Topic 01: Introduction to Databases

ICT285 Databases

Dr Danny Toohey

Topic Learning Outcomes

- At the completion of this topic, you should be able to:
  - Define what a database is and compare the database approach with the use of file systems to manage data
  - Define the features of a data model and explain several data models including the relational data model
  - Discuss various approaches to database architectures and data independence
  - List and explain the features of a relational DBMS

Topic Outline

- What is a database?
- Data models
- Three-level architecture
- Functions of a DBMS

Topic 01: Part 01

What is a database

What is a database?

- Definitions of ‘database’ usually include some mention of a *collection or set of facts about something of interest*
  - This collection allows us to record things that have happened
  - …and to make conclusions about the area of interest
  - Which means not having to “…start from scratch” all the time
- So, how can we record the things that have happened?

Attribute:Value

- We can record things as attribute:value pairs
  - E.g., Name: Bryan…Name: Rodney
- When the data become more complex, we will need more complex structures to represent them
  - In this example, the *subject* (which is ‘person’) has only one *property* (‘Name’)
  - In most cases, subjects will be more complex and so we need to be able to represent sets of properties

An example

- The *subject* School has the *properties* Name and Office (among others)
  - We can show this as:

    School (Name, Office)

    School (IT, 245.2.026)
  - We could also explicitly match the *values* with their properties:
    - School (Name:IT, Office: 245.2.026)
  - We could also explicitly state the *measure*, or range of possible values for each property:

    School (Name:Set of School Names, Office:Set of university rooms

Different types of databases

- Where types of database differ is how they represent this Label, Value, Measure triplet
- For example, GPS data is structured as 'sentences'
  - Each sentence begins with an ID which tells us the subject of the sentence
  - The values of each of the properties are 'labelled' by their position in the sentence
  - The measure of each value is known to the applications using the data

$GPGGA,181908.00,3404.7041778,N,07044.3966270,
*W,4,13,1.00,495.144,M,29.200,M,0.10,0000*40*

http://aprs.gids.nl/nmea/

Relational Databases

- In a relational database table, the *label* is provided by the attribute name, the *measure* or domain by the definition of the attribute, and the *values* are shown in the cells of the table structure

StudentID	FamilyName	Degree	Major	GPA
12345678	WELLS	BSc	ISD	3.00
12456789	NORBERT	BSc	CS	2.70
23456789	KENDALL	BSc	GT	3.50

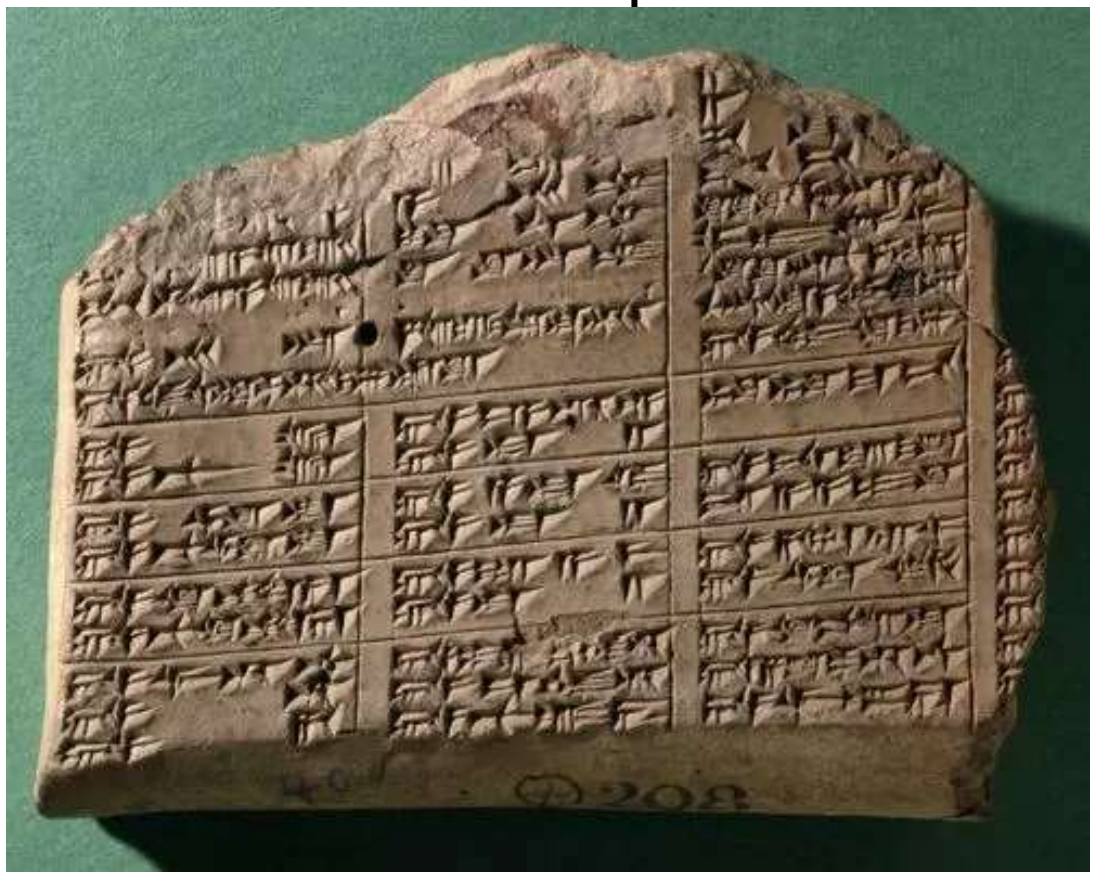
So then, what is a database?

- "…(it) *is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database*"
- "…(it) *represents some aspect of the real world … changes … are reflected in the database*"
- "…(it) *is designed, built and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested*."

