LEVEL 5

DATABASE DESIGN AND DEVELOPMENT

Lecturer Guide

Modification History

Version	Date	Revision Description
V1.0	October 2011	For release
V1.1	November 2015	Assessment Methodology Updated

Published by: NCC Education Limited, The Towers, Towers Business Park, Wilmslow Road, Didsbury, Manchester M20 2EZ, UK.

Tel: +44 (0) 161 438 6200 Fax: +44 (0) 161 438 6240 Email: info@nccedu.com
http://www.nccedu.com

CONTENTS

1. Module Overview and Objectives

This module builds on a basic understanding of database concepts to develop students' skills in the design and development of databases and database management systems, as well as investigating enterprise applications of databases.

2. Learning Outcomes and Assessment Criteria

Learning Outcomes; The Learner will:	**Assessment Criteria;** The Learner can:
1. Understand the enterprise application of database systems	1.1 Summarise the common use of distributed database management systems 1.2 Explain the meaning of the term disributed database management system 1.3 Describe the components of a disributed database management system 1.4 Summarise the common use of data warehouses 1.5 Explain the meaning of the term data warehouse 1.6 Describe the structure of a data warehouse
2. Understand how to enhance the design of and further develop a database system	2.1 Describe how tables that contain redundant data can suffer from update anomalies 2.2 Explain how to overcome update anomalies using normalisation 2.3 Describe how to retrieve data from one or more tables using SQL
3. Be able to enhance a logical database design	3.1 Check the tables are well-structured using normalisation 3.2 Define the integrity constraints on the tables
4. Be able to develop a physical database design	4.1 Map a logical database design to a physical database design 4.2 Design tables for a target DBMS 4.3 Design a representation of derived data 4.4 Design integrity constraints for the target DBMS 4.5 Denormalise tables where appropriate
5. Be able to enhance a database system using SQL	5.1 Apply integrity constraints 5.2 Retrieve data from one or more tables using join 5.3 Retrieve data from one or more tables using sub-queries

3. Syllabus

Syllabus			
Topic No	Title	Proportion	Content
1	Key Concepts in Databases and Database Development	1/12 2 hours of lectures 2 hours of tutorials 1 hour of laboratory sessions	• Review of key material from Level 4 databases module • Common uses of databases • Types of databases • Overview of database development ***Learning Outcome: All***
2	Enhancing Design 1	1/12 2 hours of lectures 2 hours of tutorials 1 hour of laboratory sessions	• Introduction to normalisation • The concept of functional dependency • Data redundancy and update anomalies • Overcoming anomalies with normalisation ***Learning Outcome: 2***
3	Enhancing Design 2	1/12 2 hours of lectures 2 hours of tutorials 1 hour of laboratory sessions	• Deriving a set of relations from a conceptual data model • Validating relations using normalisation • Integrity constraints on tables ***Learning Outcome: 3***
4	Data Retrieval 1	1/12 2 hours of lectures 2 hours of tutorials 1 hour of laboratory sessions	• Table and view structure in a relational database • Data types • Null values • Retrieving data using SQL ***Learning Outcome: 2***

5	Data Retrieval 2	1/12 2 hours of lectures 2 hours of tutorials 1 hour of laboratory sessions	• Referential integrity in relational databases • Types of joins • Retrieving data using joins • Retrieving data using sub-queries ***Learning outcome: 5***
6	Physical Design 1	1/12 2 hours of lectures 2 hours of tutorials 1 hour of laboratory sessions	• The purpose of physical design • Mapping the logical database design to a physical database design • Designing tables for the target DBMS ***Learning Outcome: 4***
7	Physical Design 2	1/12 2 hours of lectures 2 hours of tutorials 1 hour of laboratory sessions	• The concept of derived data • Designing a representation of derived data ***Learning Outcome: 4***
8	Physical Design 3	1/12 2 hours of lectures 2 hours of tutorials 1 hour of laboratory sessions	• Types of constraints • Designing integrity constraints for the target DBMS ***Learning Outcome: 3 & 5***
9	Physical Design 4	1/12 2 hours of lectures 2 hours of tutorials 1 hour of laboratory sessions	• Understanding transactions • Denormalisation • Improving performance • Estimating the size of the database ***Learning Outcome: 4***

10	Distributed Databases	1/12 2 hours of lectures 2 hours of tutorials 1 hour of laboratory sessions	• The need for distributed databases • Components of distributed databases • Advantages and disadvantages of distributed databases • Homogenous and Heterogeneous distribution • Distributed Database Design ***Learning Outcome: 1***
11	Data Warehouses	1/12 2 hours of lectures 2 hours of tutorials 1 hour of laboratory sessions	• The need for business intelligence and the concept of the data warehouse • The difference between Online Transaction Processing (OLTP) systems and data warehousing • The architecture and main components of a data warehouse ***Learning Outcome: 1***
12	Summary	1/12 2 hours of lectures 2 hours of tutorials 1 hour of laboratory sessions	• Summary of module, linking units to objectives and to each other • Clarification of material and related issues as identified by students ***Learning Outcome: All***

4. Related National Occupational Standards

The UK National Occupational Standards describe the skills that professionals are expected to demonstrate in their jobs in order to carry them out effectively. They are developed by employers and this information can be helpful in explaining the practical skills that students have covered in this module.

Related National Occupational Standards (NOS)
Sector Subject Area: 6.1 ICT Professionals **Related NOS:** 4.1.P.1 – Contribute, under supervision, to the preparation of a data analysis assignment; 4.1.P.2 – Assist in the development of data analysis models; 4.1.P.3 – Manage the outcomes from the data analysis assignment; 4.5.P.2 – Manage, under supervision, the maintenance of data design assignments; 4.5.P.1 – Provide others, when requested, with specified information relating to data design activities; 4.5.S.1 – Select and implement appropriate data design processes; 4.5.S.2 – Manage the progress of data design assignments; 4.5.S.3 – Review the effectiveness of data design deliverables.

