students to see the inherent difficulties in formalizing, often vaguely written specifications.

Teaching hints

It is important that students fully understand the purpose and benefits of using a high-level conceptual data model such as the ER model in designing a database. With this aim in mind, before introducing the concepts associated with the ER model, it is very helpful for students to see a possible end product of ER modeling. Even though they may not fully understand what they are looking at, it allows students to 'see' where they are heading. As with the learning of any practical skill, expertise in ER modeling is achieved through providing students with numerous worked examples. It is also useful for students to work in groups when creating an ER model. This allows students first-hand experience in understanding the problems associated with the interpretation of user requirement specifications and in the building of a conceptual data model that has the consensus of the group.

We recommend that the students start with simple examples of specifications from which they have to derive ER diagrams and more complexity is gradually introduced. We further recommend that the specifications are not too complex at this stage – more complex specifications should be given once the methodology for conceptual and logical database design has been covered in Chapters 15 and 16.

An alternative approach to learning about ER modeling is to provide students with various examples of ER models and request that they interpret what each model represents.

CHAPTER 12

Enhanced Entity–Relationship Modeling

Aims

To recognize the inherent problems associated with representing complex applications using the concepts of the ER model. To describe three of the most important and useful additional concepts that have been added to the original ER model resulting in development of the Enhanced Entity–Relationship (EER) model, namely specialization/generalization, aggregation, and composition.

Points to emphasize

- The limitations of the basic concepts of the Entity–Relationship (ER) model and the requirements to represent more complex applications using additional data modeling concepts.
- The most useful additional data modeling concepts of the Enhanced Entity–Relationship (EER) model called specialization/generalization, aggregation, and composition.
- A diagrammatic technique for displaying specialization/generalization, aggregation, and composition in an EER diagram using the Unified Modeling Language (UML).

Possible difficulties

Students may have some difficulty in recognizing when it is useful to use the additional concepts of the enhanced ER. The guidelines for use should be clearly given to clarify the student's understanding of how these concepts differ and consequently when it is appropriate to use these constructs. It may be helpful to ask students to construct both ER and EER models based on a users requirements specification and to discuss the benefits or otherwise of using the concepts of the enhanced model.

Teaching hints

As with the previous chapter, expertise in EER modeling is achieved through providing students with numerous worked examples. It is also useful for students to work in groups when creating an EER model.

An alternative approach to learning about EER modeling is to provide students with various examples of EER models and request that they interpret what each model represents.

CHAPTER 13

Normalization

Aims

To discuss how the process of normalization supports the database designer in developing a logical data model that is an accurate representation of the data requirements of an enterprise. To illustrate the problems that may arise in poorly designed relations. To describe the concept of functional dependency and demonstrate how this concept is applied within the technique of normalization. To illustrate the process of normalization using worked examples.

Points to emphasize

- The purpose of normalization.
- How normalization can be used when designing a relational database.
- The potential problems associated with redundant data in base relations.
- The concept of functional dependency, which describes the relationship between attributes.
- The characteristics of functional dependencies used in normalization.
- How to identify functional dependencies for a given relation.
- How functional dependencies identify the primary key for a relation.
- How to undertake the process of normalization.
- How normalization uses functional dependencies to group attributes into relations that are in a known normal form.
- How to identify the most commonly used normal forms, namely first (1NF), second (2NF), and third (3NF) normal forms.
- The problems associated with relations that break the rules for 1NF, 2NF, or 3NF.
- How to represent attributes shown on a form as 3NF relations using normalization.

Possible difficulties

Students often have problems understanding the purpose and process of normalization within the database design lifecycle. This may be due to the fact that to some extent the process of normalization 'formalizes' what may appear to be a 'common sense' approach to good database design.

Students also often have problems understanding the purpose and properties of later normal forms such as BCNF, 4NF, and 5NF. This is partly caused by the fact that violation of later normal forms occurs infrequently and examples of violations often appear obscure and contrived to the student.

Teaching hints

It is critical that students fully understand the concept of functional dependencies between attributes before describing the process of normalization. Achieving this understanding, often requires using numerous worked examples. It is also important to discuss the purpose of normalization and where it is applied within the database design lifecycle before describing the technique. It should be stressed that normalization can be used either as a bottom-up approach to database design or as a validation technique that simply provides database designers with a formal framework for assessing the appropriateness of the structure of each relation. (Note that the methodology described in the accompanying textbook uses normalization as a validation technique.) The importance of creating appropriately structured relations is achieved through the discussion on possible update anomalies. Students quickly appreciate the benefits of good database design by comparing good and badly composed relations and illustrating the types of anomalies that can be avoided by good design.

It is obviously better to introduce the process of normalization using simple worked examples. However, when using simple examples, students may sometimes fail to appreciate the usefulness of this technique. Therefore, it is important to stress that the technique becomes invaluable in more complex situations where normalization helps the database designer gain a greater understanding of the attributes and the relationships between the attributes.

CHAPTER 14

Advanced Normalization

Aims

The aim of this chapter is to examine the more advanced aspects of normalization and, in particular, Boyce–Codd Normal Form (BCNF), Fourth Normal Form (4NF), and Fifth Normal Form (5NF).

Points to emphasize

- How inference rules can identify a set of functional dependencies for a relation.
- How inference rules called Armstrong's axioms can identify a *minimal* set of useful functional dependencies from a set of all functional dependencies for a relation.
- Normal forms that go beyond Third Normal Form (3NF), which includes Boyce–Codd Normal Form (BCNF), Fourth Normal Form (4NF), and Fifth Normal Form (5NF).
- How to identify Boyce–Codd Normal Form (BCNF).
- How to represent attributes shown on a report as BCNF relations using normalization.
- The concept of multi-valued dependencies and 4NF.
- The problems associated with relations that break the rules of 4NF.
- How to create 4NF relations from a relation which breaks the rules of 4NF.
- The concept of join dependency and 5NF.
- The problems associated with relations that break the rules of 5NF.
- How to create 5NF relations from a relation which breaks the rules of 5NF.

Possible difficulties

Students often have problems understanding the purpose and process of normalization within the database design lifecycle. This may be due to the fact that to some extent the process of normalization 'formalizes' what may appear to be a 'common sense' approach to good database design.

