Database Systems

**Database Systems : Foundation terms & concepts**

Learning outcomes: at the end of this section the student should be able to

* Explain the differences between a database system and other data processing systems.
* Describe the components of a database system
* Explain the basic architecture of a database system
* Relate the theory of database systems to actual database implementations including SQL and specific database products such as MySql.

Basic terms

* Hardware
* Software

Hardware

Hardware refers to all the physical pieces of equipment (device) that form the computer. You have two main types:

* Processor: main controlling element of the computer e.g. Intel processor.
* Peripherals: any other device of hardware that connects to the processor. Three sub categories,
  + Input device: e.g. keyboard, mouse
  + Output Device: e.g. screen, printer
  + Storage(memory): disk, USB stick

A given computer is made up of a combination of these peripherals as required.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Software: instructions to run the computer hardware. A set of instructions is called a program. There are two main categories of software:

Application software: programs that perform a specific use for the user e.g. accounting, emailing, word processing, databases

System software: programs that run and coordinate the hardware. System software also allows the application programs to run and use the hardware. E.g. An Operating System (OS) is an example of system software. Windows XP, Unix etc are all Operating Systems. You must have an operating system running on a computer if it is to work. Note, all the applications use the services provided by the OS i.e. to store your data. Access, Excel, Word etc all use the OS to store and manage the computer memory.

Processor (with its own memory called Main Memory)

* Hardware

Input e.g. keyboard

Computer Peripherals Storage (secondary) e.g. disk

Output e.g. D: isplay

Systems software

* Software

Application software

People who make programs are programmers or software developers. You are a software developer as you develop database applications (e.g. a student record system)

Computer Data: this can be a slightly confusing term in that everything stored on a computer can be referred to as data, but in reality there are two principle types of data

Programs

Computer Data

Processing data

**What is data in human terms? That is, what data do we store and process?**

Let us take a sentence from earlier. This is a piece of class notes data.

computerdatathiscanbeaslightlyconfusingterminthateverythingstoredonacomputercanbereferredto asdatabutinrealitytherearetwoprincipletypesofdata

How about if the data was like this:

sonraíríomhaireisféidirseoabheithinatéarmabeagánmearbhalligobhfuilgachrudatástóráiltear ríomhaireachurfaoibhráidmarsonraí,achindáiríretádháchineálprionsabalnasonraí

How do humans store and process this data? One process is ‘to make sense of’ the data. How do we do that?

Language; (grammar & vocabulary). It is critical to process the data using the correct language!

**What is the basic Data in a computer system?**

Binary (either a 0 or a 1); nothing else. All data is made up of these two values.

**How do programs make sense of computer data?**

Computers can only store and process data that is in a special form called binary i.e. either a 1 or a 0. So all data, (programs and processing data) is stored in binary form. Why?

Because a mechanical/electrical devices can work with two states (0/1; on/off; +5V / 0 Volts)

Disk 100101000101000101001010100🡪RAM

Note: both directions

Data is stored physically on some **media** or memory: two types, primary/main and secondary. Why are the two required? Primary memory is called RAM; Random Access Memory.

**Why not use all RAM all the time?** **Or, why do we need a second type of memory?**

RAM is volatile storage; data lost if power goes.

As its name suggests RAM needs to be able to access individual memory cells at random to process them; however the size of RAM is limited by the size of the processor (CPU). This is called the word size of the CPU and so computers can be described by terms like its 16 bit or 32 bit. The bigger the word size the more RAM the CPU can use; however RAM is always too small for the vast amount of data people store. So, we always need a second type of storage that has a large capacity.

All data (programs & processing) that needs to be ‘saved’ for long term storage must use a storage medium such as a disk. Disks are an example of what is called 2nd storage.

Computer main memory (RAM) is memory for **processing data. Main Memory(RAM) is termed volatile. Volatile means it needs power to store data**. So when the power is switched off, you lose the data. Hence the need for saving long term data to 2nd storage.

