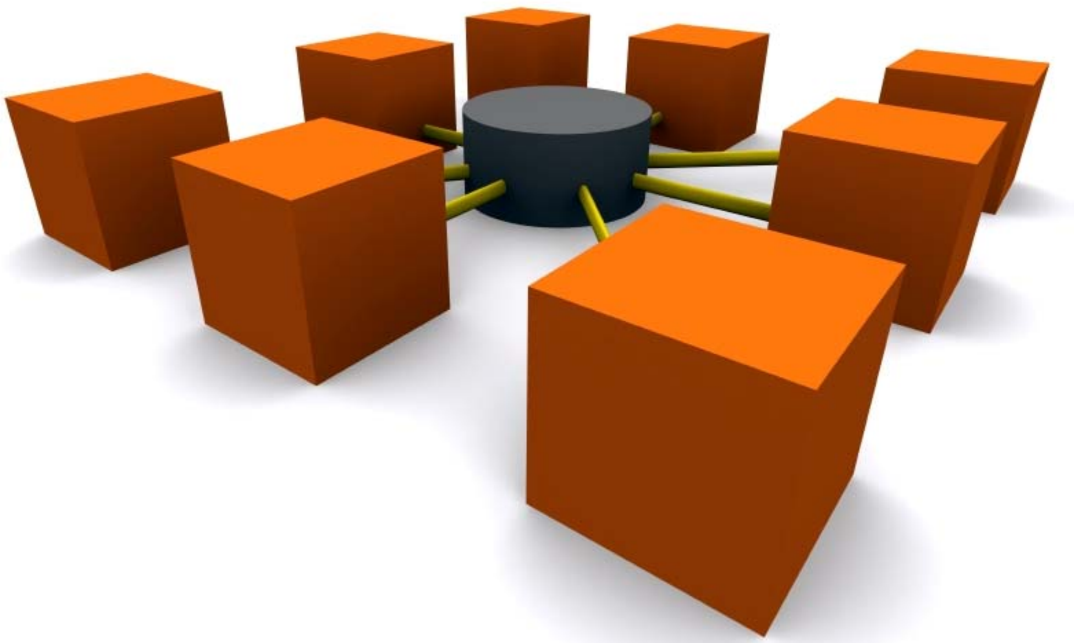
Department of Computer, Communication and Interactive Systems



INTRODUCTION TO DATABASE DEVELOPMENT

M1G505190

11/10/11 JP

1. Databases and database design

What is a database?

Any collection of data can be described as a database, whether it is computerised or not. For example, card-index databases were widely used in libraries long before computer systems became available. Computerised database systems are now very commonplace.

A well-designed computer database system can make access to data fast, and allows many questions to be answered about the data. For example, when you order a book from an online store website, the website can often tell you about the books which have most often been bought by customers who have bought the one you have just chosen. This is possible because the website can access a large database holding information about all the store's books and customers.

Databases play an important part in all our everyday lives. Information is stored in a database every time we use a bank account, book a travel ticket, make an appointment with a doctor, and so on.

Databases vs. Database Management Systems

A database is simply the collection of data which you need to store, for example a list of customers. To actually store the data, and to do anything useful with it, you need software known as a Database Management System (DBMS). A DBMS controls the way the data is stored on the computer, and provides ways of getting data in and out of the system.

Relational and other database systems

The way in which data is organised for storage in a database is known as the **data model**. Early computer databases developed in the 1960's used a hierarchical model, rather like the way files and folders are still organised in modern computer file systems.

Most data does not fit very well into a simple hierarchy. Think of a book store, for example: the database needs to store information on books, authors, publishers, customers, addresses, orders, invoices, and probably much more. Which of these would be at the top of the hierarchy (like the root of the C: drive of a Windows PC)?

Relatively complex data like this is better handled with the **relational model**. Many databases nowadays are relational databases, and you will find out about and work with the relational model throughout this module. A database management system which uses the relational model is called an **RDBMS** (relational database management system).

NOTE

Relational databases are **not new**. The model was devised by Dr Edgar Codd around 1970, and has been used in the most popular commercial database systems for many years.