Students also often have problems understanding the purpose and properties of later normal forms such as BCNF, 4NF, and 5NF. This is partly caused by the fact that violation of later normal forms occurs infrequently and examples of violations often appear obscure and contrived to the student.

Teaching hints

It is critical that students fully understand the concept of functional dependencies between attributes before describing the process of normalization. Achieving this understanding, often requires using numerous worked examples. It is also important to discuss the purpose of normalization and where it is applied within the database design lifecycle before describing the technique. It should be stressed that normalization can be used

either as a bottom-up approach to database design or as a validation technique that simply provides database designers with a formal framework for assessing the appropriateness of the structure of each relation. (Note that the methodology described in the accompanying textbook uses normalization as a validation technique.) The importance of creating appropriately structured relations is achieved through the discussion on possible update anomalies. Students quickly appreciate the benefits of good database design by comparing good and badly composed relations and illustrating the types of anomalies that can be avoided by good design.

It is obviously better to introduce the process of normalization using simple worked examples. However, when using simple examples, students may sometimes fail to appreciate the usefulness of this technique. Therefore, it is important to stress that the technique becomes invaluable in more complex situations where normalization helps the database designer gain a greater understanding of the attributes and the relationships between the attributes.

Part 4

Methodology

CHAPTER 15

Methodology – Conceptual Database Design

Aims

This chapter introduces a methodology for database design and discusses how this methodology relates to the database system development lifecycle described in Chapter 9. It presents an overview of the main steps of the methodology for the three main phases of database design, namely: conceptual, logical, and physical design:

- **Conceptual database design** – to build the conceptual representation of the database, which includes identification of important entities, relationships, and attributes.
- **Logical database design** – to translate the conceptual representation to the logical structure of the database, which includes designing the relations.
- **Physical database design** – to decide how the logical structure is to be physically implemented (as tables) on the target database management system (DBMS).

The methodology is described in four chapters (Chapters 15–18). The focus of Chapter 15 is the methodology for conceptual database design and describes in detail the activities associated with building a conceptual data model.

Points to emphasize

- The purpose of a design methodology.
- Database design has three main phases: conceptual, logical, and physical design.
- How to decompose the scope of the design into specific views of the enterprise.
- How to use Entity–Relationship (ER) modeling to build a local conceptual data model based on the information given in a view of the enterprise.
- How to validate the resultant conceptual model to ensure it is a true and accurate representation of a view of the enterprise.
- How to document the process of conceptual database design.
- End-users play an integral role throughout the process of conceptual database design.

Possible difficulties

Students may view the methodology as being too procedural and may not be willing to repeat previous steps, where necessary.

Students are sometimes confused by the distinction between conceptual and logical data models. It is important to stress that a conceptual data model is entirely independent of *all* implementation details such as the target data model, DBMS software, hardware platform, or any other physical considerations. In contrast, the logical data model is influenced by the target data model for the database (for example, the relational data model).

Teaching hints

It is important that students fully understand how to apply the techniques of Entity–Relationship modeling described in Chapters 11 and 12 before being introduced to the part of the methodology described in Chapter 15.

Students generally are able to understand the steps involved in conceptual database design but find it much more difficult to apply the steps, particularly if the case study is large or complex. Start with small case studies involving just a couple of entities, attributes, and relationships. Build up the students' confidence by successively adding more complexity. Work through as many case studies as you can – students tend to understand the application of the methodology through practice. There are several case studies in Appendix B for the students to work through and these can be supplemented or replaced with your own case studies.

Database design is an iterative process, which has a starting point and an almost endless procession of refinements. It is important to stress to students that although the methodology is listed as a series of steps and appears sequential, it does not imply that it should be performed in this manner. It is likely that knowledge gained in one step may alter decisions made in a previous step. Similarly, we may find it useful to briefly look at a later step to help with an earlier step. The methodology should simply act as a framework to guide a database designer through the process of database design, effectively.

Stress the importance of identifying the transactions that have to be supported and checking that their conceptual data model supports these transactions. This is a common mistake that students (and practitioners) make, resulting in a system that doesn't support the requirements. Step 1.8 can be tedious but it is vital to the overall success of the system.

Throughout this methodology, it is important to emphasize to students that users play a critical role in continually reviewing and validating the data model and the supporting documentation.

Note that Appendix G presents a summary of the methodology, for those students who are already familiar with database design and simply require an overview of the main steps.

CHAPTER 16

Methodology – Logical Database Design

Aims

The aim of this chapter is to present a step-by-step methodology for logical database design for the relational model. It shows how to map a conceptual data model to a logical data model and how to validate it against the required transactions using the technique of normalization. For database systems with multiple user views, this chapter shows how to merge the resulting data models together into a global data model that represents all the views of the part of the enterprise being modeled.

Points to emphasize

- How to derive a set of relations from a local conceptual data model.
- How to validate these relations using the technique of normalization.
- How to validate a local logical data model to ensure it supports the required user transactions.
- How to merge local logical data models based on one or more user views into a global logical data model that represents all user views.
- How to ensure that the final logical data model is a true and accurate representation of the enterprise.

Possible difficulties

Students are sometimes confused by the distinction between conceptual and logical data models. It should be emphasized that once a model is altered to meet the requirements of the target data model (for example, relational model), it is more correct to refer to the model as being a logical model.

Teaching hints

It is important that students fully understand how to apply the techniques of Entity–Relationship (ER) modeling (Chapters 11 and 12) and normalization (Chapters 13 and 14) before being introduced to the part of the methodology described in Chapter 16.

If desired a simplified version of the methodology can be taught to students by introducing only Step 1 and Step 2.1–2.5 (i.e. exclude Step 2.6 Merge logical data models into global model); in other words, only have one user view. Introducing the students to Step 2.6 is only necessary where there are multiple views of the database. You may also want to further simplify the mapping by excluding specialization/generalization.

A useful approach to using the methodology is to provide students with various case studies that become increasingly complex. Simpler case studies may only have a single view and therefore it is not necessary to

proceed to the Step 2.6. For example, the case studies described in Appendix B provide alternative examples for working through the database design methodology.

Note that Appendix G presents a summary of the methodology, for those students who are already familiar with database design and simply require an overview of the main steps.