Disk Computer Processor

DATA RAM

Means both directions

RAM is processing memory; power required to manipulate bits to do work. Therefore, long term storage must store data without need of power e.g. a disk using magnetic level to store 0/1 sequences.

**How is binary data understood?**

How is a stream of binary data understandable? How is meaning given to the Data?

The first step is using a defined standard for breaking binary into sections e.g. bytes where each byte can store a piece of data. For character data we apply a standard coding system to allow us to understand that data e.g. ASCII. So Binary byte(base 2) 01000100 = 136 in Base 10; ACSII char 136 = D.

**How is data meaningful?**

It is still not enough just to adopt a standard representation like ASCII. Data stored in a computer (i.e. sequences of binary 1,0s) must be processed to find if it is meaningful or not. Recall that ‘ksdjfsdjksdfjjhdsfjk’ is acceptable text but it is meaningless. A language must be used: a programming language. Just like there are different human languages, so too, there are computer languages and they are not compatible.

So every computer program (of any language) must have a section where it structures the data it will expect to process.

**How do programs/developers ‘make sense’ of binary data?**

They use a program to define some data structure (data definitions) with attributes of the required data types e.g. a record with attributes,

Name Character(20),

Address Character(100),

DateOfBirth Date,

Salary Integer etc. .

Programs not only instruct the CPU to ‘do things’ but it also defines the data values using variables and data types e.g. Count integer, Pay real, Name char etc. So a computer program is able to ‘make sense’ of the physical bits and bytes of storage by linking items of physical data to meaningful items of data i.e. age, count etc. This is called the **data definition** section of a program.

So when executed, the program maintains the **data definitions** and looks up the details each time a variable is referenced e.g. Cin(count) Or Scanf(Count). So the stream of binary data, is divided into blocks(bytes); the program controls what each byte will represent e.g. integer, real, character etc.

Computer users also use the notion of ‘files’ and folders/directories. Computer data is organised into a logical unit called a file; when you have many files you can use a logical organisation such as folders/directories to organise your data files. This ‘file management’ is one of the functions of the Operating System. So, the OS manages data but the application programs define and process.

Are all application programs the same?

No, There are different categories of application programming languages. Modern programming application languages are either procedural or non-procedural. A database system language like SQL is an example of non procedural, while C is procedural (C++ is an object based extension).

3rd generation languages are all

1. general programming languages i.e. that can be used to develop any type of application (an advantage). However, these languages are
2. procedural, well structured
3. complex, the language has many constructs, instructions.
4. may take a long time to learn and
5. take a long time to develop.

Most general purpose programming languages use files to store data; program storage /files are controlled by the Operating System. We say, the program interacts directly with the OS to handle data storage in files. For the rest of this section we will compare this type of system with databases.

<http://en.wikipedia.org/wiki/Third-generation_programming_language>

4th generation languages such as a Database system are

1. non-procedural
2. simpler to learn, basic small set of operations (Insert, Update, Select)
3. require less time to develop but to give these advantages they are restricted to a specific type of application i.e. database application (this is a disadvantage).

So a database system can only manage and share data that is in the form of records and it cannot be used to develop any other type of application. Note reference: Object Oriented (C++) or Web (Java).

Database systems can share and manage data more easily because they radically change the way data is defined and processed in the program. In doing so, it

* maximises something called ‘data independence’ and
* enables data sharing and better controls over data.

**Data independence:**

A 3GL procedural program when it refers to a file in its program code, uses a reference such as

C:\data\byrondata\order.txt

We understand this as, the file ‘Order’ of type ‘.Txt’ in a folder ‘byrondata’ on the C drive of the storage device disk.

The program then just issues an instruction such as

Read (C:\data\byrondata\order.txt, name, address, phone number).

There is no physical detail here, no reference to computer binary data, no type of storage e.g. USB flash or Internal Disk, no manufacturer name etc. This is an example of what we call **Physical Data independence**.