(Elmasri & Navathe, 2011, p.2)

Early databases

- Record keeping didn’t start with computers…



https://www.ancient.eu/cuneiform/

Manual filing systems

- More recently, many record-keeping systems were manual (and many still are). A good example is filing cabinets
- Typically, the files will be physically grouped according to some logical grouping

- We can get an idea of what is in the files by:
  - Their location
  - Their label

Computerised file-based systems

Early computerised 'databases' attempted to replicate the manual filing systems they were intended to replace

- Data that were logically related were stored physically together in **files**
- Typically, each application would have its own data files: e.g. a university student records system:

> *The examinations office keeps a file of students and their grades. Programs are written to print academic transcripts, and enter new grades. The finance office keeps a file of students and the fees they pay. Both offices are interested in data about students, but have separate files and programs to manipulate the data in those files – because each requires data not available from the other's files.*

(Adapted from: Elmasri & Navathe, 2000)

This is OK when…

The data are:

- Simple
- Small in number
- …and the reporting requirements of the organisation are few and won't change much

BUT…

When this isn't the case, we get problems with:

- Separation and isolation of data
- Data redundancy
- Data dependence

Database approach

In the database approach, the data is defined (in a data dictionary), stored and managed *independently* of the applications that access it

- **data independence**

- This means that many applications can access the *same* data

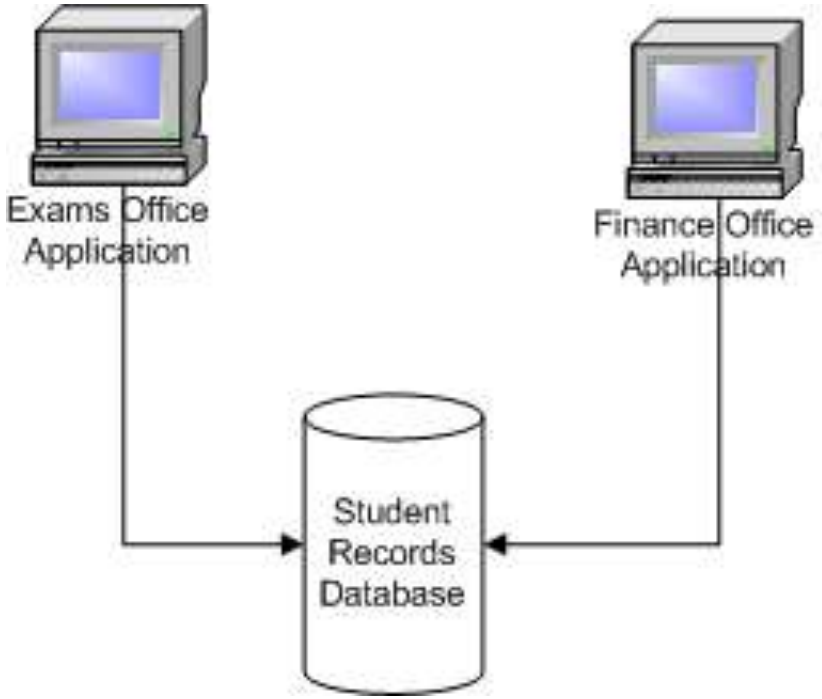


Image from Elmasri, R., and Navathe, S.B. 2011,
Database Systems: Models, Languages, Design, and Application Programming, 6th Ed., Pearson

The database approach…

Some of the potential positive results of this approach could include:

- Ability to change the structure of the data without affecting the applications that access the data
- Control of data redundancy
  - Less data duplication
- Data consistency
- Better access to data
- Sharing of data
- Improved data integrity
- Improved security
- Enforcement of standards
- Economies of scale

What do you think some of the negatives might be?

Implementing the database approach

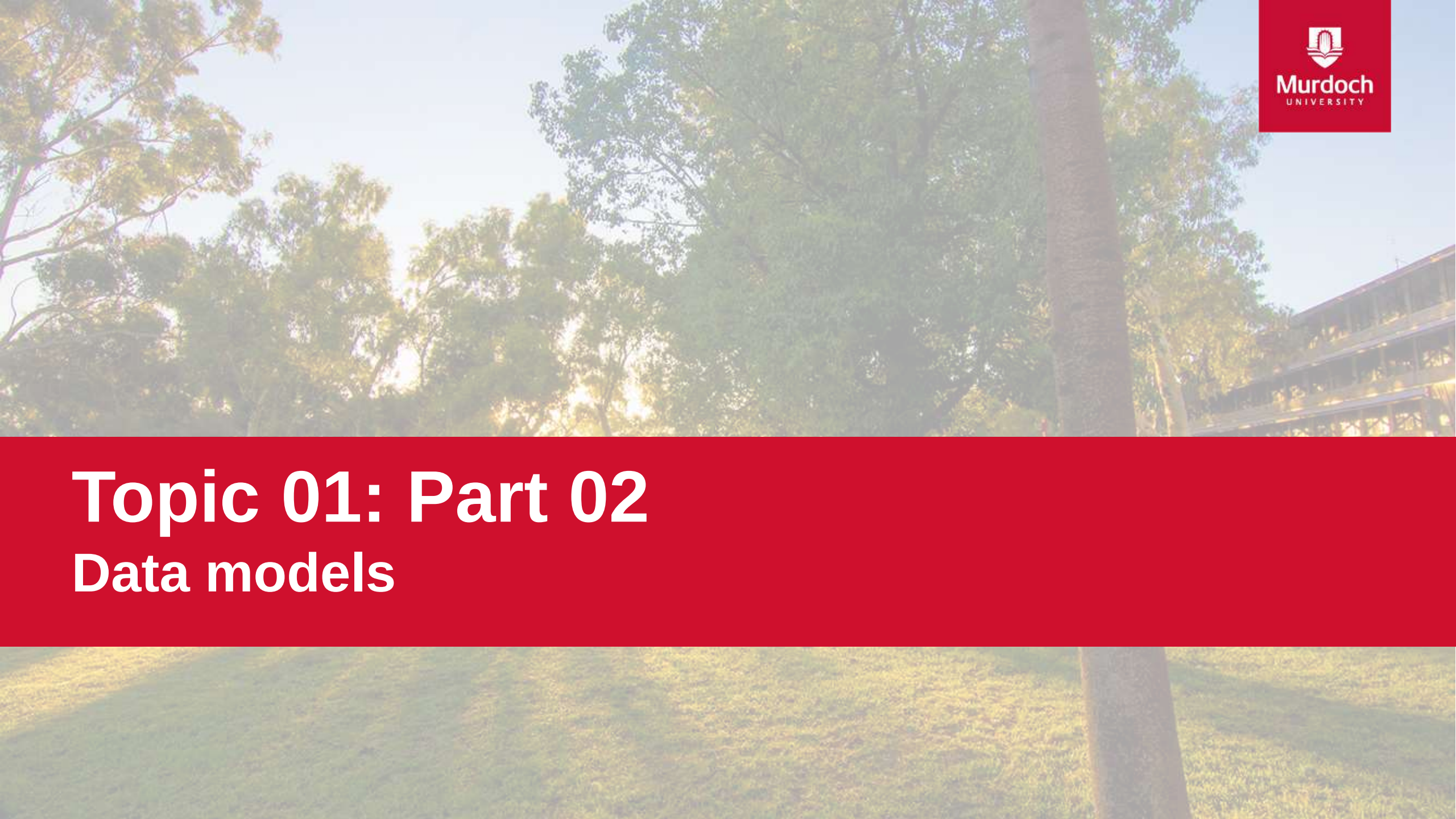
A number of approaches have been used to implement the ‘database approach’