CHAPTER 17

Methodology – Physical Database Design for Relational Databases

Aims

The aim of this chapter is to present a step-by-step methodology for physical database design for relational DBMSs. It shows how to translate the logical data model developed during logical database design into a physical design for a relational system. The methodology addresses the performance of the resulting implementation by providing guidelines for choosing file organizations and storage structures.

Points to emphasize

- The purpose of physical database design.
- How to map the logical database design to a physical database design.
- How to design base relations for the target DBMS.
- How to design general constraints for the target DBMS.
- How to select appropriate file organizations based on analysis of transactions.
- When to use secondary indexes to improve performance.
- How to estimate the size of the database.
- How to design user views.
- How to design security mechanisms to satisfy user requirements.

Possible difficulties

Students will need to have undertaken sufficient practical work on a specific DBMS to be able to design such things as referential integrity constraints and general constraints, or else have the practical sessions linking into, or following on, from the lectures.

Students may view the methodology as being too procedural and may not be willing to repeat previous steps, where necessary. Also, it is important that students recognize that we must adapt the physical database design methodology to the target DBMS. Some steps are not applicable to some database systems. For example, if the target DBMS is Microsoft Office Access then Step 4.2 of the methodology is not applicable, as this step involves selecting an appropriate file organization for database tables, which is not available to users of the Microsoft Access DBMS.

In a teaching environment, it may be difficult to allow students access to a 'realistic' DBMS that contains a substantial number of tables and records. However, this is important if students are to gain first-hand experience of the benefits of implementing a correct and efficient system.

Teaching hints

It is important that students fully understand how to create a logical data model before moving to the third phase of the database design methodology. Also, students should be encouraged to fully investigate the functionality of their target DBMS before they attempt to translate the logical data model into a physical design and ultimately into an implementation.

The physical design phase considers the storage structures and access methods required for efficient access to the database on secondary storage. It is helpful for students to have a good background in the file organization and data structures concepts covered in Appendix C before covering the material in Chapter 17 on physical database design. This background ideally will have been obtained from a prior course. If this is not possible, then the material in Appendix C can be presented near the beginning of the database course, immediately following Chapter 1.

Having the students use the case studies in Appendix B should reinforce the methodology for physical database design.

Note that Appendix G presents a summary of the methodology, for those students who are already familiar with database design and simply require an overview of the main steps.

CHAPTER 18

Methodology – Monitoring and Tuning the Operational System

Aims

The aim of this chapter is to describe and illustrate by example the final two steps of the physical database design methodology for relational databases. The chapter provides guidelines for determining when to denormalize the logical data model and introduce redundancy. The chapter discusses the importance of continuing to tune the operational system. It discusses how to measure efficiency and how system resources can impact performance.

Points to emphasize

- The meaning of denormalization.
- When to denormalize to improve performance.
- The importance of monitoring and tuning the operational system.
- How to measure efficiency.
- How system resources affect performance.

Possible difficulties

The major difficulty is leaving the students with the impression that every database design is subsequently denormalized for implementation. It should be emphasized that it is neither essential nor necessary to denormalize a design unless performance considerations are paramount. In considering denormalization, future developments must always be borne in mind, as altering the design for the current application could cause major problems for future applications.

Teaching hints

If the students have implemented the case studies in Appendix B, new requirements can be created and the students asked to modify their earlier design and implementation to cope with these new requirements. Additionally, ask the students to consider denormalization but ensure that they provide an adequate justification for any redundancy introduced.

Note that Appendix G presents a summary of the methodology, for those students who are already familiar with database design and simply require an overview of the main steps.

Part 5

Selected Database Issues

CHAPTER 19

Security

Aims

The aim of this chapter is to discuss database security, giving examples of the types of threats and the types of countermeasures that can be used to safeguard a system. The chapter provides an overview of risk analysis and discusses issues of data protection and privacy with regard to the handling of personal data. The chapter also identifies the security measures associated with DBMSs and the Web.

Points to emphasize

- The scope of database security.
- Why database security is a serious concern for an organization.
- The types of threat that can affect a database system.
- How to protect a computer system using computer-based controls.
- The security measures provided by Microsoft Office Access and Oracle DBMSs.
- Approaches for securing a DBMS on the Web.

Possible difficulties

Computer security is a large and complex area. Chapter 19 attempts to provide an overview of the important issues and in particular those related to database security.

There is a close relationship between database integrity and security; however, integrity is not included in this chapter as it is described in earlier chapters of the book. In particular, integrity is described in detail in the chapter on the relational model (Chapter 3) and SQL: Data Definition (Chapter 6).

Teaching hints

In discussing threats, students generally are aware of the more obvious but less likely types such as fire and flood. The case studies can be used to focus on potential problems with respect to particular systems and can serve to highlight differences between systems.

Countermeasures should be discussed generally. The case studies can again be used here for the students to determine appropriate safeguards against perceived threats. Depending on the focus of the course, the emphasis could be on computer-based controls or non-computer-based controls.

Section 19.5 on DBMSs and Web security may be left until the Web is being discussed in Chapter 29.

CHAPTER 20

Transaction Management

Aims

The aim of this chapter is to introduce three of the functions that a DBMS should provide namely, transaction management, concurrency control, and recovery control. These functions are intended to ensure that the database is reliable and remains in a consistent state, when multiple users are accessing the database and in the presence of failures of both hardware and software components.

Points to emphasize

- The purpose of concurrency control.
- The purpose of database recovery.
- The function and importance of transactions.
- The properties of a transaction.
- The meaning of serializability and how it applies to concurrency control.
- How locks can be used to ensure serializability.
- How the two-phase locking (2PL) protocol works.
- The meaning of deadlock and how it can be resolved.
- How timestamps can be used to ensure serializability.
- How optimistic concurrency control techniques work.
- How different levels of locking may affect concurrency.
- Some causes of database failure.
- The purpose of the transaction log file.
- The purpose of checkpoints during transaction logging.
- How to recover following database failure.
- Alternative models for long-duration transactions.
- How Oracle handles concurrency control and recovery.

Possible difficulties

The transaction is fundamental to an understanding of concurrency and recovery control. However, some students are uncertain about what constitutes a transaction. There are a couple of examples in Section 20.1, but you may wish to supplement these to reinforce the learning.