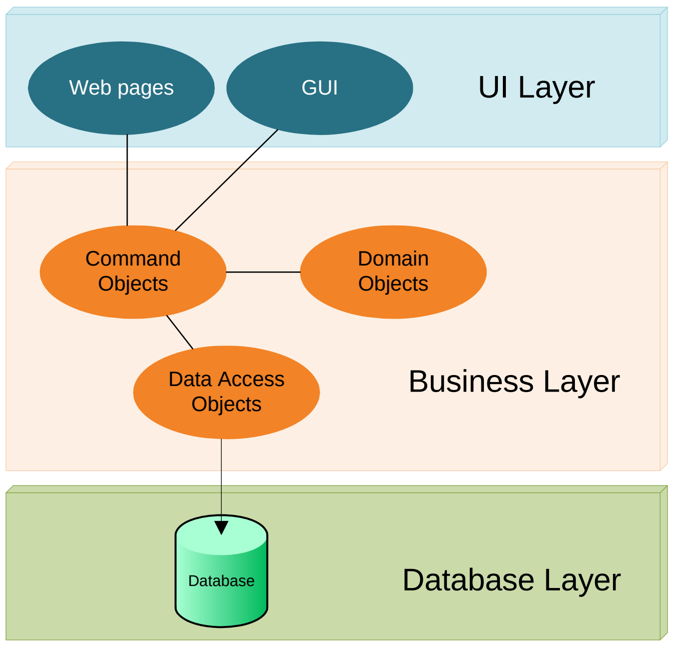
They are also **not the only type of database**. In the early years of relational databases there was competition from databases which used the **network model**, which was related to the hierarchical model. Relational databases became dominant largely due to the development and wide acceptance of a standard language for retrieving data, SQL. In more recent years, the growth of object-oriented programming languages prompted development of **object-oriented databases**, in which the data model closely resembles the models used in developing object-oriented programs. These databases have advantages when it comes to storing very complex data used in object-oriented applications. They continue to be used very successfully in some specialised applications, but have not succeeded in replacing relational databases in the mainstream. **XML databases**, which store data in XML form, are also used in some applications.

Very recently, companies such as Google and Facebook have come up against limitations in using relational databases in web applications with huge numbers of users, and have contributed to the development of a diverse new set of specialised databases. These are collectively known as **NoSQL databases** since the main thing they have in common is that they do not use the relational model or SQL. It remains to be seen what impact these will turn out to have.

Databases and enterprise information systems

In your first year modules you are looking at different computer technologies which can be used together to implement a system which can meet the needs of a large company or organisation (this kind of system is often called an enterprise information system). So how does the database or DBMS fit in?

As you may have seen (in Intro to Comp Sys Dev), the **database layer** provides the system with the ability to store data permanently and to retrieve and update data as required.