These approaches fall into a few main groups:

- Navigational
- Relational
- Object-oriented
- Non-relational (noSQL)

The take-aways…

- A database is a self-describing collection of logically related records
- The database approach enabling the sharing of the *same* pool of data for different applications
- This reduces problems relating to separation, isolation and duplication of data within different systems that can occur with the file system approach
- The trade-off for these advantages is much greater complexity in the database approach



Topic 01: Part 02

Data models

Data Models

In order to discuss the various approaches to databases, we first need to understand what a data model *is*…

- Date (2000) says that it is

  "…an abstract, self-contained, logical definition of the objects, operators, and so forth, that together constitute the abstract machine *with which users interact. The objects allow us to model the* structure *of the data. The operators allow us to model its* behaviour*."* (p. 14)

- Garcia-Molina et. al., (2009) also suggest that a data model consists of *constraints* that can be applied against the data in the database

So, we can define a data model as an abstract concept that has:

- **Structure**
- **Operators**
- **Constraints**

Navigational databases

- The major *structural* feature of navigational databases is that the records are related in a fixed (hierarchical or network) structure
  - Two major commercial implementations
    - IMS
    - IDS (later IDMS)

IMS

IBM's Information Management System

Developed in the 1960's for the Apollo moon landing program

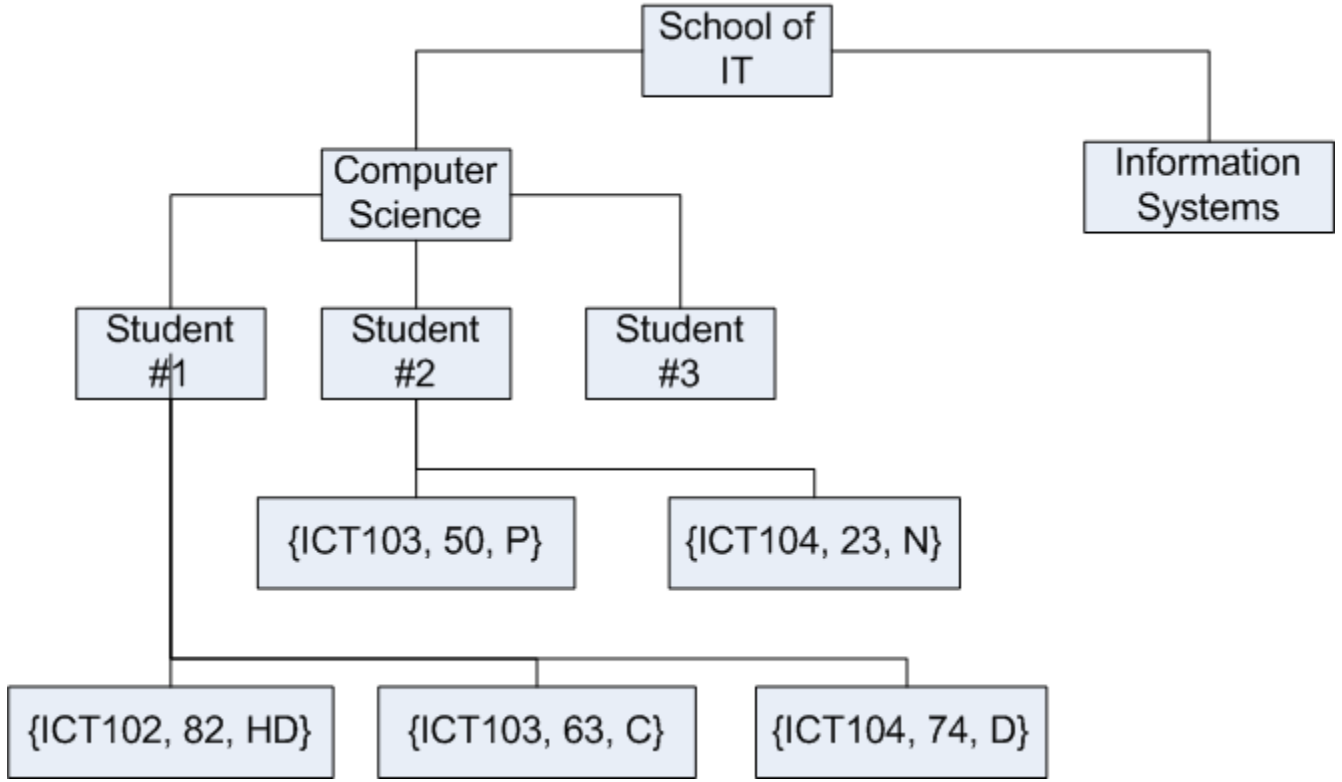
- Need to manage 'bill of materials' data

Resulted in

- Simplification of the representation of the data involved in the project
- Standardised data definitions
- Reduction in the amount of programming that had to be done
  - "*By building data base capabilities in the form of software, one team could build the data base software once and all the rest of the application programmers could benefit from it*." Berman (2007)

https://www.ibm.com/support/knowledgecenter/SSEPH2_11.1.0/com.ibm.imsintro.doc.intro/Intro.html

Hierarchical records…



This could be arranged in many different ways, for example, we could have units at the top, then students in the units and their results….