There is a significant amount of material in this chapter. Depending on the level of detail you wish to go into on this area, the sections on serializability (20.2.2), multiversion timestamp ordering (20.2.6), and hierarchy of

granularity (20.2.8), and the more in-depth look at the various concurrency and recovery protocols could be omitted. Similarly, the advanced transaction models discussed in Section 20.4 could be omitted or left until Chapter 25, when advanced database systems are introduced.

Teaching hints

Make sure the students understand what a transaction is and how it is perceived from the user and DBMS perspectives.

Work through the overheads on the three concurrency problems: the **lost update problem**, the **uncommitted dependency problem**, and the **inconsistent analysis problem**. Then, once you have introduced 2PL, go through how 2PL prevents these problems. It may be useful to have these six overheads as handouts to help this process.

It may be useful to go through how the protocols for timestamping and optimistic techniques would prevent these problems for occurring, similar to Examples 20.6–20.8.

Make sure the students understand that serializability is not itself a protocol for concurrency control and that they appreciate the difference between conflict serializability and view serializability.

When covering the material on recovery, it may be useful to present another example similar to Example 20.11/20.12 to reinforce the learning.

You may not wish to go into the various recovery techniques in Section 20.3.4 in too much detail, but an overview of the differences may be beneficial.

Students may find it useful to examine what concurrency and recovery protocols are implemented by the DBMSs they are using. For example, some DBMSs allow users to examine the transaction log file, and a handout showing a sample portion of this file may be helpful to aid understanding.

If the course is not using Oracle, Section 20.5 can be omitted.

CHAPTER 21

Query Processing

Aims

The aim of this chapter is to examine query processing and query optimization. The chapter considers the two main techniques for query optimization: the use of heuristic rules that order the operations in a query, and the other technique that compares different strategies based on their relative costs, and selects the one that minimizes resource usage. It also briefly examines how Oracle performs query optimization.

Points to emphasize

- The objectives of query processing and optimization.
- Static versus dynamic query optimization.
- How a query is decomposed and semantically analyzed.
- How to create a relational algebra tree to represent a query.
- The rules of equivalence for the relational algebra operations.
- How to apply heuristic transformation rules to improve the efficiency of a query.
- The types of database statistics required to estimate the cost of operations.
- The different strategies for implementing the relational algebra operations.
- How to evaluate the cost and size of the relational algebra operations.
- How pipelining can be used to improve the efficiency of queries.
- The difference between materialization and pipelining.
- The advantages of left-deep trees.
- Approaches for finding the optimal execution strategy.
- How Oracle handles query optimization.

Possible difficulties

The students may experience significant difficulties with the technical details of query processing and optimization. Certainly, they will need to have covered the section on relational algebra in Chapter 4. Even if this has been covered, it may be worthwhile having a quick review of this prior to covering the material in this chapter.

Give the students plenty of opportunity to understand the aims and objectives of query processing, perhaps using Example 21.1 as a focus, or supplementing this example with one or two other relevant examples.

Similarly, on a first reading at least, it may be helpful if the students do not get too embroiled in understanding the costs provided for the relational algebra operations, but instead concentrate on what each approach to optimizing the relational algebra operation is trying to achieve.

Teaching hints

Students may find the heuristical approach to query optimization easier to understand, and you may wish to concentrate more effort here, at least in the early stages, until the students understand what query processing is trying to achieve. Going through a number of different examples in a tutorial environment of drawing (canonical) relational algebra trees and applying the heuristics to produce a reduced tree is potentially the best way for the students to understand this approach. Use the Exercises at the end of this chapter and the additional questions in Part IV of this Instructor's Guide to give students practice of producing relational algebra trees.

Similarly, working through a number of different examples using the cost estimation approach generally helps aid learning. You should consider whether you want students to know how to derive the cost estimates for each of the relational algebra operations or whether it is sufficient for them to know that such estimates exist and concentrate instead on applying the estimates to determine the cost of a given strategy.

Spend some time on Section 21.5 Enumeration of Alternative Execution Strategies so that students understand the complexity of trying to find the optimal execution strategy. You may even want to introduce this near the start of the coverage of Query Optimization to highlight the problem faced.

Some DBMSs may provide a facility to examine the execution strategy for a query. If this is available, it would be worthwhile letting the students experiment with various alternative formulations of a query, examining the strategy produced by the DBMS, and comparing it with the strategies produced by the application of the heuristic rules and cost estimation.

Again, if the course is not using Oracle, Section 21.6 can be omitted (although it would still be beneficial to cover the subsection on histograms).

Part 6

Distributed DBMSs and Replication

CHAPTER 22

Distributed DBMSs – Concepts and Design

Aims

Distributed database management system (DDBMS) technology is one of the recent major developments in database systems. The previous chapters of this book concentrate on centralized database systems, that is, systems with a single logical database located at one site under the control of a single DBMS.

The aim of this chapter is to introduce the concepts and problems of DDBMSs where users can not only access the database at their own site but also access data stored at remote sites. There are claims that in the next few years centralized database systems will be an 'antique curiosity' as most organizations move towards distributed database systems.

Points to emphasize

- The need for distributed databases.
- The differences between distributed database systems, distributed processing, and parallel database systems.
- The advantages and disadvantages of distributed DBMSs.
- The problems of heterogeneity in a distributed DBMS.
- Basic networking concepts.
- The functions that should be provided by a distributed DBMS.
- An architecture for a distributed DBMS.
- The main issues associated with distributed database design, namely fragmentation, replication, and allocation.
- How fragmentation should be carried out.
- The importance of allocation and replication in distributed databases.
- The levels of transparency that should be provided by a distributed DBMS.
- Comparison criteria for distributed DBMSs.

Possible difficulties

There is a significant amount of material in this chapter. For a first-level course in database management, you may have to significantly curtail what you cover. For an understanding of fragmentation, you will need to have covered the section on relational algebra in Chapter 4. Even if this has been covered, it may be worthwhile having a quick review of this material prior to the discussion of fragmentation. You could always use SQL instead of the relational algebra operations if you prefer.

