LEARNING OBJECTIVES

On completion of this session you will be able to:

- Describe the 3 level ANSI SPARC Database Architecture and the advantages which its inherent data abstraction provide to the database developer
- Explain the role of database development within an information system
- Describe the steps involved in the Systems Development Life Cycle (SDLC)
- Explain the steps involved with the Database Life Cycle (DBLC)
  - Initial Study
  - Database Design
  - Implementation and loading
  - Testing and evaluation
  - Operation
  - Maintenance and evaluation
- Explain, in detail, within the Database Design phase the role of
  - ER modelling and Normalisation
  - Data Model Verification
  - Distributed Database Design
  - Logical and Physical Design
- Describe the database design strategies which exist
  - Top-down vs. bottom-up design
  - Centralized vs. decentralized design
Reading

Prescribed readings

Chapter 2 - Section 2.5, Chapter 8

Chapter 2 – Section 2.5, Chapter 9

Further references

Website:
Rob & Coronel textbook

Where we are

- **Introduction to Database Systems**
- **The Relational Model**
- **Database Lifecycle**
  - Conceptual Design
  - Logical Design
  - Physical Design
- **Normalisation**
- **Implementation**
  - SQL (DML)
  - SQL (DDL & DCL)
  - Transaction Management
- **Database Administration**
- **Data Warehousing, Data Mining, ECommerce**
1. **Introduction**

In your first study guide you have been introduced to the fundamental concepts of database systems. This study guide provides an overview of the manner in which databases are designed and how such design relates to the wider manner in which information systems are developed.

2. **The ANSI/SPARC database model**

In the first study guide you have examined the range of data models that are commonly found in the database area. An alternative way of viewing these models is by examining the level of data abstraction (the degree to which the implementation details are hidden). One of the major reasons for the success of the relational database model lies in the fact that the relational database architecture allows database objects (tables) to be created without any reference to the physical details of how the table is actually stored on a computer’s disk drive.

ANSI SPARC has developed a model for database architecture which is known as the “**Three Level ANSI/SPARC Database Model**”. This model is based on a number of fundamental principles:

- Users should not need to know the details of the physical database storage
- All users should be accessing the same set of data – the subset of the data which a particular user can see is known as a *User View*. Changes in one user view should not affect other non-related user views. For example, in a company employee system there will be views of the data as seen by the payroll department and a more restricted view as seen by say the company social club.
- There should exist a single collective view of the entire database (the *Conceptual View*) which may be modified without impacting all user views, and
- The conceptual view should be immune to changes in the structures under which the data is stored on a disk drive (the *Internal View*). For example, the data may be stored using ISAM files or B-Trees (these are different storage structures which offer a range of performance advantages and disadvantages).
This model can be represented by:

![Diagram of database levels]

The External Level represents the users' view of the database; it describes that part of database that is relevant to a particular user.

The Conceptual Level represents the community view of the database and describes what data is stored in the database and what relationships exist among the data.

The Internal Level is the physical representation of the database on the computer; it describes the data structures which are used to store data.

Each of these levels can be represented by a schema, essentially a ‘map’ representing that particular level.

By separating the external schema and the conceptual schema, the community view of the database can be changed (for example by adding a new ‘thing’ we wish to capture data about) - this is known as Logical Data Independence. In a similar manner, separating the conceptual schema and the internal schema means that we can change a storage structure without it affecting the community view of the database – this is known as Physical Data Independence.

Much of our work in this unit will concentrate at the conceptual level and be involved with the database design techniques we use to arrive at a conceptual schema.
The conceptual level of a database can be represented by several techniques. The following illustrate a system for recording tour details and customer bookings for the 'Fly By Night' travel agency.

(i) As a non DBMS specific Entity Relationship Diagram (ERD):

![ERD Diagram](image1)

(ii) As a non DBMS specific Data Structure Diagram (DSD) or Extended Bachman Diagram:

![DSD Diagram](image2)
Some authors do not differentiate between the ERD and DSD as pictured above. R&C describe the ERD presented here as a Chen Model and that shown under the DSD heading is 'similar' to their Crows foot model. Many different ERD diagramming conventions exist; you should not be concerned about this – all modelling approaches using the Entity-Relationship approach show entities, attributes and the relationships between entities. These are the base objects and properties from which we can create a conceptual map of the database storage structures as a set of 'tables'.