However you should note that the code above is still dependent on the logical Drive, path and file type (i.e. .txt). We cannot move the file Order, or change its type (to let’s say a .doc format) without editing the code and recompiling. This is an example of **Logical Data Dependence.**

The program only needs to define the variables it needs and define a suitable data type for them.

Therefore, for the above e.g. the program must first define

Name varchar(20), Address Varchar (100), Phone\_number integer

This defining of data variables and allocating data types is called ‘data definition’.

**How is data independence implemented?**

If an application program doesn’t deal in physical details, how does it actually work on the computer? Since computers can only be controlled by software programs, if the application program doesn’t deal with the physical details of the computer that the program runs on, then some other piece of software must.

The Operating System(OS) is a set of **system programs** that enable use of the physical hardware on the computer. We can think of the OS as a layer of programs that separate the user **application program** from the physical details of the computing machine. So traditional 3GL application programs use the Operating System to give them physical data independence.

Standard OS/file system & procedural programming language system

Read (name, phone\_no) Program User independent of implementation

Level allows for some independence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Disk Read,bytes read for

Varchar and integer OS Physical implementation hidden from

User. Internal.

Disk

**What are the advantages of Data Independence**

Data independence allows the application program respond to changes to any physical element within the overall system. For example, if the entire computer was replaced with a new, faster one. Our program still works because it is physically independent. We do not need to edit the program code in any way to work with the new hardware (physical computer).

But the program is still data dependent, in that it is logically dependent on the drive organisation, the folder/directory structure and the file type (.txt). If we move the file, or want to change the type of file to say .doc, we cannot do so without modifying the program.

**Traditional procedural (file based) programs do not have logical data independence**.

In a database system, the language is not only physically independent but also logically independent. Structured Query Language (SQL) is the industry standard database language. When we have an SQL program (query) such as

Select \* Note: \* means ‘all column’

From Students.

T**his SQL program (query) is both logically and physically independent**.

**Where is data independence exhibited in the above SQL program (query)?**

The program has no data definition (or so it seems) and it has no unnecessary logical details (or dependencies). The minimum the user needs to do is specify the name(s) of the tables it wants to process.

So, if the SQL non procedural program isn’t handling the link to the drive: folder and file, who is?

Remember that everything in computing is controlled by software (programs), so if one piece of software isn’t performing a critical function then some other piece of software is. So, besides the database SQL Select program, what other pieces of software are in the systems? OS, DBMS? Which one manages the link from data definition to storage out on the disk? See next page.

**A Database Management System** (DBMS) can be viewed as an extra layer of software that separates the user from the database itself. An arrow ( ) in the diagram indicated one program communicates with another.

The DBMS is a collection of programs that provide communication & data services to the SQL programmers. The DBMS controls access to the database, and communicates with the Operating System that in turn controls the physical storage devices of the computer hardware. MySQL, Oracle, SQLServer are all DBMS, just like Windows, Unix are examples of Operating Systems.

A Database System usually refers to a DBMS and a bank of data records (database).

**USER SQL Program**

**DBMS**

**Operating System**

**DATABASE**

We can say that a database system maximises the design goal of both logical and physical.

data independence.

Example of a complex SQL query program using 3 tables of data for students, module and enrollments.

SELECT Avg(Mark)

FROM Student, Module, Enrols

WHERE Student.CITNo = Enrols.CITNo AND Module.MId = Enrols.MId AND Name = ‘Byron Treacy’

GROUP BY MId

ORDER BY Enrol.Date

So far, we have examined the code above using notions such as Data independence.

What else might we notice for the SQL program?

Note the complex processing such as sorting (Order by); reorganising results into logical groups (Group By), and performing common function such as calculating average (AVG).

The question is where are the programming instructions that perform the actual ‘Sorting’, or the ‘grouping’?

