

# **(Free) Sample 1**

*Taken from:*

*E-book 1: Database Fundamentals*

## **Knowledge River Ltd**



(Free) Sample 1

Welcome to **Knowledge River** – the on-line e-learning store for Computer Science practitioners, academics and students.

Thank you for taking the time to download and examine this free sample – based on one of our popular e-learning products. In this free sample we have tried to give you a flavour of the full document upon which it is based. If you like what you see and would like to purchase the full document then please follow the instructions on our website: [www.knowledge-river.co.uk](http://www.knowledge-river.co.uk)

Whilst every effort has been made to ensure the accuracy of this material and any included software code has been tested carefully, they are only intended for instructional and illustrative purposes and are not guaranteed for any particular application, purpose or function. Knowledge River Ltd does not offer any warranties or representations, nor do we accept any liabilities with respect to any code examples included here.

Please note that Knowledge River provides Computer Science educational material in electronic format – we do not offer an on-line IT consultancy or help-desk function and so we cannot get into detailed correspondence on specific technical issues. However, we would appreciate general feedback and comments on what you think of the material, its layout, its usefulness and how it could be improved etc.

Please email us at: **feedback@knowledge-river.co.uk**

Copyright © Knowledge River Ltd 2006-7

All rights reserved.

Please see the accompanying notes on the Knowledge River Ltd usage policy.

## Knowledge River Ltd - Usage Policy (How to use our products)

We hope you enjoy using these e-learning products and get a lot of benefit by doing so. However, these electronic resources do represent a very considerable investment in intellectual effort, time and money on our behalf and naturally we want to protect our intellectual property. These products are meant for *your own personal educational use* and should not be modified, altered, edited, re-sold or transferred to a third party.

Please use and enjoy our work but please respect our efforts too.

The table below summarizes what you can and cannot do with our e-learning products:

Action	Allowed?	Notes
Reading	Yes	Obviously! On-screen or off-screen (see printing).
Printing / Making a Hardcopy	Yes (Limited)	Only single copies for <i>your own personal use</i> . Multiple copies and/or photocopies intended for distribution or sale to third parties is NOT allowed.
Editing / Updating / Extending	No	The e-learning products you purchase from Knowledge River Ltd must NOT be edited or modified in any way. This is how they were intended to be and this is how we want them to stay. Edits are our responsibility.
Copying (Electronic)	Yes (Limited)	You may copy the purchased e-learning product for <i>your own personal use</i> within your own computing equipment and storage devices (eg on a desktop and laptop with a backup on a portable storage device). However, you must NOT make multiple copies of these products with the intention of passing these copies onto third parties (so doing copies for your friends or putting copies on a server for sale or distribution is not allowed). In short, keep one or two copies for <i>your own personal convenience</i> but do not give copies to other people. This rule applies regardless of whether or not you charge money for it.
Transferring / Passing On	No	Each e-learning product purchased is intended for <i>one individual</i> – so please do not pass copies onto third parties – let them purchase their own. This rule applies even if there is no money involved or financial gain for yourself – please do not pass these products around – ever. Knowledge River products are non-transferable.
Re-selling	No	It goes without saying that this is never allowed – taking a copy of our work and then selling it on is simply theft and is immoral - please don't even think about doing this.
Anything Else	No	Anything not already covered is not allowed – please treat our company and products with respect. Enjoy.

## **The full *E-book 1* contents are:**

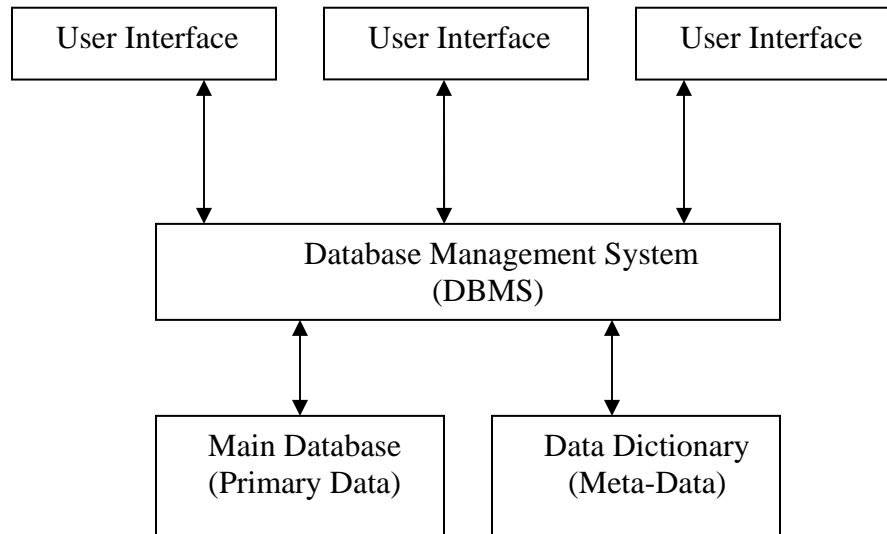
- 1. What exactly is a database?**
- 2. What is a Universe of Discourse?**
- 3. What is persistent data?**
- 4. What is meta-data?**
- 5. What type of data goes in a database?**
- 6. What is the architecture of a database system?**
- 7. What is ANSI-SPARC?**
- 8. Why Three Levels?**
- 9. What is data independence?**
- 10. What is the history of databases?**
- 11. How was data stored before databases?**
- 12. What exactly is the program-data dependence problem?**
- 13. Are there any more limitations with files?**
- 14. What benefits does a DBMS provide?**
- 15. What functionality does a DBMS provide?**
- 16. Are there any disadvantages to the database approach?**
- 17. Who are the key suppliers in the database world?**
- 18. What are the main roles within a database system?**
- 19. Which type of data is good for databases (and why)?**
- 20. Which type of data is bad for databases (and why)?**
- 21. Can you change or delete the database structure and/or contents?**
- 22. How do you go about building a database?**
- 23. Once built, how do you interact with a database?**
- 24. How does data, information and presentation relate?**
- 25. Are databases held in one location?**