Due to space considerations, there is only a brief overview of networking. For the most part, this overview is sufficient for an understanding of the material covered in this chapter. However, for a more advanced level course, you may wish to supplement this material with additional readings on networking concepts (see the bibliography and further readings for this chapter for some suggestions).

Teaching hints

Make sure the students understand the difference between the distributed DBMS and distributed processing before embarking on the rest of the chapter.

Coverage of the advantages/disadvantages is well worth a revisit on completion of the material, when the student has acquired a deeper understanding of the concepts and the discussion will be less abstract.

Use the Exercises at the end of this chapter and the additional questions in Part IV of this Instructor's Guide to give students practice of distributed database design.

It is worthwhile stressing to the students that not all facilities are provided by commercial implementations. For example, few systems implement fragmentation as originally conceived, and replication with all copies concurrently updated.

The criteria for distributed systems in Section 22.6, as proposed by Date, are sometimes regarded as academic but may still be a useful yardstick for comparison. However, many are ideals that no DDBMS will comply with for the foreseeable future.

Allow the students to form their own opinion as to the advantages and disadvantages of the distributed DBMSs – beware of material that is significantly biased against such systems.

CHAPTER 23

Distributed DBMSs – Advanced Concepts

Aims

The aim of this chapter is to examine various advanced concepts associated with distributed DBMSs. In particular, it concentrates on the protocols associated with distributed transaction management, concurrency control, deadlock management, and database recovery. The chapter also examines the X/Open Distributed Transaction Processing (DTP) protocol and distributed query optimization. It also provides an overview of how Oracle handles data distribution.

Points to emphasize

- How data distribution affects the transaction management components.
- How centralized concurrency control techniques can be extended to handle data distribution.
- How to detect deadlock when multiple sites are involved.
- How to recover from database failure in a distributed environment:
  - two-phase commit (2PC);
  - three-phase commit (3PC).
- The difficulties of detecting and maintaining integrity in a distributed environment.
- About the X/Open DTP standard.
- About distributed query optimization.
- The importance of the Semijoin operation in distributed environments.
- How Oracle handles data distribution.

Possible difficulties

The material on distributed transaction management may be difficult for some students to grasp, particularly if they do not fully understand the objectives of transaction management for centralized systems. If this is the case, review the objectives for centralized systems before proceeding. The same applies to query optimization. Similarly, the material on the X/Open DTP standard may appear abstract, and it may be worthwhile reviewing some examples of TP systems to make the objectives and discussion more concrete.

Teaching hints

It is important that students fully understand the aims of transaction management for a centralized database system and the ACID properties before undertaking a study of distributed transaction management. Many of the algorithms for distributed transaction management are best illustrated in a tutorial-style environment by going though examples, building from simple to more complex scenarios.

If the students have access to a distributed DBMS, it would be worthwhile going through the distributed functionality of this system, and examining the particular algorithms used.

For an understanding of query optimization, you will need to have covered the material in Chapter 21. If the course is not using Oracle, Section 23.7 can be omitted.

CHAPTER 24

Replication and Mobile Databases

Aims

The aim of this chapter is to discuss replication servers as an alternative to distributed DBMSs and to examine the issues associated with mobile databases. It also provides an overview of how Oracle handles replication.

Points to emphasize

- How a replicated database differs from a distributed database.
- The benefits of database replication.
- Examples of applications that use database replication.
- Basic components of a replication system.
- How synchronous replication differs from asynchronous replication.
- The main types of data ownership are master/slave, workflow, and update-anywhere.
- The functionality of a database replication server.
- Main implementation issues associated with database replication.
- How mobile computing supports the mobile worker.
- Functionality of a mobile DBMS.
- How Oracle DBMS supports database replication.

Possible difficulties

Some students may have difficulty appreciating the differences between distributed DBMSs and replication servers and you may need to take some time going through these differences.

Teaching hints

If students have access to a DBMS with replication facilities, it would be worthwhile going through the features provided by this system, and examining the methods for publish and subscribe.

Part 7

Object DBMSs

CHAPTER 25

Introduction to Object DBMSs

Aims

The preceding chapters of this book concentrate on the relational model and relational systems. The justification for this is that such systems are currently the predominant DBMS for traditional business database applications. However, relational systems are not without their failings and the object-oriented DBMS and the object-relational DBMS are major developments in the database systems area that attempt to overcome these failings.

The aim of this chapter is to act as an introduction to the following three chapters that discuss these systems in detail. The chapter first examines the types of advanced database applications that are emerging and discusses the weaknesses of the relational data model that make them unsuitable for these types of applications. The chapter then proceeds to discuss the main concepts of object orientation.

Points to emphasize

- The requirements for advanced database applications.
- Why relational DBMSs currently are not well suited to supporting advanced database applications.
- The concepts associated with object-orientation:
  - abstraction, encapsulation, and information hiding;
  - objects and attributes;
  - object identity;
  - methods and messages;
  - classes, subclasses, superclasses, and inheritance;
  - overloading;
  - polymorphism and dynamic binding.
- The problems associated with storing objects in a relational database.
- What constitutes the next generation of database systems.
- The basics of object-oriented analysis and design with UML.

Possible difficulties

Students may find the material on advanced database applications abstract, particularly if they have no prior experience of such applications. In this case, go through a sample application that they may be able to relate to in some detail, so that the students get a feel for how different such applications are from traditional business applications. With the widespread use of the Web, students may be able to relate more with the example given on interactive and dynamic Web sites.

If the students have no prior experience of object-orientation, the examples in Section 25.4.2 may seem rather obscure.

The material on object-oriented database design in Section 25.6 is also useful, although due to space considerations it is brief. It may prove very difficult for students to comprehend how to identify methods without additional teaching or even taking a course in object-oriented analysis. If you want to cover this in more detail, supplementary examples will prove helpful.

Teaching hints

When covering the material on advanced applications, it would be useful to expand on some of the areas and give more detailed examples, and to cite some experiences with object-oriented systems.

Telling the students that the relational model has weaknesses may be very eye-opening for them. Until now, they may have heard no criticisms uttered against the relational model. However, the so-called weaknesses have to be tempered with the wealth of benefits relational systems have afforded, and it is worthwhile spending some time on this. Care should be taken with using reference material that is very biased towards or against the relational data model. Instead, students should be allowed to form their own opinions on the advantages and disadvantages of this model.