5. Resources

Lecturer Guide: This guide contains notes for lecturers on the organisation of each topic, and suggested use of the resources. It also contains all of the suggested exercises and model answers.

PowerPoint Slides: These are presented for each topic for use in the lectures. They contain many examples which can be used to explain the key concepts. Handout versions of the slides are also available; it is recommended that these are distributed to students for revision purposes as it is important that students learn to take their own notes during lectures.

Student Guide: This contains the topic overviews and all of the suggested exercises. Each student will need access to this and will need to bring it to every taught hour for the module.

This module also makes use of SQL. You may choose which version of SQL is to be used but students will need to have access to this during laboratory and private study time. You may wish to consider MySQL. This is open source software which is available from http://www.mysql.com/downloads.

6. Pedagogic Approach

Suggested Learning Hours					
Lectures:	**Tutorial:**	**Seminar:**	**Laboratory:**	**Private Study:**	**Total:**
24	24	-	12	90	150

The teacher-led time for this module is comprised of lectures, laboratory sessions and tutorials. The breakdown of the hours is also given at the start of each topic.

6.1 Lectures

Lectures are designed to start each topic and PowerPoint slides are presented for use during these sessions. Students should also be encouraged to be active during this time and to discuss and/or practice the concepts covered. Lecturers should encourage active participation wherever possible.

6.2 Tutorials

These are designed to deal with the questions arising from the lectures and private study sessions. For some topics these will be structured sessions with students engaging in tasks related to the lecture. Other sessions will involve problem solving and trouble-shooting discussions related to the practical work.

6.3 Laboratory Sessions

During these sessions, students are required to work through practical tutorials and various exercises. The bulk of the tutorial sessions will be related to gaining a sufficient level of mastery of the chosen database tool and the SQL language sufficient to implement the assessment task. Students will be introduced to relevant topics in SQL in the laboratory sessions and this learning will be augmented by lecture and tutorial sessions. The details of these are provided in this guide and also in the Student Guide.

6.4 Private Study

In addition to the taught portion of the module, students will also be expected to undertake private study. Exercises are provided in the Student Guide for students to complete during this time. Teachers will need to set deadlines for the completion of this work. These should ideally be before the tutorial session for each topic, when Private Study Exercises are usually reviewed.

7. Assessment

This module will be assessed by means of an assignment worth 50% of the total mark and an examination worth 50% of the total mark. These assessments will be based on the assessment criteria given above and students will be expected to demonstrate that they have met the module's learning outcomes. Samples assessments are available through the NCC Education Campus (http://campus.nccedu.com) for your reference.

Assignments for this module will include topics covered up to and including Topic 7. Questions for the examination will be drawn from the complete syllabus. Please refer to the Academic Handbook for the programme for further details.

8. Further Reading List

A selection of sources of further reading around the content of this module must be available in your Accredited Partner Centre's library. The following list provides suggestions of some suitable sources:

Benyon-Davies, P. (2003). *Database Systems,* 3rd edition*.* Palgrave Macmillan.
ISBN-10: 1403916012
ISBN-13: 978-1403916013

Connolly, T. and Begg, C. (2003). *Database Solutions: A step-by-step guide to building database*, 2nd edition. Pearson Addison-Wesley.
ISBN-10: 0321173503
ISBN-13: 978-0321173508

Connolly, T. and Begg, C. (2014). *Database Systems: A Practical Approach to Design, Implementation and Management,* 6th edition. Pearson Addison Wesley.
ISBN-10: 0132943263
ISBN-13: 978-0132943260

Dietrich, S. (2001). *Understanding Relational Database Query Languages.* Pearson Prentice Hall.
ISBN-10: 0130286524
ISBN-13: 978-0130286529

Kroenke, D. (2013). *Database Processing: Fundamentals, Design and Implementation,* 13th edition. Prentice Hall.
ISBN-10: 0133058352
ISBN-13: 978-0133058352

Topic 1: Key Concepts in Databases and Database Development

1.1 Learning Objectives

This topic provides an overview of the module as a whole, a review of key material from Level 4 and an introduction to the stages of database design and development.

On completion of the topic, students will be able to:

- Understand which topics will be covered in the module;
- Recognise key material from the Level 4 Databases module;
- Outline common uses of databases and different types of databases;
- Give an overview of database development.

1.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practise the skills during the private study, laboratory and tutorial sessions.

1.3 Timings

Lectures:	2 hours
Laboratory Sessions:	1 hour
Tutorials:	2 hours
Private Study:	7.5 hours

1.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Scope and coverage of the module
- Overview of key topics from the Level 4 Databases module
- Outline of database development

1.4.1 Guidance on the Use of the Slides

Slides 2-3: Go through the scope and coverage of the module. Give an outline of the learning outcomes of the current topic. Note that the learning outcomes will be revisited at the end of the session to ensure that they have been met.

Slide 4: The focus of this module is on database development and particularly the various stages of design: conceptual, logical and physical. Students will learn how to move from a set of requirements to a fully implemented database, using techniques from the Level 4 module and new techniques and using the SQL language. There are also some additional topics towards the end of the module: Distributed Databases and Data Warehouses. Distributed Databases will enable us to examine the challenges of design where a database is spread geographically and/or between the different departments. Data Warehouses will enable us to examine the challenge of integrating data from different databases into a larger system used for gathering management intelligence.

Slide 5: Topics will be introduced in lecturers and then followed up in private study and tutorial sessions. Tutorial sessions will also be used to examine and discuss coursework. Laboratory sessions will be used to develop skills in SQL to enable the students to implement a database.