If you understand what we’ve discussed so far, you should be able to answer this question. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The SQL programmer just indicates they want certain logical processing; they do not contain the procedure for how to do that work. That is why, SQL is easier to learn. The work is now being done by the Database Management software (DBMS). The SQL programmer is independent of exactly how that processing is implemented. This is why SQL is called a **non procedural programming language, and why it is both logically and physically independent.**

To isolate a data set, we generally need to specify the qualifying parameters/criteria e.g. patients with flu. We therefore need to specify **what data is required rather than how to get it** i.e. we don’t specify what file to open, what structure the data in the file is, how to step through the entire file looking for the exact match on the qualifying criteria etc.

This is the difference between non-procedural and procedural (file based) data manipulation. Standard programming languages e.g. C are procedural, database languages are non-procedural.

Database languages are commonly termed ‘query languages’ due to their emphasis on data retrieval

Although query seems to indicate retrieval operations only, SQL also supports the other database operations of updating, deleting etc. In addition SQL also provides the data definition facilities ( define tables, record formats etc.) as well as data reorganisation.

**Sharing data**

**Are there any particular problems if the data is to be shared between multiple users?**

Traditional procedural program is where each program having its own private data definition and files. So, think of a company with a computer in the Sales office running a program for its Sales staff. There is also a computer (maybe in a separate office) running a Management program maintaining the Contract information.

These two programs may be programmed independently of one another; possibly by different programmers. The data is not shared or integrated i.e. stored in separate files.

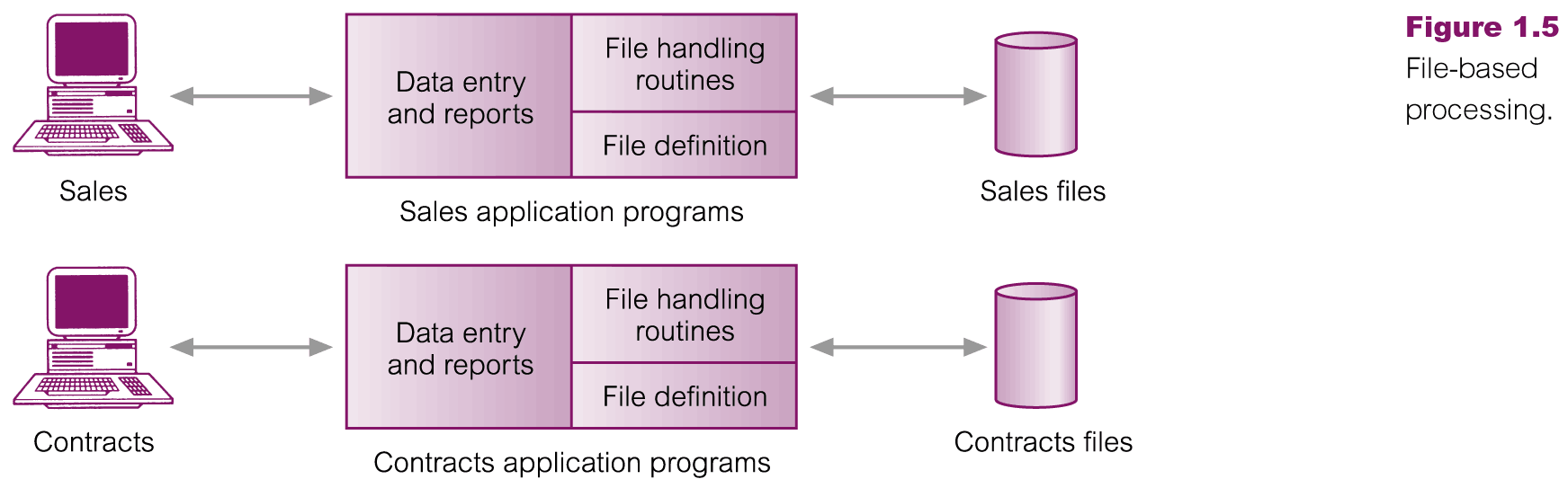
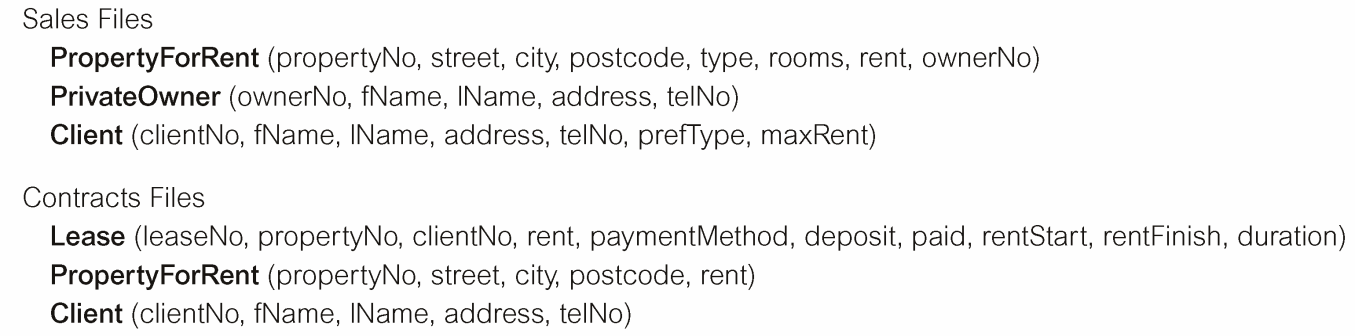