Image by D.Toohey

IDS

- IDS was developed by Bachman at General Electric Corp in the 1960's
- Network database developed to enable representations of more complex data structures than available in hierarchical systems

http://wp.sigmod.org/?p=688

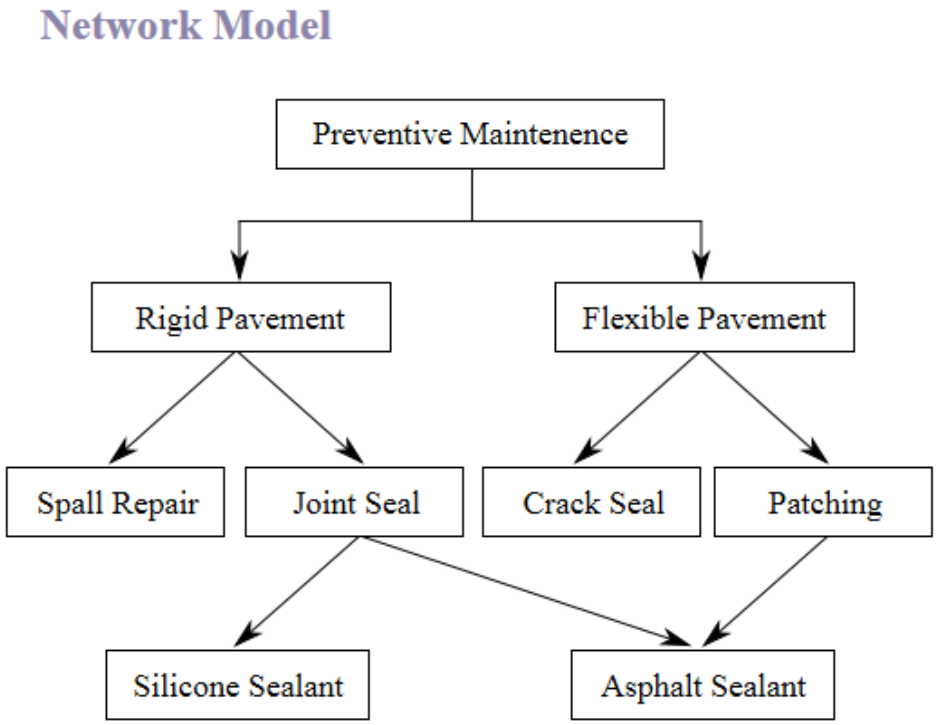


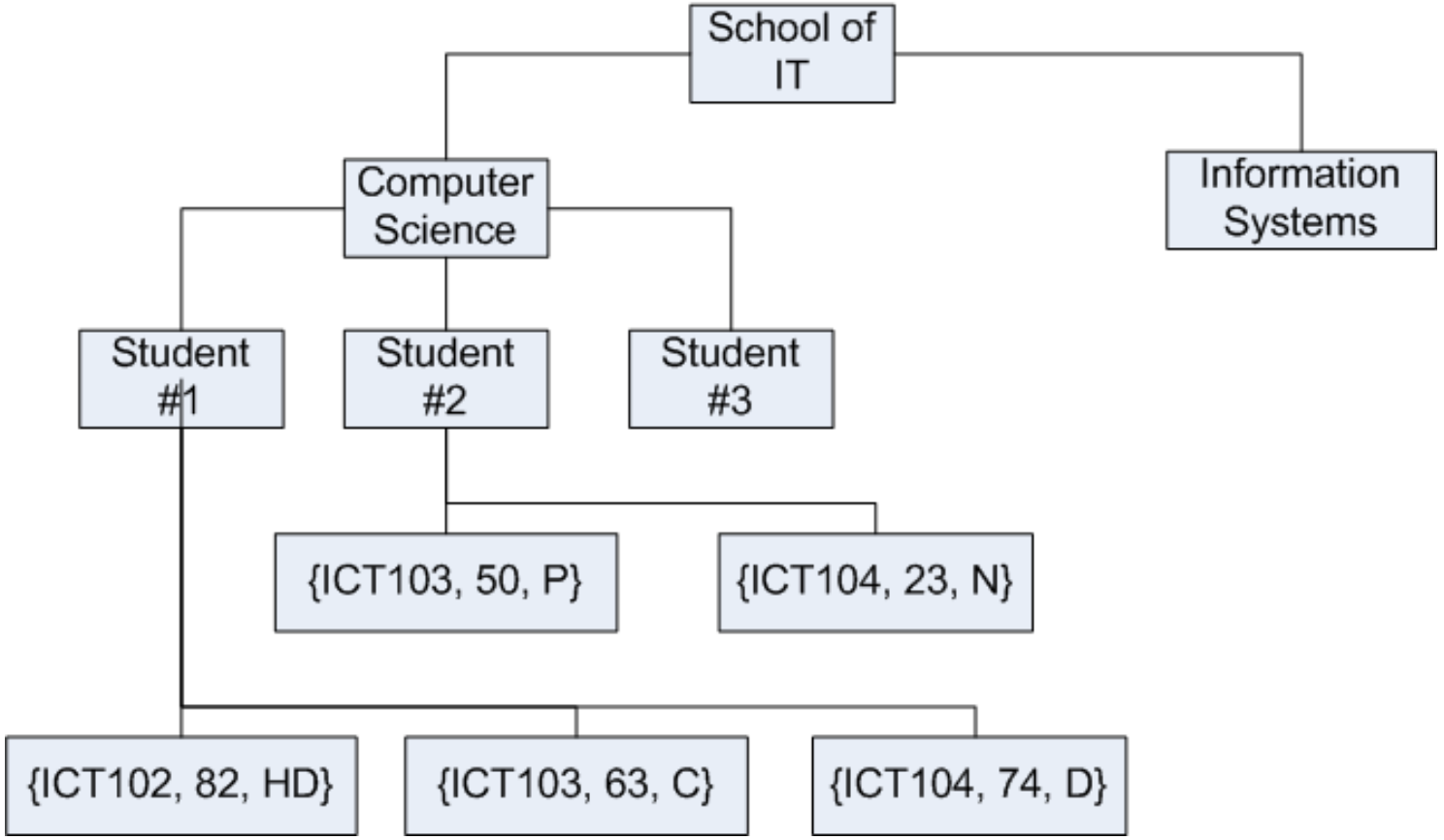
Image from Wikimedia

Operations on navigational databases

The *operators* which allow data to be manipulated in this type of database are *navigational*

- The tree or network needs to be traversed in order to find the required record(s)
- Once the required record(s) is(are) located, records can be inserted, deleted, updated etc.