If the students have no prior experience of object-orientation, Section 25.3 can be used as an introduction. Understanding Section 25.4.1 is important to appreciate the objectives of object-oriented DBMSs.

Use the Exercises at the end of this chapter and the additional questions in Part IV of this Instructor's Guide to give students practice of UML notations, particularly use cases.

CHAPTER 26

Object-Oriented DBMSs – Concepts

Aims

The aim of this chapter is to examine the object-oriented DBMS (OODBMS). The chapter first provides an introduction to object-oriented data models and persistent languages. It then discusses the difference between the two-level storage model used by conventional DBMSs and the single-level model used by OODBMSs, and how this affects data access. It also discusses the various approaches to providing persistence in programming languages and the different techniques for pointer swizzling, and then examines extended transaction models, version management, schema evolution, and OODBMS architectures. The chapter briefly shows how the methodology presented in Part 4 of the book may be extended for object-oriented databases.

Points to emphasize

- The framework for an object-oriented data model.
- The basics of the functional data model.
- The basics of persistent programming languages.
- The main points of the OODBMS Manifesto.
- The main strategies for developing an OODBMS.
- The difference between the two-level storage model used by conventional DBMSs and the single-level model used by OODBMSs.
- How pointer swizzling techniques work.
- The difference between how a conventional DBMS accesses a record and how an OODBMS accesses an object on secondary storage.
- The different schemes for providing persistence in programming languages.
- The advantages and disadvantages of orthogonal persistence.
- About various issues underlying OODBMSs, including extended transaction models, version management, schema evolution, OODBMS architecture, and benchmarking.
- The advantages and disadvantages of OODBMSs.

Possible difficulties

There is a significant amount of material in this chapter. For a first-level course in database management, you may elect to omit this material or significantly curtail what is covered. Sections 26.1 and 26.5 may act as a useful subset to cover.

Many of the concepts of OODBMSs may appear quite abstract without a deeper understanding of how such systems work in practice. Clearly, if you have an OODBMS in your environment, you can use it in practical sessions to support the lectures. Depending on the background of your students, it may be worthwhile giving them a programming project that aims to make objects persistent. One such project is described in Part IV of the Instructor's Guide.

Teaching hints

Coverage of the functional data model is useful and helps illustrate some of the features of object-oriented systems and, in particular, path expressions in the functional query language.

The issues presented in Section 26.4 are only a subset of the issues that exist with the OODBMS. You may wish to limit/extend these issues depending on the level of your course.

If the students have access to an OODBMS, many of the issues discussed in this chapter will be best illustrated by examining the features of this system. If the students have access to more than one OODBMS, you will be able to get the students to compare and contrast the features provided by each one.

Allow the students to form their own opinion as to the advantages and disadvantages of the OODBMS as compared the RDBMS – beware of material that is significantly biased towards one system.

CHAPTER 27

Object-Oriented DBMSs – Standards and Systems

Aims

The aim of this chapter is to examine the new object model proposed by the Object Data Management Group (ODMG), which has become a *de facto* standard for OODBMSs. The chapter also aims to demonstrate the functionality of one commercial OODBMS, namely ObjectStore.

Points to emphasize

- About the Object Management Group (OMG) and the Object Management Architecture (OMA).
- The main features of the Common Object Request Broker Architecture (CORBA).
- The main features of the other OMG standards including UML, MOF, XMI, CWM, and the Model-Driven Architecture (MDA).
- The main features of the new Object Data Management Group (ODMG) Object Database Standard:
  - Object Model;
  - Object Definition Language (ODL);
  - Object Query Language (OQL);
  - Object Interchange Format (OIF);
  - language bindings.
- The main features of ObjectStore, a commercial OODBMS:
  - the ObjectStore architecture;
  - data definition in ObjectStore;
  - data manipulation in ObjectStore.

Possible difficulties

If the students have no prior experience of object-orientation apart from the concepts covered in Chapter 25, Section 27.3 may be difficult for the students to comprehend. In this case, concentrate on segments of the sample code that relate to creating named roots and objects and then retrieving objects from the database. Similarly, the material on CORBA and the other OMG specifications in Section 27.1 could be omitted.

The material on the ODMG Object Model in Section 27.2 may prove to be too abstract unless you have access to an ODMG-compliant system. However, the material is useful for comparative purposes. Similarly, OQL may be quite abstract without the ability to try OQL queries out.

Teaching hints

Many students will find the ODMG object model easier to understand once they have studied Example 27.1. You may wish to reinforce the learning by providing another example in a tutorial. Similarly, the ODL will be easier to understand once the students work through Examples 27.2–27.6. Again, supplement these examples with additional ones in a tutorial.

If you have access to a different OODBMS, you may wish to omit Section 27.3 and provide other material related to your OODBMS. If you do not have access to any OODBMS, as mentioned above you may wish to just concentrate on segments of the sample code that relate to creating named roots and objects and then retrieving objects from the database.

When covering OQL it is useful to show the similarities with path expressions from the functional data model covered in the previous chapter. Use the Exercises at the end of this chapter and the additional questions in Part IV of this Instructor's Guide to give students practice of OQL.

If you do not have access to ObjectStore (although the single user system is currently freely downloadable from the ObjectStore web site), you may wish to omit Section 27.3 or curtail the details covered (for example, to simply illustrating data manipulation in Section 27.3.4).

CHAPTER 28

Object-Relational DBMSs

Aims

The aim of this chapter is to examine the object-relational DBMS, and provide a detailed overview of the object management features that have been added to the new release of the SQL standard, SQL:2003. The chapter also discusses how query processing and query optimization need to be extended to handle data type extensibility efficiently. To make the concepts more realistic, the chapter examines how Oracle has been extended to handle objects.

Points to emphasize

- How the relational model has been extended to support advanced database applications.
- The features proposed in the third-generation database system manifestos presented by CADF, and Darwen and Date.
- The extensions to the relational data model that have been introduced to Postgres.
- The object-oriented features proposed in the new SQL standard, SQL:2003, including:
  - row types;
  - user-defined types and user-defined routines;
  - polymorphism;
  - inheritance;
  - reference types and object identity;
  - collection types (ARRAYs, MULTISETs, SETs, and LISTs);
  - extensions to the SQL language to make it computationally complete;
  - triggers;
  - support for large objects – Binary Large Objects (BLOBs) and Character Large Objects (CLOBs);
  - recursion.
- Extensions required to relational query processing and query optimization to support advanced queries.
- Some object-oriented extensions to Oracle.
- How OODBMSs and ORDBMSs compare in terms of data modeling, data access, and data sharing.