Diagram is from Connolly Begg Database Systems text book

Notice the duplication in the data between Sales and Contracts. But more importantly note the incompatibilities/differences between the data used by the different applications. List some?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**But if data is to be shared, it must have a common definition to all users/applications.**

How can we get every user to use the same data definition? We cannot allow different users have different interpretations of the data, e.g. if one program thinks Name varchar(10) and another

defines Name as varchar(20), one will read the disk data differently that the other.

Solution:

In a database system; the software architecture of the DBMS splits data definition away from processing. In SQL you now get two

The two basic sub-components of the DBMS language are

1. **data definition**(DDL) : of record structures and attribute data types
2. **data manipulation**(DML) : insert/update/select etc.

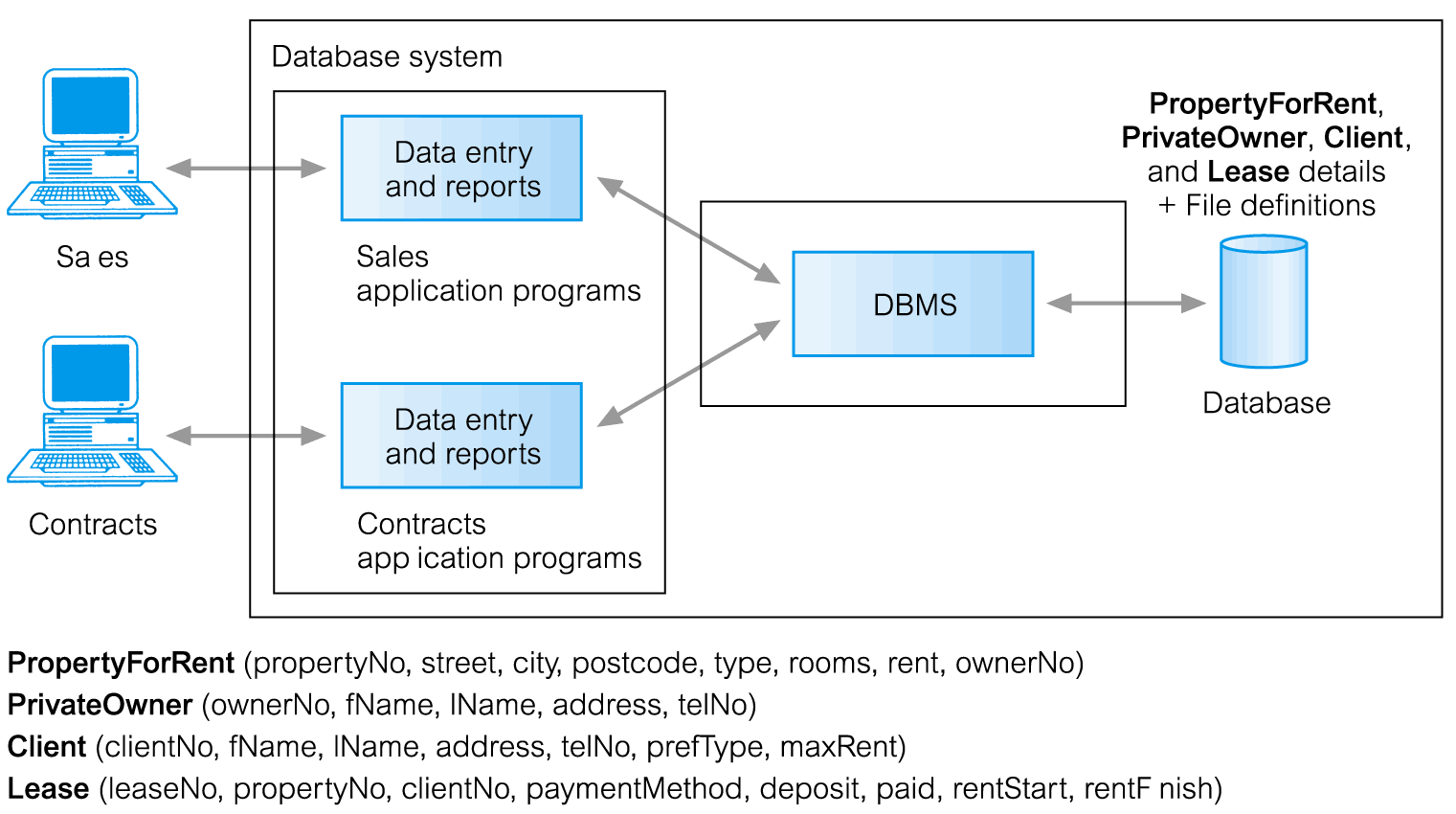
The DDL is essentially similar to what is required by any programming language with some statements to specify the necessary databases, tables, fields and their datatype/formats etc. Note however that the structure of data is limited (table). So far we have introduced the concept of a table as being the only structure that is requires to store the two basic types of data that are of concern to a database system i.e. objects and associations.

The important thing to note is the separation of the definition from the instructions that manipulate/process the data. Recall shared data integration and data independence from earlier pages. Without these, a programming language cannot split these functions. This also forms the basis of a move to 4th Generation languages.