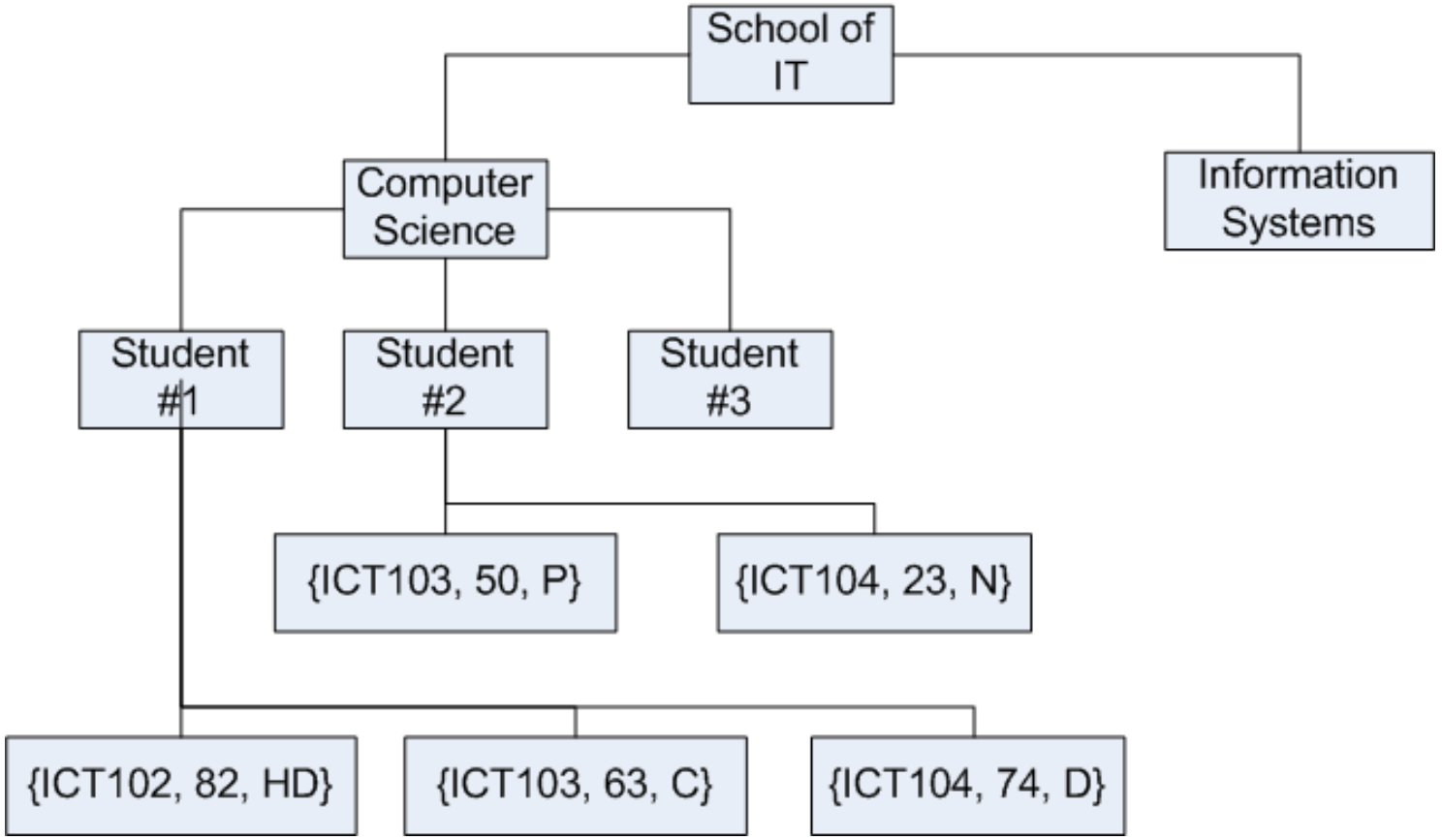
Operations on navigational databases



Find all the results for student #2
Find student #2's result for ICT103

Image by D.Toohey

Operations on navigational databases



Find ALL the results in ICT103… ?

Image by D.Toohey

Difficulties with navigational databases

Inherently inflexible

- The application programmer must know the path to the record of interest in order to retrieve the desired information
- The relationships are static; to search the records in a different way is problematic

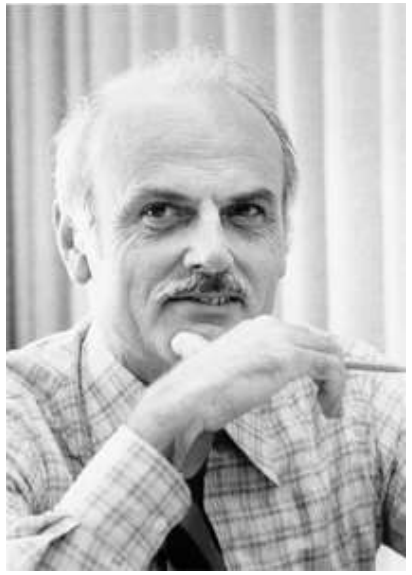
Redundancy

- The possibility of redundant data arises because of the parent-child nature of the hierarchy

The Relational Model

The relational model was created by E.F. Codd in the late 1960's

- RM was developed in response to problems of inflexibility associated with the navigational data models
- Relationships in RM represented through *values*, not predefined links
- This results in a high degree of *data independence*
- Many relational DBMSs today; SQL as standard query language



https://www-03.ibm.com/ibm/history/exhibits/builders/builders_codd.html
https://www-03.ibm.com/ibm/history/ibm100/us/en/icons/reldb/

RM concepts

- Conceptually, a **relation** is a table of values:
  - Each ***tuple*** (row) in the table represents a collection of related data values
  - Each ***attribute*** (column) of the table specifies how to interpret the data values in each row
  - The type of data (or set of allowable values) that can appear in each attribute is called the ***domain***
  - Each tuple is unique and is identified by a ***key***

StudentID	FamilyName	Degree	Major	GPA
12345678	WELLS	BSc	ISD	3.00
12456789	NORBERT	BSc	CS	2.70
23456789	KENDALL	BSc	GT	3.50