Slide 6: This slide provides key features of a database. The information should be familiar to students from the Level 4 module (this diagram was originally based on diagrams in D. M. Kroenke's *Database Fundamentals (*2005); however, this is not in any way a required textbook). The key points to note are that in a database system, data is integrated in one place, the database. Different applications can access the same data in the database. The interaction between applications and the database is handled by the database management system (DBMS).

Slide 7: This slide looks at common types of database systems.

- *Transaction Processing System (TPS)* - Typical 'everyday' database system. When an application, such as a form, updates something, then it will be saved to the database straight away. For example, entering a new customer's details onto a sales system; when 'Save' is pressed, the new customer is saved to the database. Later in the module, the concept of a 'transaction' will be revisited and a more precise definition will be given.

- *Management Information System (MIS)* - System that looks at data in various ways to get knowledge. Reports will be produced from either a TPS or data warehouses.

- *Data Warehouses* - Amalgamations of databases. Data is not live on these systems, because it can't be updated; it will be drawn from TPS. These systems will be interrogated by applications to gain management information, such as discovering trends, e.g. the sorts of products that were bought in a particular geographic area.

- *Distributed Databases* - Large organisations might be spread geographically and so their databases have to cope with this. Data might be split or replicated.

Slide 8: This slide gives some common uses of databases; students will have seen similar examples in their previous study. Ask students for any other examples that they can think of.

Slide 9: The students should be asked to come up with their own definitions and examples here.

- *Entity*: something of significance about which we want to keep data. This could be a person, thing or concept.

- *Attribute*: the quality or property of an entity.

- *Relationship*: the way one entity is linked to another. This represents real world relationships, such as a customer buying a product. It can also represent concepts within our data model, such as customer types relating to customers.

Slides 10-11: The students should attempt to draw the ER diagram. Mention as part of the explanation why it is not necessary to have a relationship between student and course - because it is derivable IF a module belongs to only one course. How would they model it otherwise?

Slide 12: Students should have already covered the concept of metadata in previous study. You should ask students to give a definition.

As a recap from previous study, metadata is data that contains the structure of other data. The structure of the tables in a relational database is kept within the database itself in the form of metadata. This defines the name of the table, the name of the column, the length of the column and the data-type. The students could be asked at this point about their understanding of data-types, e.g. what data-types are available? It should be pointed out that different implementations of relational databases from different vendors might have slightly different names for the same data-type, even though there are standards. Metadata is important, because it is the way a database knows about its own structure and is able to realise one of the advantages of databases: program-data independence. This is because with metadata, the structure of the database, such as the tables, is stored in the database itself. A programmer only needs to know how to access this in order to be able to retrieve data from the database. The collection of metadata in a database is known as the data dictionary.

Slide 13: This slide looks at key concepts of the Relational Model.

- *Relations and Tables* - Tables can be thought of as the most basic structure in the relational model. Within the model itself, as opposed to its implementations, the equivalent of table is known as a relation. Tables are made up of attributes. Relations are implemented in the database as two-dimensional tables made up of columns and rows.

- *Attribute* - The qualities or property of an entity (already mentioned). It corresponds to a column in a table.

- *Domain* - The set of valid values of an attribute. For example, the valid values in the domain 'sex' would be 'Male' and 'Female'.

- *Tuples and rows* - An instance of an entity is known as a tuple. It corresponds to a row in a table. Each tuple or row represents the set of values in the attributes of a particular table.

- *Primary Key* - The attribute or attributes that uniquely identify a row in a table.

- *Foreign Key* – This is an attribute which references an attribute (usually the primary key) in another table. Foreign keys are the way in which relationships are represented in the implemented database.

Slides 14-15: Point out that database development involves skills from other disciplines within ICT, e.g. systems analysis. There are particular issues with regards to database development that are different from other types of development, for example the development of networks or websites. This module focuses on the unique elements.

'Requirements Gathering' is part of systems analysis. There are different methodologies available, but recently iterative approaches have been popular. These entail a large amount of user involvement, for example by using prototypes that are shown to the user to enable them to better identify what they want.

Slide 16: This slide gives a definition of database design from Connolly and Begg. The important thing to get over to students is that database design involves moving from a set of requirements to implementing these with database technology.

Slides 17–18: The different phases of database design are the key focus of the module. Each of the phases will be outlined in this session and also examined in greater detail in the appropriate topic (see Slide 18).

Slide 19: The first phase is conceptual database design. At this stage, data is investigated and design is undertaken without regard either to the physical implementation OR the data model. So the designers at this stage do not even assume they will be using the relational model. The sorts of areas of investigation are: what data does the enterprise hold? What format is it in? How is it used? How might the use of data affect how it is held? What terminology do users have for their data?

The activities that go to make up this phase of the design process will be discussed in more detail later in the module. Gaining an overview of these activities will be part of this topic's private study and tutorial.

Slide 20: Ask the students what they think the purpose of conceptual design is.

A possible area for discussion is that conceptual design means that we can look at data independently without being influenced by models or the particular database product we will use to implement. At this stage, the design still leaves open the question of how the requirements will be implemented.

Slides 21–23: The second stage is logical database design where the model is constructed without regard for the particular DBMS that will be used. However, the data model (e.g. the relational model) is known.

The activities that go to make up this phase of the design process will be discussed in more detail later in the module. Gaining an overview of these activities will be part of this topic’s private study and tutorial.

A key activity is normalisation which is described briefly on Slide 23.

Slide 24: The final stage is physical database design. The move from entities to tables is one of the key activities here involving what is known as designing the base relations. But this phase also involves other activities, such as indexing, denormalisation, view creation and query tuning.