So in a database system, some user must first design the database; some expertise is required. Once designed, the database is implemented using the DDL; defined just once, not by each user. The data definition is then controlled and shared by any user that wants to manipulate the database.

**Why use a database?**

A database is design and efficient for any application that is restricted to managing data in the form of records; where there are multiple users accessing the database and sharing the data.



**Data in the database can be integrated**.

The Sales and Contracts applications above access the same data tables even though they may only know of (and use) a subset of attributes in the integrated database. The DBMS is able to integrate the data so that diverse applications can transparently access the single shared data resource even though individually they use different subsets of that data.

Example:

**Employee ( Emp-No, Prsi-No, Address, HomePhone, WorkPhone, Room#, TaxBand, Salary)**

**Integration** means that there is one instance of an overall logical description of the Object, e.g. Employee in this case, but different applications can have different views of this object e.g. an application in Finance may only use and be aware of certain attributes

Employee (Emp-No, Prsi-No, TaxBand, Salary)

whereas an application in Administration only use and are aware of attributes

Employee (Emp-No, Address, HomePhone, WorkPhone, Room#)

Superficially, these look like different objects; and in procedural languages they would probably be stored in separate (unconnected) files (Note: with data on 1 employee stored and usually replicated in different places). However, in a database, integration & sharing of data enables

* control of integrity (data correctness) and the
* introduction of standards e.g. in the formats used, the names used for tables and attributes.