(iii) As a DBMS specific schema file (only part shown):

```sql
CREATE TABLE CUSTOMER (
  cust_no        NUMBER(5) NOT NULL,
  cust_family    CHAR(20) NOT NULL,
  cust_given     CHAR(20) NOT NULL,
  cust_street    CHAR(20) NOT NULL,
  cust_town      CHAR(20) NOT NULL,
  cust_postcode  CHAR(4) NOT NULL,
  cust_phone     CHAR(10),
CONSTRAINT pk_CUSTOMER PRIMARY KEY (cust_no));
...
```

This text file contains Structured Query Language (SQL) Data Definition Language (DDL) statements which are 'run' via the selected DBMS software to create the desired database storage structures to represent the conceptual model represented by the ERD. This set of DDL statements is known as a schema file – the file is a plain text (ASCII) file with a name such as fbnschema.sql. This schema file can be hand coded using a text editor or, in commercial environments, will be generated by CASE tools.
Sample run of the schema file created from the ERD shown in this section. Note that the DSD has seven record types show (boxes) which have been translated in seven tables.
3. **Database systems development**

When we speak of 'data' we are referring to raw facts – for example the value 45 carries no information in its own right. However if this data is placed in context as the maximum number of participants for say tour 50 in our tour booking system, then it now provides information which can be used to make a decision about, say, if any further bookings for this tour can be accepted.

As discussed in section 8.1 of R&C "... a **database** is a carefully designed and constructed repository of facts." Such repositories are themselves components of wider Information Systems. Information Systems are developed via a process known as the Systems Development Life Cycle (SDLC) which is presented in section 8.3 of your text. You will examine Systems Development in more detail within your course when you study FIT2001 Systems Analysis and Design. The SDLC provides an overall systems framework within which the more specific database development takes place.

The first step of the SDLC, Planning, evaluates the current system and examines alternative approaches. At this stage it might be decided that the current system is adequate or that a new/enhanced system based around a database approach might be advised. Within the SDLC, database development follows a life cycle of its' own known as the DataBase Life Cycle (DBLC). As described in section 8.4 of your text the DBLC consists of six stages:

- Database initial study
- Database design
- Implementation and Loading
- Testing and Evaluation
- Operation, and
- Maintenance and evaluation

Although we will be examining all six stages of this lifecycle our major emphasis will be on stages 2 and 3 – creation of the database design followed by conversion to a DBMS specific schema file and implementation within the chosen DBMS software.

Stage 2 Database Design is further subdivided into:

- Conceptual Design – creation of a high level abstracted model represented by the Entity Relationship Diagram
• DBMS Software Selection – selection of the specific DBMS software to be used eg. MySQL, Oracle, MS Access etc.
• Logical Design – translation of the conceptual designs ERD into a schema file for the selected DBMS, and
• Physical Design – selection of appropriate storage structures and access methods to optimise performance

Read  
R & C 6th Edn, p.396 – 426  Sections 8.1 – 8.4  
R & C 7th Edn, p.360 – 387 Sections 9.1 – 9.3

4. Database design strategies

As you read through section 8.4 you should have noted the role of entity relationship modelling and normalisation in building the conceptual model. Normalisation is used to ensure that the entities which are depicted on the ERD are in their most conceptual 'efficient' form. ER modelling is a top-down approach in which essentially we figuratively 'stand above' the model and identify entities and the relationships between them. For small models with few entities, an alternative approach – bottom-up design can be used which begins with a 'pile' of data representing the data universe for the model and applies normalisation techniques to extract entities and relationships. For this unit we will be using a top-down approach. Study Guide 4 will examine ER modelling and Study Guide 6 will look at how normalisation is used to check an entities structure.

Read  
R & C 6th Edn, p.427  Section 8.5  
R & C 7th Edn, p.387 – 388  Section 9.4
5. Centralised and decentralised design approaches

This section of your text looks at the general techniques which are used to create database models. Small systems, such as you will typically complete for this unit, can be managed via centralised design, however large complex systems mean that the design task needs to be segmented and completed by different teams. In this situation, when the various segments are aggregated to produce a single overall DBMS model, a number of aggregation problems can occur as depicted in table 8.17.

6. Summary

| Read | R & C 6th Edn, p.428 – 430  Section 8.6 |
| Read | R & C 7th Edn, p.388 – 390  Section 9.5 |

| Do | R & C 6th Edn, Chapter 8 Review Questions 1 - 11 |
| Do | R & C 6th Edn, Chapter 8 Problems 1 - 5 |
| Do | R & C 7th Edn, Chapter 9 Review Questions 1 - 12 |
| Do | R & C 7th Edn, Chapter 9 Problems 1 - 5 |