The activities that go to make up this phase of the design process will be discussed in more detail later in the module. Gaining an overview of these activities will be part of this topic’s private study and tutorial.

It should be pointed out to students that a large proportion of this module falls under the category of physical database design.

Slide 25: As well as the phases mentioned, there are additional aspects of development. These are not covered in detail on this module and will be researched as part of this topic’s private study and tutorial work. These phases should be investigated as part of the private study and tutorial activities.

- *DBMS selection* - Choosing which is the appropriate DBMS to support the database system
- *Application design* - The design of the user interface and the application programs that use and process the database
- *Transaction design* - Investigating what transactions will need to be catered for
- *Data Conversion* - If the new database system is to replace an already existing system (either manual or computerised) how will the data be transferred from one to the other?
- *Testing* - How will the database be tested? This needs to cater for tests for particular transactions, queries etc. as well as scaling which tests if the database performs well when used by lots of users or deployed across lots of machines over a network.

Slide 26: Ask the students if the learning outcomes have been met. Recap the main points of the lecture.

Slide 27: This slide gives a list of references which students may find useful.

1.5 Laboratory Sessions

The laboratory time allocation for this topic is 1 hour.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

It is envisaged that the first session will include some set up time to ensure that all students have SQL accounts. You can use any SQL which implements all the standard features such as Oracle, SQL Server or MySQL. This will need to be available to each student during all of the laboratory sessions. You will need to be familiar with the features of whichever version you are using in order to be able to instruct the students. You will also need to monitor students carefully as they work through the exercises and provide guidance as necessary

During the laboratory session, you should also allocate the topics for research in Private Study Exercise 1.

These laboratory sessions will involve you in creating two sets of tables. In this session you will create the students and courses tables that will be used throughout the lab sessions in later topics.

Later in the module you will create some tables around a scenario called 'Marlowe Interiors'. These will also be used in some laboratory sessions.

Creating Tables

You should remember how to create tables and insert data from previous study. Here is an example from the Level 4 Databases module exercises:

```
Create table departments
(dept_no  integer not null primary key,
department_name varchar(30),
location varchar(20);

Create table workers
(emp_no integer not null primary key,
first_name varchar(30),
last_name varchar(30),
job_title varchar(30),
age integer,
dept_no integer,
foreign key (dept_no) references departments)
```

Exercise 1:

Create two tables, 'Courses' and 'Students'.

The 'Courses' table should have a course_id which is an integer, and a course_name which is a varchar 30 long. Course_id will be the primary key.

The 'Students' table should have a student_id, an integer which is the primary key.

Other attributes:

```
first_name varchar(30)
last_name varchar(30)
student_type varchar(20).
```

course_id is a foreign key that references course_id on the courses table.

Suggested Answer:

```
Create table courses
(Course_id integer not null primary key,
course_name varchar(30));

Create table students
(student_id integer not null primary key,
 first_name varchar(30),
 last_name varchar(30),
 student_type varchar(20),
 course_id integer references courses(id));
```

Be aware that different vendors implement specification of foreign keys differently.

```
Create table students
(student_id integer primary key,
first_name varchar(30),
last_name varchar(30),
student_type varchar(20),
course_id integer,
foreign key (course_id) references courses(course_id));
```

Exercise 2:

Inserting data into the two tables

The format of an insert statement is shown in this example from the Level 4 module.

```
Insert into departments values ('1','Packing','Cairo');
```

Using the same format use the Insert command to create the following courses:

1. Computing Science
2. History
3. Geography

Insert the following students using the SQL insert statements:

Remember that the course_code must reference one of the course codes you have just created in the courses table.

1. Pavel Dobovitch, Home Student, 1
2. Winston Kodogo, Overseas Student, 1
3. Dawn Cove, Overseas Student, 1
4. Satpal Singh, Home Student, 2

5. Horace Smith, Home Student, 3

Don't forget to commit your data!

Suggested Answer:

```
Insert into courses values (1, 'Computing Science');
```

Repeat the insert with the additional values.

```
Insert into students values (1, 'Pavel’, ‘Dobovitch','Home Student',1);
```

Repeat the insert with the additional values.

1.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Write a report in two parts.

The first part should give an overview of the three main phases of database development: conceptual design, logical design and physical design. You should outline the main activities of each of these phases. Research these phases using textbooks and web sources. Here are some examples of textbooks you could use, but you may also find your own resources. Please note that these are not required textbooks.

- Connolly, T. and Begg, C. (2005). *Database Systems A Practical Approach to Design, Implementation and Management*, 4th edition. Addison and Wesley. Chapters 1 and 9.
- Benyon-Davies, P. (2000). *Database Systems,* 3rd edition. Palgrave. Chapter 14.
- Kroenke, D. M. (2005). *Database Processing: Fundamentals, Design and Implementation,* 10th edition. Prentice Hall.
- Applied Information Science. (1997). *Conceptual, logical and physical models.* [Available Online] http://www.aisintl.com/case/CDM-PDM.html
- Nicewarner, N. (2004). Why we have conceptual, logical and physical data modelling? *Power Builders Developer's Journal,* 1 Feb 2004. [Available Online] http://pbdj.sys-con.com/node/106944

The second part should be on one of the additional aspects of database development. Your lecturer will allocate the topics to ensure that different content is covered evenly.

- DBMS selection
- Application design
- Transaction design
- Data Conversion
- Testing

Your report should be 600-900 words in length and you should be prepared to discuss your report in the tutorial session.

Suggested Answer:

The report should be about 600 to 900 words. It should give an overview of the stages of database development AND the chosen additional aspect of development that the student has been allocated. Students should be encouraged to carry out research online and use the results of this to produce a document that is in their own words.