Integrity control is a major requirement in a large organisation and is therefore an advantage of using a database system. Integrity controls do not eliminate all redundant data (copies of data) but allows control over the consistency/correctness of that redundant data. Integrity controls also maintain the correctness of values allowed into the database e.g. Salary > 0 and Salary <500,000.

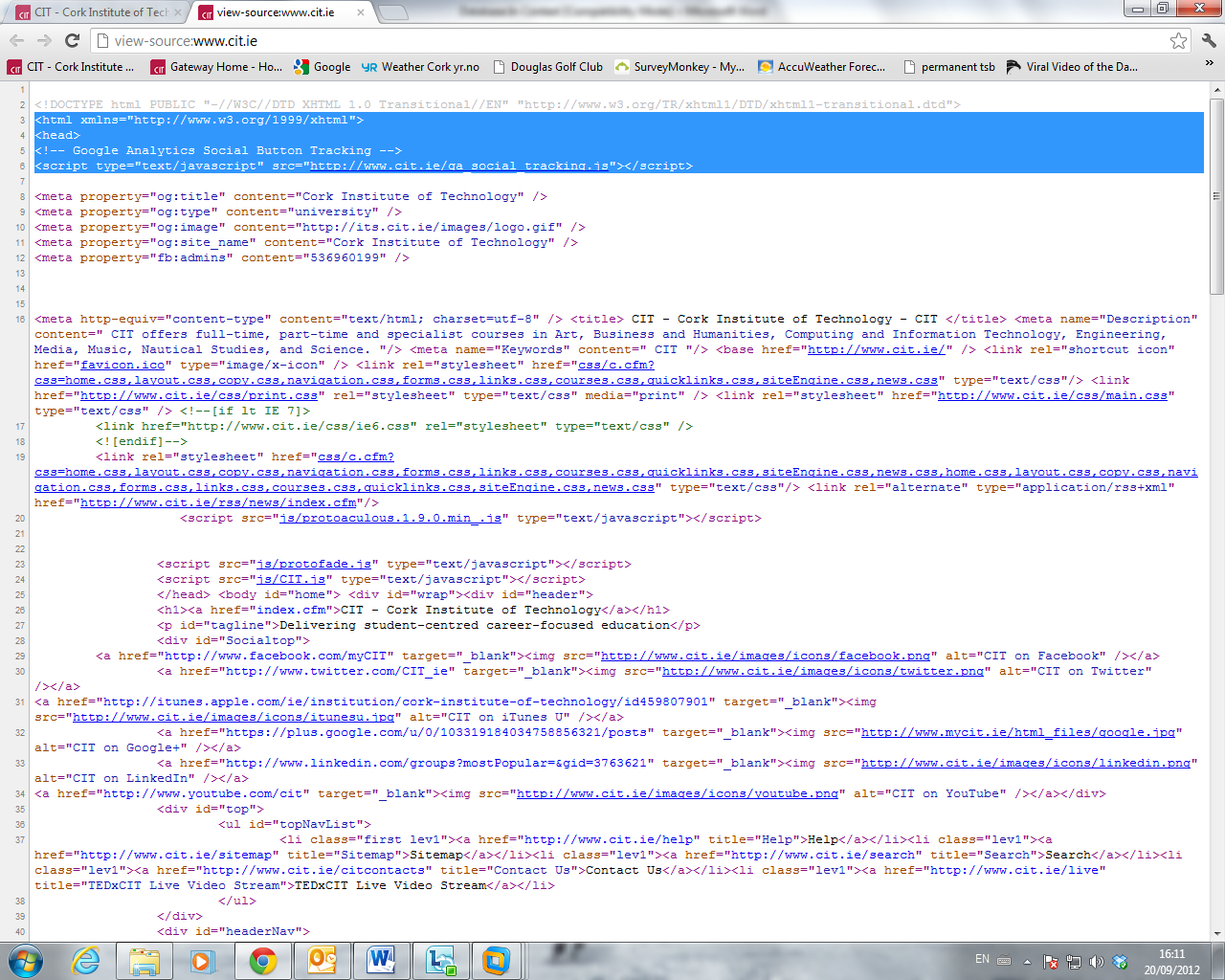
**Restricting the data structure to a table. What is the impact (or usefulness)?**