Possible difficulties

The material on Postgres (Section 28.3) may prove to be too abstract unless you have access to this system. However, the material is useful for comparative purposes.

For an understanding of query optimization, you will need to have covered the material in Chapter 21.

Due to the newness of the material on SQL:2003, and the fact that there are currently no DBMSs that are fully compliant with the new standard, much of the work may have to be paper-based (although you may be able to illustrate some of the features in some RDBMSs). Care has to be taken if you illustrate the object features within systems like Oracle, which will not fully comply with the SQL:2003 standard.

Teaching hints

Some DBMSs already have object-management features, and if the students have access to one of these systems, you can go through the features it offers. Care must be taken to illustrate how the features of this system differ from the proposed SQL:2003 standard, when necessary.

Possibly the best way to go through the material on SQL:2003 is by going through a series of worked examples, using a different case study. There are some examples in the exercises associated with this chapter, and you may wish to supplement or replace this with your own examples.

If you use Oracle, you can use Section 28.6 to provide the students with concrete examples of objects management within a relational system. Even if you do not have access to Oracle, the examples in Section 28.6 may make the concepts more realistic for the students.

Part 8

Web and DBMSs

CHAPTER 29

Web Technology and DBMSs

Aims

Part 8 of the book deals with the integration of the DBMS into the Web environment, semistructured data and its relationship to XML, XML query languages, and mapping XML to databases.

The aim of the first chapter in this part of the book is to examine the integration of the DBMS into the Web environment. After providing a brief introduction to basic Internet and Web technology, the chapter examines the appropriateness of the Web as a database application platform and discusses the advantages and disadvantages of this approach. It then considers a number of the different approaches to integrating databases into the Web environment, including scripting languages, CGI, server extensions, Java, Active Server Pages, and Oracle's Internet Platform.

Points to emphasize

- The basics of the Internet, Web, HTTP, HTML, and URLs.
- The advantages and disadvantages of the Web as a database platform.
- Approaches for integrating databases into the Web environment:
  - scripting languages (JavaScript, VBScript, PHP, and Perl);
  - Common Gateway Interface (CGI);
  - HTTP cookies;
  - extending the Web server;
  - Java, J2EE, JDBC, SQLJ, CMP, JDO, Servlets, and JavaServer Pages (JSP);
  - Microsoft Web Platform: .NET, Active Server Pages (ASP), and ActiveX Data Objects (ADO);
  - Oracle Internet Platform.

Possible difficulties

There is a significant amount of material in this chapter. For a first-level course in database management, you may elect to omit this material or significantly curtail what is covered. Further, the Web is such a dynamic environment at present, and there are many new developments in this area, that it can be difficult to decide what to cover.

The material covered in Sections 29.1 and 29.2 act as a good introduction to the Web environment. However, by itself, the chapter does not provide sufficient material to allow students to create HTML or complete Web – DBMS applications, and so additional supplementary material has to be provided (see the bibliography and further readings for this chapter for some suggestions).

Teaching hints

Concentrate initially on the introductory material in Sections 29.1 and 29.2. If the DBMSs that you use are integrated with the Web environment, this would be a sensible way to get the students started. Start off by giving them small assignments that slowly get them into creating HTML documents and integrate their database into the Web.

You may find the supplementary material on the Web site for this book that shows the implementation of part of the *DreamHome* case study using ASP useful. Appendix I on the Web site for this book is another useful source of examples. After that, other approaches can be introduced and the students can gain direct experience of the advantages and disadvantages of different approaches. The Web itself is the source of an enormous amount of material on this topic and there are many Web sites with tutorials and examples of how databases can be accessed from the Web. A number of useful developers' Web sites are listed in the Further Reading section for this chapter at the end of the book.

If the students have access to a DBMS that provides Web-related features, it would be worthwhile going through these features with the students and examining the advantages and disadvantages of each approach offered.

CHAPTER 30

Semistructured Data and XML

Aims

The aim of this chapter is to examine semistructured data and its relationship to XML. The chapter then examines XML and its related technologies, such as namespaces, XSL, XPath, XPointer, XLink, SOAP, WSDL, UDDI, XML Schema, and RDF. The chapter also examines query languages for XML and, in particular, concentrates on XQuery, as proposed by W3C. It also examines the extensions added to SQL:2003 to enable the publication of XML and more generally mapping and storing XML in databases.

Points to emphasize

- What semistructured data is.
- The concepts of the Object Exchange Model (OEM), a model for semistructured data.
- The basics of Lore, a semistructured DBMS, and its query language, Lorel.
- The main language elements of XML.
- The difference between well-formed and valid XML documents.
- How Document Type Definitions (DTDs) can be used to define the valid syntax of an XML document.
- How the Document Object Model (DOM) compares with OEM.
- About other related XML technologies, such as namespaces, XSL and XSLT, XPath, XPointer, XLink, and XHTML.
- The limitations of DTDs and how the W3C XML Schema overcomes these limitations.
- How RDF and RDF Schema provide a foundation for processing metadata.
- The W3C XQuery Language.
- How to map XML to databases.
- How the new SQL: 2003 standard supports XML.
- How Oracle supports XML.

Possible difficulties

There is a significant amount of material in this chapter. For a first-level course in database management, you may elect to omit this material or significantly curtail what is covered. Further, XML-related technologies and the XML query languages are still in development, so you will have to continue to monitor changes in this area.

Teaching hints

If you have access to a semistructured system, such as Lore, then you can make the material more concrete by giving the students additional exercises in a lab environment. Lore is freely downloadable from the Lore Web site.

Concentrate on Example 30.1 to introduce the concepts of the semistructured data model and Example 30.2 to show sample queries for a semistructured system.