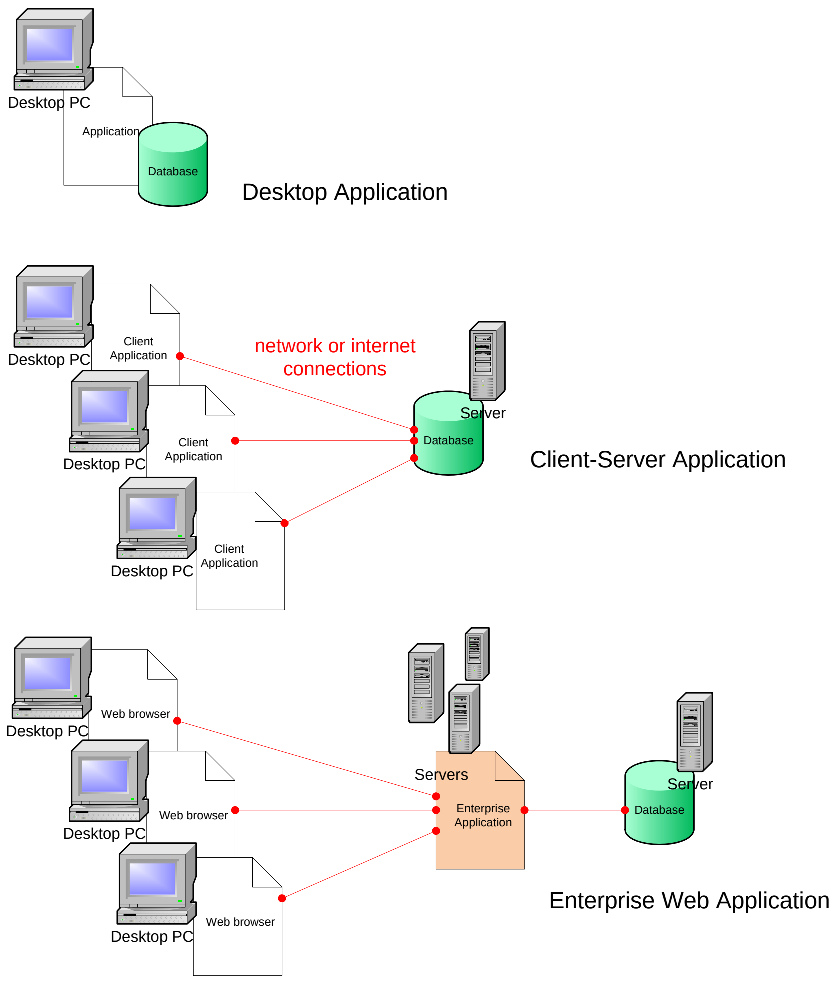


Enterprise information system architecture

In an enterprise system, other components will usually interact with the database. For example, a web page might provide the user interface for entering data, while an application running at the business layer might process the entered data in some way and hand it on to the database for storage.

Database servers

In a small desktop application the database might be part of the application or stored on the same computer as the application, where it can be accessed through a local server or simply by opening a file. Where an application may be used by many people at the same time, the database is usually on a server which is accessed over a network by client applications. In enterprise systems, clients (often web browsers) communicate with an application server which in turn accesses the database, which may or may not be on a separate server.

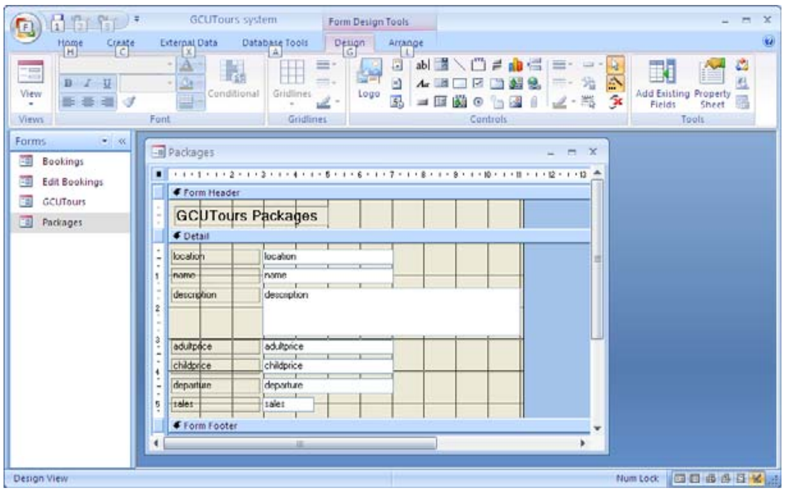


Database servers

RDBMS tools

Most RDBMSs include tools to create complete application, for example:

- **form designers** – to allow data entry forms to be created for the user interface
- **report designers** – to present data to the user
- **stored procedures** – to perform processing of data according to business rules



Designing a data entry form in Microsoft Access 2007

When designing an application you have the choice of using your RDBMS and its tools for everything, or to use the RDBMS as a component and use other tools and programming languages to create the other components. A web application like the GCUTours case study uses the second of these approaches – data is entered and submitted through web pages. In this module you will look briefly at how to create web pages which allow a user to interact with data in a database.

The language of relational databases – SQL

If you develop applications which use relational databases, you will almost certainly need to use **SQL** (it stands for Structured Query Language, and you can say it either by spelling out the letters or as the word 'sequel'). This is the language which is used to define queries – a query is a request to a DBMS for some specific information, Relational databases are sometimes referred to as SQL databases.

SQL queries can be quite easy to understand. For example, the following query finds the last name of all the customers in a database:

```
SELECT LastName FROM Customers;
```

SQL can also be used to add, update or delete data, and to build the database in the first place. You will learn much more about SQL during this module.

NOTE

SQL is supposed to be a standard language which is supported by all RDBMSs. In fact, you need to be careful because there are some important differences between the versions of SQL used by different systems.