Restricting the data structure to a table enables removing its definition from the end user, and as a result allows the system to perform many functions. If a table (variable) is named in an end user program, the system (DBMS program) can control query processing & optimisation, correctness (data integrity), concurrency, security and recovery without the end user being aware of the amount of work being performed in the background by the system. These form the basic elements of a database management system.

**But also note that there is no need for output formatting statements in the database language. The output is always a table (grid) output and the DBMS always knows the data definition required for output. The user operations can now be restricted to simple Insert, Update, Delete, Select of rows in a table. This also enables the move from procedural (3GL) to non procedural language SQL (4GL).**

In other words, a general purpose, 3GL procedural language must perform explicit instructions to handle (retrieve, format & display) what-ever output is required by the application.

Note if you point, right click, view page source on any web page, you get an idea of how much output formating instructions are required in some cases.



**The basic elements of a database management system**

* Data management: data definitions, optimisation, data dictionary.
* Data Integrity e.g. balance > 0;
* Security e.g. user ‘Treacy’ can only access table ‘Students’ and
* Recovery e.g. backup every 10 mins etc.
* Concurrency, control over multiple users accessing the shared data

These controls must be managed by a knowledgeable user/designer called a Database Administrator (DBA).

**Two important areas of data management provided by the DBMS are the control of**

1. **Access Mechanisms**
2. **Storage Structure**

**Both of these are important for optimising the Database System for efficient processing.**

**What is meant by Access Mechanism?**

Access mechanism means the method or way in which data is located or retrieved by the system/user e.g. an index (similar to a book index). This requires complex processing and the techniques are dealt with in a full section of this course later; Note that index implementation and maintenance; sort algorithm etc would require complex programming but the user in DBS is independent of this complex detail i.e. they ask for data to be sorted but the system does the sort implementation. Note that the Order By clause in SQL allows the SQL code to be independent of the implemented sort algorithm to do the sorting.

**What is meant by Storage Structure?**

A given stored file can be physically stored on the storage medium (disk) in a number of different ways. It may or may not be stored in a sequence based on the values of some field e.g. alphabetic order of name field. It may be organised (stored physically) to allow efficient lookup of individual records based on their key value.

**An advantage of Data Independence is that the DBMS are programmed to provide the SQL programmers with Access technique and Storage structure independence. The DBA uses the DBMS to manage these functions**

**What benefit is this?**

The Database Administrator (DBA) must be able to **change the storage structure or access technique** in response to

- changing requirements : for instance a change in the way the data is processed e.g. users may start processing the data on a particular field e.g. Zip code or phone area code instead of City, or

- new developments : for instance faster or more reliable storage devices or access techniques.

If you were to decide not to use a Database System, then you would either

1. Have to program these services yourself, which would be a very complex programming task.
2. Do without those services i.e. your application may not need them

Learning objectives and exam style questions: be able to describe the following

* Do you understand the role of the Operating System in computing applications?
* Can you explain the difference between standard procedural programming Systems and Databases?
* What are the disadvantages of programming using a traditional (3GL) procedural language that uses an Op Sys for storage?
* Explain the terms logical and physical data independence using example(s)
* Restricting the data structure to one type is critical to the development of DBS. Discuss.
* What is meant by the term data integration; use an example?
* Sharing data causes problems for traditional programming systems, explain.
* Data consistency can be an issue for large data applications, discuss.