RM concepts

EmpNo	FamilyName	GivenName	DeptNo
12345678	Smith	John	5
23456789	Wong	Franklin	2
34567890	Zelaya	Alicia	
45678901	Wallace	Jennifer	2

DeptNo	DeptName
1	Research
2	Admin
3	HQ
5	Youth

- Relations are linked through matching values of *primary keys and foreign keys*
- This gives the relational model great flexibility and the ability to ask ad-hoc queries

Relational Model concepts

We said earlier that a data model has 3 features:

- **Data Structure**

  In the RM, the structure is the relation, made up of tuples, attributes, domains

- **Operators**

  Relational calculus, relational algebra

- **Constraints**

  Keys, Integrity constraints

We'll look at these in more detail in Topics 2 and 4

Object Oriented data model

Databases built on an OO data model appeared in 1980s in response to perceived weaknesses of the relational model:

- Limited constraint representation
- Limited relationship representation
- Simplistic/limited data manipulation languages

Claimed to provide the following advantages:

- Addition of semantic content
- Inheritance
- Reuse

OO databases - disadvantages

- Lack of a standard implementation
- Increased complexity
- By the time it came along there was a large installed base of relational DBMSs

Many commercial DBMS support several (but not a consistent set of) OO features in addition to their relational foundations

Much OO modelling (as you may have done in ICT284) still ends up being implemented in a relational database

Non-relational or NoSQL databases

- Relational databases are very good at managing the data across a large enterprise for online transaction processing
- But relational databases don’t scale up well to massive, distributed environments such as Amazon or Facebook and the challenges of ‘Big Data’
- Non-relational or noSQL (“not only” SQL) databases have arisen to fill these needs
- Aims are simplicity of design, scalability, speed
- No single data model, but various approaches

Non-relational or NoSQL databases

- NoSQL databases are modelled in ways other than tables
- Four main categories:
  - Key-value stores
  - Graph stores
  - Column stores
  - Document stores

(see http://www.jamesserra.com/archive/2015/04/types-of-nosql-databases/)

The take-aways…

- A **data model** defines the structure, constraints, and operations that are possible on an organised set of data
- In order of appearance:
  - Hierarchical
  - Network
  - Relational
  - Object-oriented and object-relational
  - Non-relational
- However, this sequence doesn’t imply replacement – many of the earlier ones are still around

Topic 01: Part 03

Three-level architecture

The ANSI-SPARC Three-Level Architecture

- Developed by ANSI-SPARC in 1975
- Proposal for a general architecture for database systems
- Identifies the three distinct abstraction levels at which data items can be described:
  - External level
  - Conceptual level
  - Internal level

- Although no DBMSs fully adhere to the 3LA, it provides a useful way to understand the functionality of a DBMS

ANSI-SPARC Three-Level Architecture

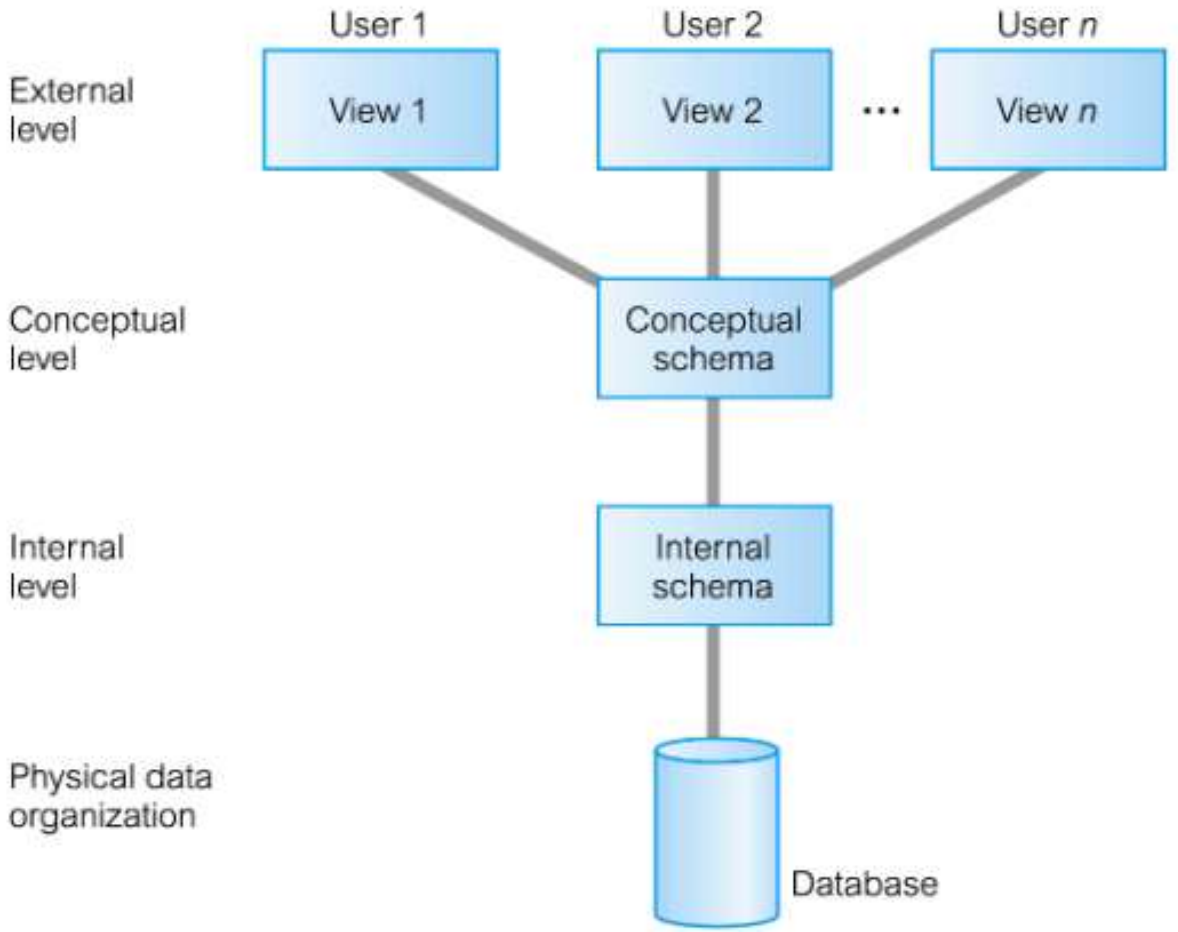


Figure from 'Database Systems: A Practical Guide to Design, Implementation and Management' T. Connolly and C. Begg