Points that should be covered by students:

DBMS Selection

- Need to assess the needs of the organisation in terms of size. If it is a small organisation can a desktop database such as MS Access meet the needs? If a larger product is needed then there are lots to choose from?
- Compatibility with existing systems could be an issue.
- Cost.
- Scalability. How efficient is the DBMS when working at larger scales with lots of rows of data?
- How well supported is the product by the vendor?
- How does the product fit with the applications that the user will want?

Application Design

- User requirements. This will involve a full sub-project as part of the main database development project. The users will need to have their requirements investigated as to what applications they need.
- Types of applications. Different sorts of applications should be discussed. Forms. Reports. Batch processes. Web-based applications.
- User roles: who will use these applications and to what purpose?
- User Interface Design. To many users the screen-based forms or web-pages they use ARE the database! The design of these is very important and the disciplines of user interface design with its guidelines and principles should be applied.

Transaction Design

This is linked to application design in the sense that many of the applications will be performing transactions.

Questions that will need to be investigated are:

- What is the purpose of the transaction in a business sense?
- What data will the transactions affect?
- What operations will it carry out?
- How will it alter the data?
- How often does the transaction run?
- How much data does the transaction affect?
- As the development process progresses transactions will be examined at a greater degree of detail. For example when the table structures have been defined then operations at that level can be documented with CRUD matrices.

Data Conversion

This concerns moving data from the systems that already exist to the new database system.

Areas of investigation are:

- Does the data already exist or will it need to be gathered?

- What systems exist that contain data that the new system will need?
- For each of the systems that exists what form are they in? Systems could be paper-based, or they could be computer systems. Computer systems could be flat files, spreadsheets or database systems.

Testing

A rigorous approach to testing will be needed.

In an iterative approach then a certain degree of testing is incorporated in the evaluation process of each cycle.

Final testing will involve:

- Unit testing: each 'unit' of the database such as an application and the data structures that it accesses.
- Link testing: how one part of the database interacts with another.
- System testing: how the database and its applications operates as a system.
- For each system, and for each data structure in that system the state of the data will be need to be examined. Is the structure of the data compatible with the new system? Is the data in an accurate state or will it need to be 'clean' before it can be loaded into the new system?
- The data loading itself. How will this be done? Will it be manually accomplished by people typing in the data? Will some sort of data loading tool be used?

Exercise 2:

Read the following scenario about a company called Marlowe Interiors. This scenario will be used as an example in future private study, tutorial sessions and lab sessions.

> *Background on the firm*
>
> Marlowe Interiors is a trendy interior design and building firm located in Islington, North London. They specialise in 'makeovers' to individual rooms or entire properties. They evolved from the traditional building firm of Ted and Arthur Barrett, two brothers who had done all sorts of work in the area for many years. With the craze for makeovers taking off, the brothers decided to seize the time and change the nature of the business. This involved recruiting some new workers, notably Horace the interior designer, and changing their image with a new name and logo. However old habits die hard and the accounts and records of the company were kept in the same old way, in files in Ted and Arthur's office. For example there is a card box with index cards for customer records and a box file with copies of all the old invoices that have been sent out. Now the brothers have decided that since the firm is doing very well then it is time to have a computerised record system that can hold all the information they need in one place.

Determine what entities and attributes you can identify. Also suggest what further materials or information from the firm would be useful in identify entities and attributes.

Suggested Answer:

It should be pointed out that at this stage there is not the complete information about the company. Other materials that could be useful are things like invoices, internal documents, the index cards for customer records etc. Samples of these will be given later in the module.

Students should, however, be able to suggest some entities: customer, worker, possibly job. They might also suggest invoice, which is fine at this stage, but later will come to be seen as derived data.

1.7 Tutorial Notes

The time allowance for tutorials in this topic is 2 hours.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

The tutorial should be used as a discussion forum for the reports produced as part of the private study time. You should also answer any questions from students regarding the content of this topic.

Exercise 1:

In small groups based on the topic that you have been given, discuss the topic and develop a presentation for the rest of the class. The presentation should last approximately 5 minutes. There should be presentations on each of the following areas:

- DBMS selection
- Application design
- Transaction design
- Data Conversion
- Testing

Suggested Answer:

After each presentation lead the class in a group discussion of the topic, introducing any relevant information that was not contained within the presentation.

Exercise 2:

Again in different small groups, discuss the following stages of database design and present the information to the rest of the class:

- Phases of database design: conceptual design
- Phases of database design: logical design
- Phases of database design: physical design

Suggested Answer:

After each presentation lead the class in a group discussion of the topic, introducing any relevant information that was not contained within the presentation.

Topic 2: Enhancing Design 1

2.1 Learning Objectives

This topic provides an overview of normalisation. On completion of the topic, students will be able to:

- Explain the reason why the process of normalisation is carried out;
- Normalise a document to third normal form.

2.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practise the skills during the private study, laboratory and tutorial sessions.

2.3 Timings

Lectures:	2 hours
Laboratory Sessions:	1 hour
Tutorials:	2 hours
Private Study:	7.5 hours

2.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Introduction to normalisation
- The concept of functional dependency
- Data redundancy and update anomalies
- Overcoming anomalies with normalisation

2.4.1 Guidance on the Use of the Slides

Slides 2–3: The focus of this lecture will be on normalisation as a way of producing a set of relations in a desired state so as to minimise repetition. Normalisation is a 'bottom-up' approach that should complement the 'top-down' approach of ER modelling. Some of the material in this topic will recap material covered in the previous study. An extended case study will be introduced that will allow students to examine the concept of normalisation.

Slide 4: Recap on some of the key concepts of the relational model that are necessary for an understanding of normalisation.

