

2. INFORMATION ACCESS TO STRUCTURED / DATABASES

This section of the course is meant to make students familiar with the environment in which databases are designed and developed, and an appreciation of the problems associated with the area. A proper DBMS course would take 25 hours or more, and we have less than two hours, so all I expect to cover is an overview and at the end of this I hope that those who have not covered databases formally in the past, will be able to gauge what a database can provide and what it cannot do. For the rest of you, it's a re-run of already familiar materials.

What is a database ?

- A software package that runs on a mainframe, server, desktop, or mobile device which at least provides fast access to structured data. Databases also do much more than this. Popular databases include ORACLE (the big one), Sybase, DB2, SQL Server, etc. The third largest software company in the world is a database company. Databases allow data to be inserted, deleted, modified or retrieved.

Why use a database in the first place ... what do they give us ?

- Scale – if a database can be configured to manage a small amount/volume of data effectively and efficiently, then it is usually trivial to zoom up to huge data volumes – need to have a minimal volume of data anyway in order to justify the overhead;
- Speed – databases provide fast access, often the fastest possible access, to data for certain types of retrieval requests. Databases do query optimisation which means they can do clever things to improve speed of access;
- Flexibility – databases can be used to query underlying data in an enormous number of possible ways – not all these are efficient, but all can be executed;
- Concurrency – databases control concurrent use (simultaneous read and write) by large numbers of users, without application developers having to worry about such problems;
- Backup and recovery – data can be backed up and automatically regenerated or recovered in the event of catastrophe or minor errors;
- Transaction management – sometimes we want to do a single logical operation which actually involves more than one physical operation and databases can regard a sequence of separate physical operations as one atomic event ... either they all happen, or none happen. This adds robustness in the event of failure in mid-execution or offers the possibility of controlled rollback;
- Distributed databases – databases can support distributed processing and that's growing in importance;

- Data integrity can be maintained and a database can be prevented from having its data resources going into an invalid or impossible status ... this is difficult to describe except by example (see later);

Overarching all of this, using a DBMS makes the development of applications faster, less prone to errors, and more flexible. The downside, is that a database costs resources (CPU, memory, disk) and sometimes money as well.

An indirect advantage of databases is that databases formalise vague and imprecise information organisation by forcing the development of a database schema which is a map of data organisation. That forces us to have to think about what we're doing and what we want from our data.

The relational model is the *de facto* standard model for building databases even though it has been around forever.

The important concepts in this are:



- Table
- Tuple / row
- Column / attribute
- Primary key
- Foreign key
- Referential integrity
- Indexes and index files and fast access
- SQL
- ER diagrams
- Joins

Lets explain the terms with a simple (well-worn, sorry !) worked example of suppliers, products and shipments

Suppliers

S#	S-Name

Products

P#	P-Name

Shipments

S#	P#	Volume

Suppliers are ... DELL, Compaq, IBM, HP, Sony
 Products are ... Desktop, Notebook, Server, Mainframe
 Shipments are ...

What columns would we create indexes on ?
 What kinds of questions can we ask, apart from the obvious table lookup?

What volume of shipments come from supplier number X	SELECT SUM(VOLUME) FROM SHIPMENTS WHERE S#=X;
How many shipments come from supplied number X	SELECT COUNT(*) FROM SHIPMENTS WHERE S#=X;
How many shipments come from any suppliers in IRELAND	SELECT COUNT(8) FROM SHIPMENTS, SUPPLIERS WHERE SHIPMENTS.S#=SUPPLIERS.S# AND SUPPLIERS.COUNTRY = "IRELAND";

Finally, once built, a DBMS schema is not forever, but can be altered and added to – but rarely taken away from – so it can be an evolving work in progress.