Like this sample, the full document is in PDF format and can be purchased individually at a modest price or you may like to buy several different documents and get *very substantial discounts* – full details on our website. Thank you once again for taking the time to examine this **Knowledge River** educational material – please have a look at our website or drop us an email.

**Below are three samples...**

## What is the architecture of a database system?

This is another way of saying ‘how are database systems put together?’ By their very nature, database systems are complex pieces of software with different, co-operating components but the diagram below gives the general picture:



As you can clearly see, there are three distinct levels to a database system – a common model described by the **ANSI-SPARC** architecture (see next question). As we have already stated, the real-world data is held in the main database while the database definition (the meta-data) is held in the data dictionary. Both are under the direct control of the DBMS (the DBMS is the Database Management System and is a specialized and complex piece of software that controls all database activity – well known DBMS examples are Oracle™, Microsoft™ SQL Server and IBM™ DB2). Note – the DBMS is not the same as the database. Each DBMS - a complex piece of software - will supply a complex set of functions for the users of the system – these are covered soon. A database is simply a pool of stored data held on a permanent storage device – usually a set of one or more large hard disk(s).

As for the user interface, these can be quite varied – depending upon the user requirements. For example, an administrative person may need a forms-based or web-based graphical user interface – for data insertion and extracting data. A technical person (such as a database programmer) may need to write and execute computer code to run against the DBMS – they can use a text-based (command-line) type of interface. Database administrators may use either type of interface. It is also possible to run other application code (such as Java, C or COBOL programs against the DBMS) – so that they can interrogate the database.

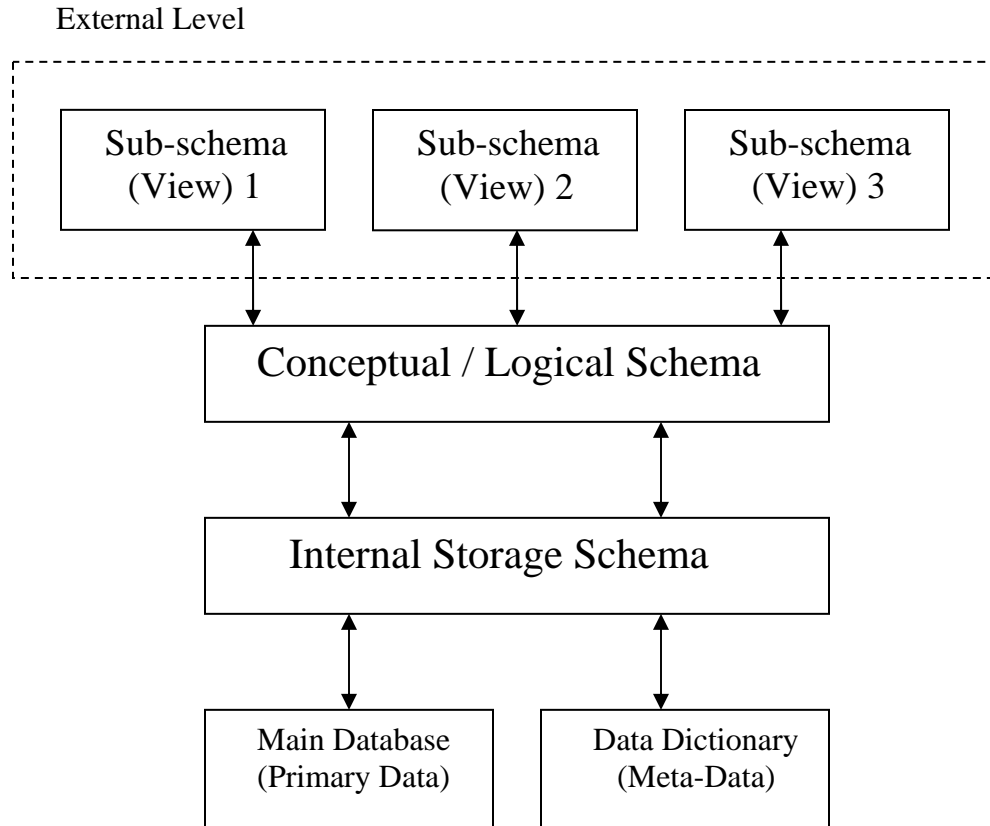
So there are many ways to interact with a DBMS. The common issue, regardless of the precise type of interface, is that all end-users or application programs must go through the DBMS to access the underlying data – there is no such thing as direct, external access with a database system. Think of a DBMS as a gatekeeper to the database – you must pass through the gate.