- ***Relational integrity*** refers to the different rules that exist within the model to make sure that it is made of relations.
- ***Nulls*** represent values of an attribute that are unknown. Note that this does NOT mean blank or zero. Because null means unknown it is NOT possible to say that an attribute with a value of null is equal to another attribute with a value of null.
- ***Entity integrity***. This rule is about making sure that each tuple (or row) in a relation is unique. The rule states that: no attribute of a primary key can be null.
- ***Referential integrity***: if a foreign key exists in a relation it must much a candidate key in its home relation or must be null.
- ***General constraints***: any additional rules that are set up at the behest of the users to satisfy their requirements. For example in a database of voters in an election a rule could be set up that said all voters must be over a certain age.

Slide 5: ***Activity:*** Ask the students why an attribute that is a primary key (or part of a primary) key cannot not be null. Why would this potentially violate uniqueness?

Slide 6: ***Answer:*** Because a null value, being unknown, might be the same as the value in the primary key of another tuple. Ensure that students understand this answer.

Slide 7: Properties of a relation.

- Relation has a name that is unique within the relational schema.

- Each cell in the relation contains a single (or atomic) value.
- Each attribute has a name unique within the relation.
- The values of any one attribute should be drawn from the same domain.
- There are no duplicate tuples (rows) within a relation.
- There is no significance to the order of attributes.
- There is no significance to the order of tuples.

Slides 8–9: Within a relation if it is said that ‘A determines B’ then this means that if you know the value of ‘A’ then you will know the value of ‘B’. So A ***functionally determines*** B. Or B is ***functionally determined by*** A. Note that the reverse is not true. The diagrammatic representation of this is:

A → B

A concrete example is attributes from a table of students.

StudentID → StudentName

If we know the StudentID then we know the StudentName. But the reverse is not true because we might have students who have the same name.

Slides 10 –11: Students should be asked to identify the candidate key. Remember the formal definition of a candidate key is that firstly it is a super-key (an attribute or set of attributes that uniquely identifies a tuple). Secondly it is a super-key such that no part of it could be a super-key on its own. Therefore it is the minimum set of attributes that could uniquely identify a tuple. It is called a ‘candidate key’ because it is a candidate to become a primary key.

In this example the functional dependencies can be examined to help the student identify a candidate key. If StudentId is known, does that mean that the other two attributes are known? The answer is no. If activity is known then the fee would be known, but not the StudentId. Knowing the fee tells nothing about the other attributes so there are functional dependencies there. What is clear is that the candidate key is a combination of other attributes.

However, if this doesn’t ‘feel’ right, semantically speaking, this is because this relation is not fully normalised, there are anomalies.

Slide 12: ***Activity:*** The students should be asked to think about loss of information if a tuple is deleted from this relation. The answer is that we would lose the price of Fencing as well as the fact that student 55 was taking Fencing. This deletion anomaly is one of the anomalies that can occur when relations are not fully normalised.

There are also insert and update anomalies.

If we want to record a new activity but no one has yet taken it we cannot do so because we need a StudentID because the StudentID is part of the primary key and therefore cannot be null. This is an insert anomaly.

If we wanted to change the cost of Dancing to ‘75’ we would have to do it for every tuple where someone was taking Dancing. In a relational database doing such repetitive updates should not be necessary. It points to an update anomaly.

The problem with this relation is that it contains details about two separate facts. Who is taking an activity and how much the activity costs? The solution is to split the relation.

Slide 13: The anomalies in the previous section should be revisited. They have been overcome. This is because the relations are now normalised.

The functional dependencies here are much clearer. In student-activity the two attributes are self-contained since a student may take many activities they both must be part of the primary key. In activity-cost there is a functional dependency whereby activity determines cost. If we know the activity we know the cost. The attribute ‘activity’ will be the primary key.

Slide 14: There is a more formal process of normalisation. What follows is one example of it. There are other approaches although the rules for each normal form are the same. The starting point is a paper document of the sort that is used to store data in a manual system.

It is worth noting a number of features of this document. It is a document containing data about customers of an art supplies shop and the items they have ordered.

Note also that this is information for one customer and that for that one customer, as for all the other customers in the system, there is information about more than one item they have ordered. In this example there are seven rows of data about modules. These rows are called, in the language of normalisation, ‘repeating groups’. For each customer there are repeating groups of information about items, price, suppliers etc.

Slide 15: The first step is to identify which attributes belong to the repeating group. The attributes are listed as shown. Those attributes where there is one occurrence are annotated with a ‘1’. Those attributes where there is a repeating group are annotated with a ‘2’. The ‘2’ in this case simply means ‘more than one’. The tentative primary key is also underlined. In this case it is Customer Number.

Slide 16: First Normal Form. In the column marked ‘1NF’ (for first normal form) the repeating group information is separated out. Note the important step also: the Customer Number which is the primary key of the upper set of attributes, is also copied down to become a foreign key in the lower block. Foreign keys are identified by a star. This maintains the link between the customer information and the item information. The repeating group data is item information for a particular customer. The primary key of the lower block of attributes (the repeating group) is also identified as being the combination of Customer Number and Item Number: once again this makes sense semantically because this is data about items that a particular customer has ordered.

It is worth stressing that this step, identifying the primary key of the repeating group block, is often a cause for confusion. If this is not done properly then the remaining steps will go wrong.

Slide 17: For second normal form we need to examine the functional dependencies that exist when there is a primary key that is made up of more than one attribute.

Activity: Which block of data do we have to examine now?

Answer: the repeating group block because it has a primary key made up of more than one attribute.

Examine each non-key attribute in the relation and see if it is fully dependant on the WHOLE primary key.