Once you have introduced XML, get the students to create XML files and display them within a suitable browser (such as Internet Explorer 6). After that, gradually introduce the XML-related technologies (such as DTDs, XML Schema, and namespaces) and again get the students to experiment with these technologies in a lab environment.

If you have access to an XML query language (there are several freely downloadable versions of XQuery), again get the students to experiment with this language in a lab environment to make the material provided in the chapter more realistic.

Due to the newness of the material on XML/SQL:2003, and the fact that there are currently no DBMSs that are fully compliant with the new standard, much of the work in this section may have to be paper-based (although you may be able to illustrate some of the features in some RDBMSs). Care has to be taken if you illustrate the XML features within systems like Oracle, which will not fully comply with the SQL:2003 standard.

Part 9

Business Intelligence

CHAPTER 31

Data Warehousing Concepts

Aims

This chapter aims to introduce the concepts of data warehousing and to discuss how these systems are capable of potentially delivering competitive advantage to an organization. The chapter also describes the relationship between data warehousing and Online Transaction Processing (OLTP) systems and to identify the main characteristics of these systems. It identifies the main problems associated with the development and management of a data warehouse, and describes the architecture, tools, and technologies of a data warehouse. The chapter also introduces the concept of data marts and the issues associated with the development and management of these systems.

Points to emphasize

- How data warehousing evolved.
- The main concepts and benefits associated with data warehousing.
- How Online Transaction Processing (OLTP) systems differ from data warehousing.
- The problems associated with data warehousing.
- The architecture and main components of a data warehouse.
- The important information flows or processes of a data warehouse.
- The main tools and technologies associated with data warehousing.
- The issues associated with the integration of a data warehouse and the importance of managing metadata.
- The concept of a data mart and the main reasons for implementing a data mart.
- The main issues associated with the development and management of data marts.
- How Oracle supports data warehousing.

Possible difficulties

Data warehousing is a large and complex area and this chapter aims to simply provide an overview and identify the important issues of this subject.

As data warehouses are corporate-wide systems it may be difficult for students to visualize the potential complexity and size of these systems. The theoretical material should be supplemented with case studies to ensure that students can appreciate the 'real' benefits and problems associated with these systems.

Teaching hints

Data warehousing is a relatively new technology and attention should be given to the latest developments.

Students may have difficulties understanding how a database built for strategic decision support differs from a traditional database system. Students should be given or asked to provide typical queries for each type of database and then these can be used to identify the requirements of a database to answers these questions.

CHAPTER 32

Data Warehousing Design

Aims

This chapter focuses on the issues associated with data warehouse database design. Since the 1980s, data warehouses have evolved their own design techniques, distinct from transaction-processing systems. Dimensional design techniques have emerged as the dominant approach for most data warehouse databases.

Points to emphasize

- The issues associated with designing a data warehouse database.
- A technique for designing a data warehouse database called dimensionality modeling.
- How a dimensional model (DM) differs from an Entity–Relationship (ER) model.
- A step-by-step methodology for designing a data warehouse database.
- Criteria for assessing the degree of dimensionality provided by a data warehouse.
- How Oracle Warehouse Builder can be used to build a data warehouse.

Possible difficulties

The design of a data warehouse may appear to be very abstract if the students then do not go on to implement it. If you have access to a data warehouse then get the students to go on to implement their designs.

Teaching hints

In this chapter, we describe one approach to designing the database component of a data warehouse / data mart. The approach is illustrated using a relatively simple example drawn from the *DreamHome* case study. This example does not include a discussion on the decisions taken in reaching an optimum design as this is out of the scope of the chapter.

CHAPTER 33

OLAP

Aims

The aim of this chapter is to describe online analytical processing (OLAP) and the main features associated with OLAP applications. The chapter discusses how multi-dimensional data can be represented and the main categories of OLAP tools. It also discusses the OLAP extensions to the SQL standard and how Oracle supports OLAP.

Points to emphasize

- The purpose of online analytical processing (OLAP).
- The relationship between OLAP and data warehousing.
- The key features of OLAP applications.
- The potential benefits associated with successful OLAP applications.
- How to represent multi-dimensional data.
- The rules for OLAP tools.
- The main categories of OLAP tools.
- OLAP extensions to the SQL standard.
- How Oracle supports OLAP.

Possible difficulties

Students may have difficulties visualizing how three-dimensional cubes are capable of representing multi-dimensional data. A number of examples of multi-dimensional data should be used to demonstrate how cubes are ideal for representing such data. Examples should also illustrate the common analytical operations such as consolidation, drill-down, and 'slicing and dicing'.

This chapter includes a discussion on how the SQL standard has been extended to provide advanced data analysis capabilities. It is possible that students will not have access to a version of SQL supporting these advanced capabilities and therefore this discussion may not be demonstrated in a laboratory.

For an in-depth coverage of this area see the literature listed in the Further Reading section.

Teaching hints

OLAP is an emerging technology and attention should be given to the latest developments.

Students may have difficulties understanding how multi-dimensional data is best represented using a cube. It may be useful to first discuss the representation of multi-dimensional data using a structure the students are familiar with such as the two-dimensional relational table. This representation can then be compared with representing multi-dimensional data using a three-dimensional cube. The advantages of using the cube to represent multi-dimensional data should become more apparent.

The theoretical discussion on OLAP tools should be accompanied with practical experience of such tools, if possible.

CHAPTER 34

Data Mining

Aims

The aim of this chapter is to describe Data Mining (DM) and the main features of DM applications. The chapter describes the main characteristics of data mining operations and associated techniques. It describes the process of DM and the main features of DM tools with particular coverage of Oracle DM.

Points to emphasize

- The concepts associated with data mining.
- The main features of data mining operations, including predictive modeling, database segmentation, link analysis, and deviation detection and the associated techniques.
- The techniques associated with the data mining operations.
- The process of data mining.
- Important characteristics of data mining tools.
- The relationship between data mining and data warehousing.
- How Oracle supports data mining.

Possible difficulties

Data mining is a large complex area. This chapter aims only to provide an overview of the main benefits and characteristics of data mining. For a more in depth coverage of this area see the literature listed in the Further Reading section.

Teaching hints

Data mining is an emerging technology and attention should be given to the latest developments.

The theoretical discussion on data mining tools should be accompanied with practical experience of such tools, if possible.