Objectives of the 3LA

- Primary objective is to separate each user's view of the database from the way it is *physically* represented
- This is desirable because:
  - Each user should be able to access the same data, but have a view that is relevant to them
  - Users should not have to deal with physical database storage issues such as file structures
  - The DBA should be able to change the database storage structures without affecting the users' views
  - The internal structure of the database should be unaffected by changes to the physical aspects of storage, such as moving to a new storage device
  - The DBA should be able to change the conceptual structure of the database without affecting all users

The External Level

The users' view of the database

- Each user has a view of the 'real world' that is relevant to them
  - E.g., student records system
    - Staff view (MyStudents – see next slide)
    - Student view (MyInfo)
- The user does not need to be aware of other tables, columns etc that are not part of their view
- Allows for different representations of *the same* data

MyStudents…

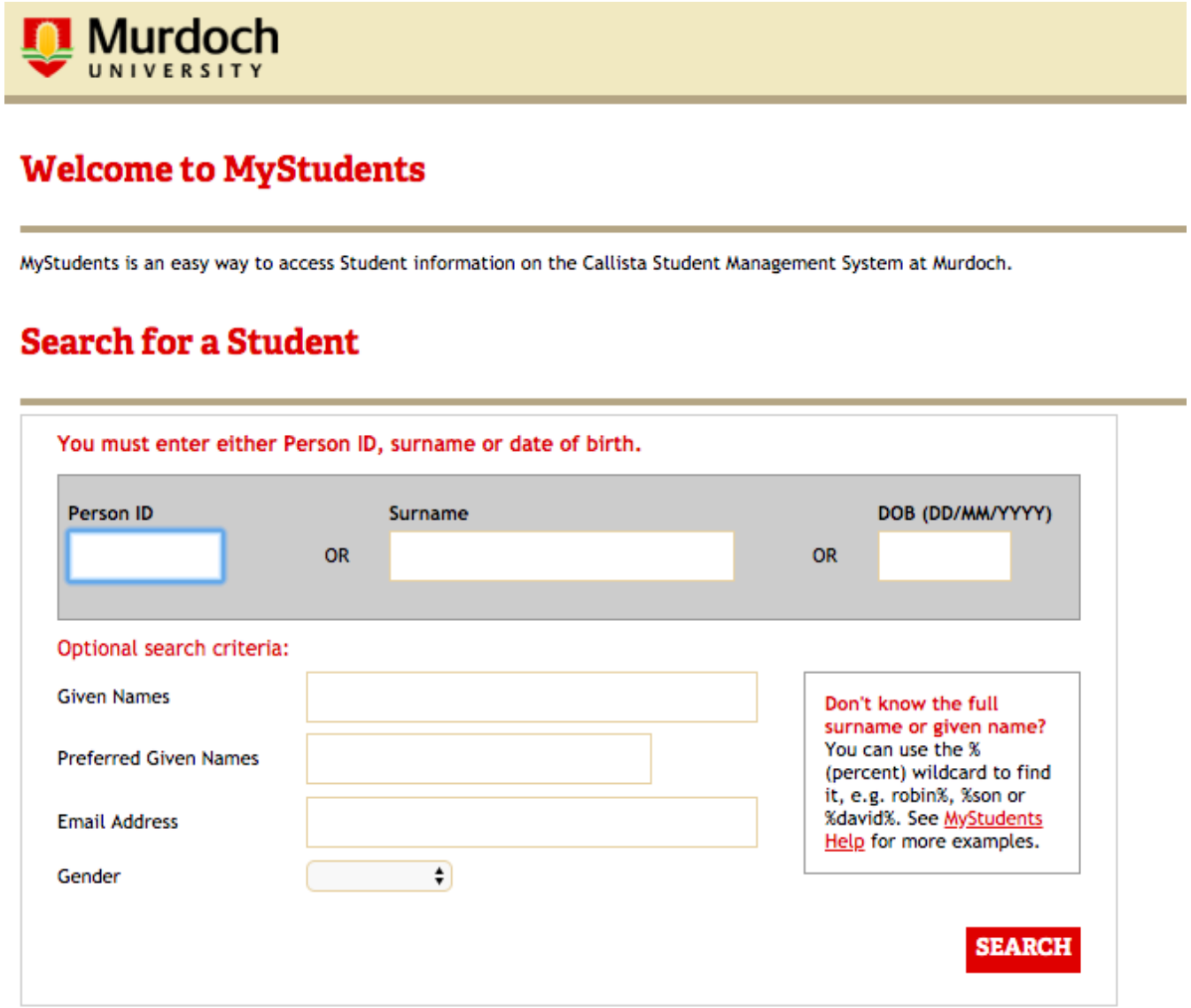


Image by D.Toohey

The Conceptual Level

This level contains the *logical* structure of the database

- Including
  - Entities, their attributes and relationships
  - Constraints
  - Semantic information about the data
  - Security information

It supports the external level in that every view must be derivable from the conceptual level

- It does not include any storage-dependent details (e.g. possibly datatype, but not how the data type is stored)

The Internal Level

Covers the physical implementation of the database to achieve optimal run-time performance and storage space utilisation

- Data structures, file organisations
- Interfaces with the OS

Data independence in the 3LA

Data independence means that changes in the different levels of the 3LA will not affect *higher* levels

- **Logical data independence**
  - The immunity of the **external schemas** to changes in the **conceptual schema**
  - Examples would include addition of new entities, attributes, relationships
- **Physical data independence**
  - Immunity of the **conceptual schema** to changes in the **internal schema**
  - Examples would include modification of indexes, storage devices or file organisations

The take-aways…

- The three level database architecture defines the different levels of abstraction at which the database can be described:
    - External (user)
    - Conceptual
    - Internal
- *Logical data independence* enables changes to the conceptual schema without affecting user views
- *Physical data independence* enables changes to internal schema without affecting logical or conceptual