### **What is ANSI-SPARC?**

This stands for American National Standards Institute – Standards Planning And Requirements Committee. This harks back to the 1970s (when great advances in database ideas and technologies were being made). Even before ANSI-SPARC, there had been the DBTG (Database Task Group) appointed by the Conference on Data Systems and Languages (CODASYL) – this was the early 1970s. So what was all this activity trying to achieve?

In short, as new database ideas and competing technologies started to emerge in the 1970s, it was desirable to have some commonly agreed terminology and models for how to put complex database systems together. The end result of all this standardization work is the well-known **3-level architecture** model – the components being: a single, central (conceptual or logical) schema that defines the logical structure of the whole database, one or more (logical) sub-schemas (sometimes called views or external schemas) that each define a specific ‘window’ onto the underlying database (so each user interface uses a specific sub-schema that defines what that user can see) – collectively these sub-schemas are called the external level - and finally, a single internal storage schema that defines how the underlying data is physically stored and organized.

This gives the familiar multi-level database architecture below:



### How was data stored before databases?

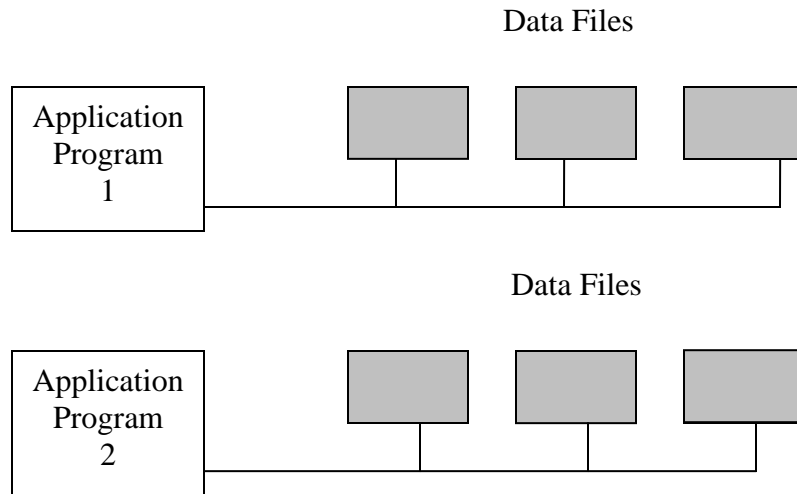
Well, having just described two approaches to databases that fell by the wayside (hierarchical and network) and whetted your appetite for the e-book on the dominant relational database model, we conclude our historical investigations by looking at how data was stored before DBMS systems were developed – enter **file-based** systems.

A file-based system is sometimes known as an *application-led* system whereby a set of application programs (written in COBOL, C, Fortran etc) each has one or more data files – the fundamental point being that the program *defines and manages its own data*.

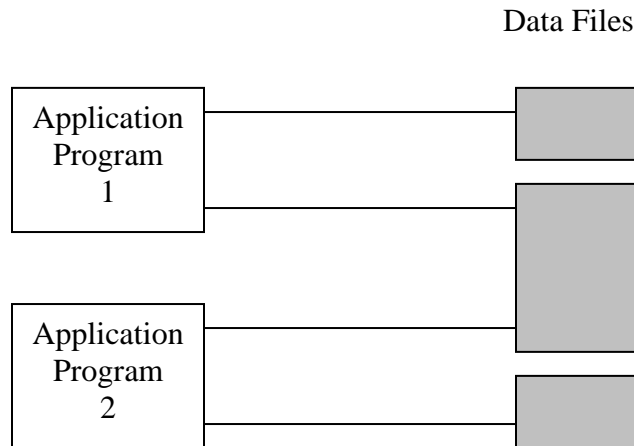
This is a key phrase because, unlike the database approach, in a file-based system the data in the files and the logic in the application code are tightly linked – one defines the other. Consequently, to change one (file or program) means changing the other too.

The diagram below sums up this situation:

(Free) Sample 1



Here the two application programs use their own separate data files – however, it is possible for the same data file to be shared between two or more application programs:



Regardless of program/file permutation, this approach to data storage has some fundamental problems – mostly caused by the fact that the precise format of the data in the files is determined by the logic in the program.

This is the so-called **program-data dependence** problem.