Popular relational database systems

There are many RDBMSs available to meet the needs of different applications. Some are suitable for large systems with many users, such as banking systems, while others are simple and compact and designed for very small systems which might run on mobile devices. Some are very expensive, while others are open-source and can be used for free. Here is a very small selection of popular systems:

- **Microsoft Access:** aimed at small businesses, and useful for desktop applications and systems with a small number of users. Sometimes used in small web sites.

- **Microsoft SQL Server:** scalable and secure, and widely used by large organisations. Other similar databases include **Oracle** and **IBM DB2**.

- **MySQL:** open-source and quite powerful, widely used in web sites.

- **Microsoft SQL Server Compact:** a compact DBMS, suitable for mobile devices in particular. Other similar databases include **JavaDB** and **SQLite**.

In this module you will use SQL Server Compact, which creates compact databases which can easily be upgraded to SQL Server if required. You will use a tool called Microsoft WebMatrix which helps you to create and work with SQL Server Compact databases.

Designing a database

A well-designed database allows the best use to be made of the available data, and helps to make sure that the data stored is accurate and consistent. A poorly designed database can do just the opposite: it can be difficult or impossible to get the information you require from it, and can let inconsistencies creep into the data.

What do we mean by inconsistencies? It would, for example, be inconsistent to store a booking without storing the details of the customer making the booking. With careful design, we can make sure the database won’t allow this to happen.

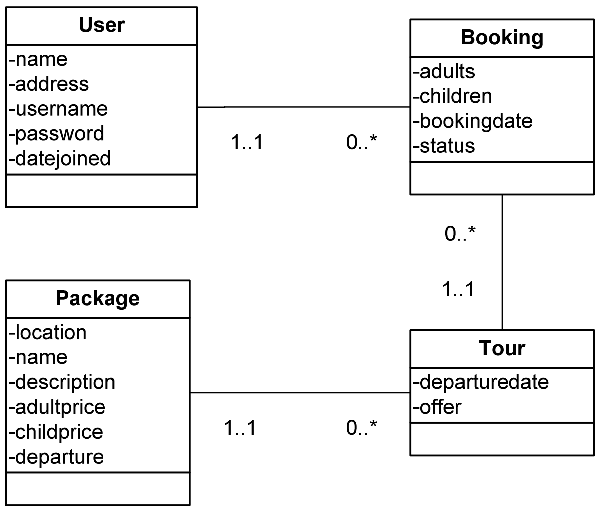
The key steps in the process of creating a database are:

- Determining the intended uses of the database
- Creating a data model
- Implementing the database

The data model

You can use UML diagrams to model the **use cases** of a system and the **classes** representing the entities in the system. The domain classes in a class diagram represent a **domain model** which will support the use cases. The **data model** consists of those domain classes which represent entities we need to store permanently. The data model will form the basis for implementing the database.

The class diagram below shows some of the domain classes for the GCUTours system. Remember that the class diagram shows the **entities**, their **attributes** and the **relationships** between them.



NOTE

We are using object-oriented techniques with UML to design our data model. There are other methods which are also commonly used in database design. One widely used method is called **Entity Relationship Modelling (ERM)**, which represents the data model as an **Entity Relationship Diagram (ERD)**.

From data model to relational database

Before you can implement the database, you need to consider how the data model can be represented in a specific RDBMS, such as SQL Server or Oracle. This requires some further design.

RDBMS software has specific ways of representing and enforcing the entities, attributes and relationships in the data model. For example, a data model entity is represented as a **table** in the relational database.

NOTE

Why do we want to have different ways of representing the same model?

In enterprise applications, it is quite common to represent the data in two different parts of the system, for different purposes:

in the business layer, for example as Java classes and objects – here the system keeps the data which it is currently using in the computer's memory, and uses methods in the Java classes to do something useful with the data, such as creating a new booking, or perhaps calculating the cost of a booking

in the database layer, in a relational database – this is where all the system's data is stored permanently. Queries are used to get the data the system needs for it to carry out a particular action, and any newly created or updated data is stored for later use.

The system needs to **map** data from database tables to classes in the business tier. In this module, though, we are only looking at how to create the database.

The following table summarises the ways of representing data model features in an RDBMS. You will learn more about these soon.

Feature in data model	Representation in RDBMS
Class (or entity)	Add a table
Attribute	Add a field (or column) with an appropriate data type to the table
Object	Add a row of data to the table
Relationship	Add a foreign key – a field containing a reference to a particular row of the related table