Topic 01: Part 04

Functions of a DBMS

Some definitions

- **Database**: a shared collection of logically related data and a description of that data
- **Database management system (DBMS)**: the software that allows the user to define, create, manipulate and maintain the database and which provides controlled access to the database e.g. Oracle, MySQL, SQL Server
- **Database application**: program(s) that process some or all of the database for a specific purpose
- **Database system**: the hardware, software, networks, procedures and people involved in using a database for a particular purpose

An example database system:

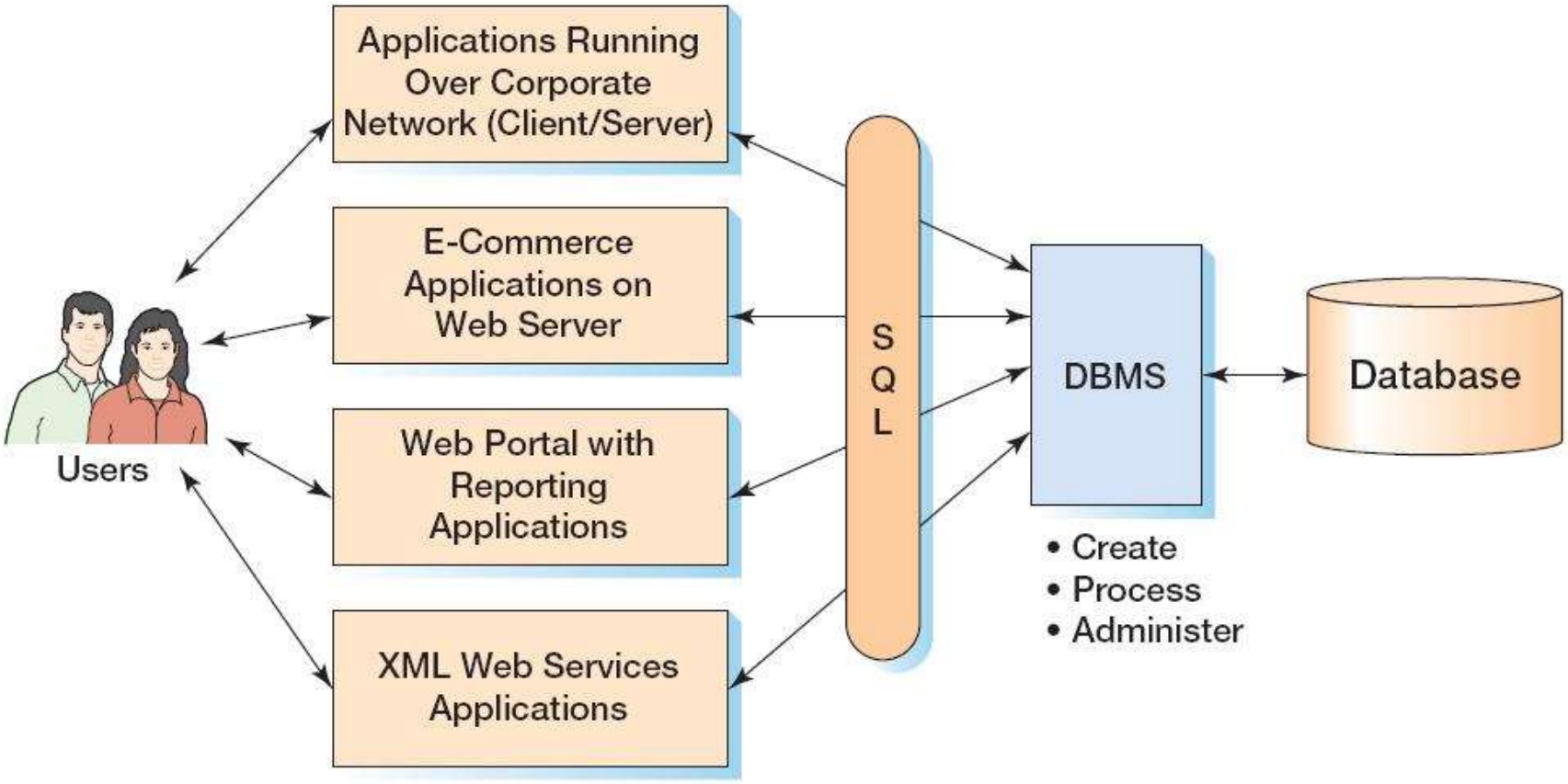


Figure 1.16 in Kroenke, D.M., and Auer, D.J., 2014, Database Processing: Fundamentals, Design and Implementation, 13th Edition, Pearson.

Functions of a DBMS

Codd (1982) listed 8 services that should be provided by any full-scale DBMS:

- Data storage, retrieval, and update
- User accessible catalogue (metadata)
- Transaction support
- Concurrency control services
- Recovery services
- Authorisation services
- Support for data communications
- Integrity services

We will examine most of these using Oracle in the labs

Topic 01: Part 05

Conclusion

What we set out to do…

- In the first topic, we'll discuss what databases are and how they differ from computerised file processing systems.
- This leads us into the concepts of program-data independence and different types of database model.
- We introduce the relational database model, which we will cover in depth in the next topic, and focus on throughout the unit.

Topic Learning Outcomes

- At the completion of this topic, you should be able to:
  - Define what a database is and compare the database approach with the use of file systems to manage data
  - Define the features of a data model and explain several data models including the relational data model
  - Discuss various approaches to database architectures and data independence
  - List and explain the features of a relational DBMS

References

- Berman, U., 2007, "*The Birth of IMS/360*", Retrieved 15th May, 2014 from http://corphist.computerhistory.org/corphist/documents/doc-4b96aca992886.pdf?PHPSESSID=8ff79d8f94a54bc736e1f03fe5b6e012 .
- Date, C.J., 2000, "*An Introduction to Database Systems*", 7th Ed., Addison-Wesley, Reading. [Available in the library]
- Elmasri, R., and Navathe, S.B., 2011, "*Database Systems: Models, Languages, Design, and Application Programming*", 6th Ed., Pearson, Boston. [Available in the library]
- Garcia-Molina, H., Ullman, J. D., and Widom, J., 2009, "*Database Systems: The Complete Book*", 2nd Ed., Pearson Prentice-Hall, Upper Saddle River