Attributes:

- Item Name – No, this is just dependent on the Item Number
- Supplier ID – No, this is just dependent on the Item Number since it indicates which supplier provides that particular item.
- Price – No, this is just dependent on the Item Number since it is the price for the item.
- Supplier Name – No, this is just dependent on the Item Number since it shows the name of the supplier that provided the item.
- Quantity – Yes. The quantity is about how many of this item was ordered by this customer. So it pertains to both parts of the primary key.

Next step.

Where we have identified attributes that are dependent on one part only of the primary key we separate them out. We take the part of the primary key out that they are dependent on. That is also left behind and becomes a foreign key. In this case we have separated out the Item information and Item Number becomes the primary key of the Item information. Item Number is left behind as a foreign key.

Slide 18: Third normal form. The process now looks for any functional dependencies in a relation that are not on the primary key. Go through each attribute in turn to see if it is dependent on the primary key directly.

In this case the Supplier Name is only dependent on the Supplier ID. So we separate this data out.

Slides 19–20: What we are left with at third normal form is different blocks of attributes that correspond to entities. We can now work from the bottom up and draw our entity diagram since we know a foreign key means the many end of a relationship.

Slide 21- 22: It should be noted that although normalisation looks sometimes like a rigorous and formal process, the steps can only be followed if the semantics (the meaning) of the data is understood. The developer would have to know that, for example, the price of an item depends on just the item. If the same item could be supplied by different suppliers at different prices then the normalisation would look different.

Activity: Ask the students what would change.

Answer: There would be a separate group of attributes with the primary key of Supplier Number and Item Number. Price would be functionally dependent on both of these attributes and would therefore be an attribute of this entity. Ensure that students understand this answer fully.

Slide 23: This example will be used throughout the module to demonstrate various aspects of development. Make the point to the students that when working in an organisational environment that part of the job of the developer will be to examine documents of various types in order to determine data structures. There is very rarely one master document that will give the developer everything they need; rather they would be working with many different documents that give fragments of the system and can often overlap.

Slide 24: The slide gives the key points from the case study that was presented in the Private Study for Topic 1. Students should have already read the full case study given below but it may be worth repeating it to jog their memories.

"Marlowe Interiors is a trendy interior design and building firm located in Islington north London. They specialise in 'makeovers' to individual rooms or entire properties. They evolved from the traditional building firm of Ted and Arthur Barrett, two brothers who had done all sorts of work in the area for many years. With the craze for makeovers taking off, the brothers decided to seize the time and change the nature of the business. This involved recruiting some new workers, notably Horace the interior designer, and changing their image with a new name and logo. However old habits die hard and the accounts and records of the company were kept in the same old way, in files in Ted and Arthur's office. For example there is a card box with index cards for customer records and a box file with copies of all the old invoices that have been sent out. Now the brothers have decided that since the firm is doing very well then it is time to have a computerised record system that can hold all the information they need in one place."

Activity: Ask the students: what is relevant here?

Answer: At this stage it is hard to tell. We are given background on the company and know that there will be various paper documents to examine.

Slide 25: This is the summary of the job sheet. Some of this might be self-explanatory but remember this process is all about the semantics of the data. In the private study and tutorial exercises there will be more detail on this that will help the students to understand the meaning of this data.

Slide 26: An initial (and wrong) approach for someone who was not a database professional would be to put all the data in the job sheet into one table or spreadsheet. The students should recognise by now that this will lead to a number of anomalies of the sort discussed earlier.

Activity: Point out some of the anomalies that would arise if this job sheet were put into one table.

- *Insert Anomalies:* To insert a new customer would require leaving a lot of null values because we would not know the details of the job at the early stage. Also parts and labour costs can only be inserted in the context of a job rather than stored separately. Also if a customer came back for a different job we would have to insert their data all over again.

- *Deletion Anomalies:* If we deleted a job that had certain unique information such as how much a sink cost then that would be gone for the whole system.

- *Update Anomalies:* changing the price of an item would mean changing it on every job that was current.

- There would be other similar anomalies with some of the other data.

- It is also important to note that the way the data is laid out (incorrectly) here suggests relationships that are ambiguous. How does the labour relate to the parts for example?

Slide 27: The exercises this topic will be to normalise the data from this case study.

Slide 28: This slide gives the learning outcomes for this topic; ask students whether they have been met.

Slide 29: This slide gives a list of references which students may find useful.

2.5 Laboratory Sessions

The laboratory time allocation for this topic is 1 hour.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

This session will use the students and courses tables created in the laboratory session for Topic 1.

Exercise 1:

Add a column to the students table to record a student's age.

Suggested Answer:

```
alter table students
add age int;
```

Update the students table to add the ages of the students:

Student ID	Age
1	30
2	21
3	21
4	33
5	19

```
update students
set age = 30
where student_id = 1;
```

Exercise 2:

These exercises serve as a recap of some basic SQL selections based on the tables created in the previous section.

1. Select first name and last name of all the students
2. Select the names of all the courses ordered by the course name
3. Select all the courses and the students that are in them
4. Select the last name and course name of all the overseas students
5. Show the course data and names of all the students studying history

Suggested Answer:

1.

```
select first_name, last_name
from students;
```

2.

```
select course_name
from courses
order by course_name;
```

3.

```
select course_name, first_name, last_name, student_type
from students, courses
where students.course_id = courses.course_id;
```

4.

```
select course_name, last_name, student_type
from students, courses
where students.course_id = courses.course_id
and student_type = 'Overseas Student';
```

5.

```
select course_name, student_id, first_name, last_name, student_type
from students, courses
where students.course_id = courses.course_id
and course_name = 'History';
```

Exercise 3: **Greater than; less than. Not equals to.**

SQL can use the following symbols:

Greater than

< Less than

<> Not equal to

>= greater than Or equal to

<= Less than or equal to

Use these structures to perform the following queries;

1. Select all the data for students over the age of 21
2. Select the first name, last name and age for students who are twenty one or over
3. Select the first name, last name, course id and course name for students who are not studying Computing Science

4. Select all the details for the students under 30 who are studying computer science
5. Show the first names for students studying Computing Science who are not Home Students

Suggested Answer:

1.

```
select * from students
where age > 21
```

2.

```
select first_name, last_name, age
from students
where age >= 21
```

3.

```
select first_name, last_name, age, courses.course_id, course_name
from students, courses
where students.course_id = courses.course_id
and course_name <> "Computing Science"
```

4.

```
select *
from students, courses
where students.course_id = courses.course_id
and course_name = "Computing Science"
and age < 30
```

5.

```
select first_name
from students, courses
where students.course_id = courses.course_id
and course_name = "Computing Sciene"
and student_type <> "Home Student"
```

2.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Read all the relevant material for this topic. This should including the slides and slide notes along with the sections from at least one relevant textbook.

You should make sure you understand the concepts of functional dependence and definitions of each normal form up to 3rd Normal Form.

You should make sure that you read some material on the existence of higher normal forms.

Prepare a short presentation that could be either a PowerPoint presentation or discussion document that describes:

a. Boyce-Codd Normal Form (BCNF)
b. Fourth Normal Form (4NF)

There should be a definition of each of these normal forms and an example. It is acceptable to use an example from a published source so long as a proper reference should be given. Be prepared to present your findings in the tutorial session for this topic.

Suggested Answer:

Note that students will not be expected to apply these higher normal forms in their development work at this level. The object of this work is to gain an awareness of them.

The content of the presentation could use the examples in any of the textbooks available to the students and/or examples from the web.

BCNF should include the definition that every determinant must be a candidate key. The concept of candidate keys has been covered during previous study and will be discussed again.

4NF deals with multi-valued dependencies. Multi-valued dependencies are where there are dependencies between attributes (A, B,C) such that for each value of A there is a set of values for B and a set of values for C but B and C are independent of each other.

4NF changes a relation such that there are NO multi-valued dependencies.

Note that there is a 5NF that deals with join dependency. This is beyond the scope of this module.

Exercise 2:

Carry out a normalisation to 3rd Normal form on the job sheet from Marlowe Interiors shown below.

Draw the ER diagram that results from this normalisation.

1. JOB SHEET

JOB NUMBER: 0023
CUSTOMER NAME: Carol Smiley
CUSTOMER ADDRESS: 1 Vickers Street, N1
JOB DESCRIPTION: Room redecoration

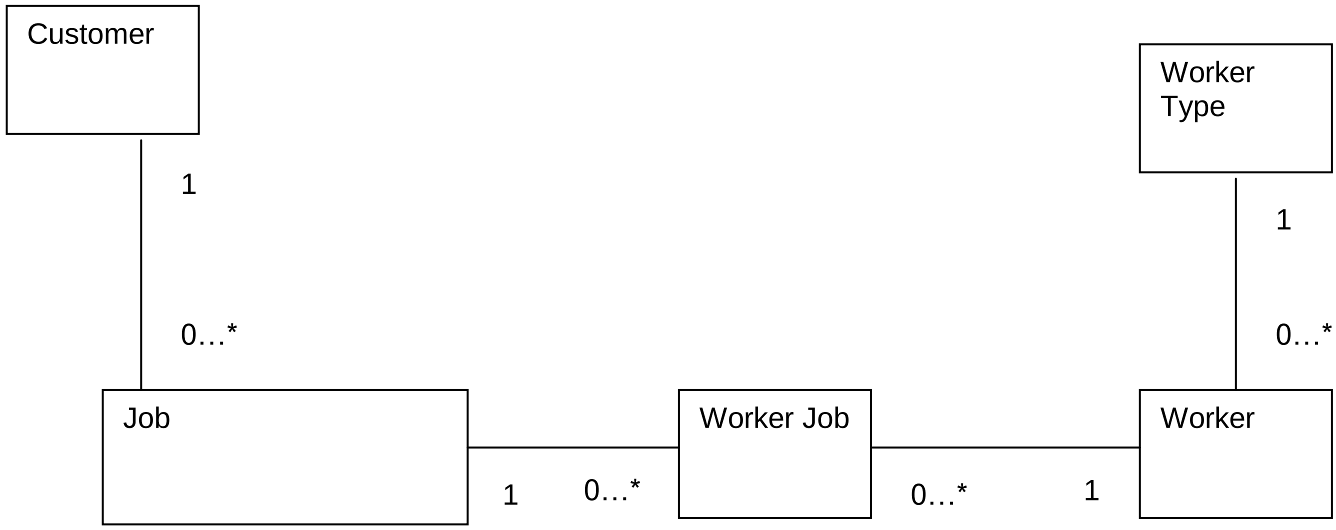
Employee No	Employee Name	Worker Type	Worker Type Hourly Rate	Hours worked on Job	Total for Job
009	Arthur Barrett	Qualified Builder	55	2	110
011	Sid Jones	Labourer	20	10	200
012	Jane Dean	Plumber	50	1	50
020	Horace D'Ville	Interior Designer	39	1	39
019	Steve Crabapple	Labourer	20	11	220

Suggested Answer:

UNF		1st Normal Form	2nd Normal Form	3rd Normal Form	Entity
Job Number	1	Job Number(PK)	Job Number(PK)	Job Number(PK)	Job
CustomerName	1	CustomerName	CustomerName	CustomerName(PK)(FK)	
Customer Address	1	Customer Address	Customer Address	Job Description	
Job Description	1	Job Description	Job Description		
Employee No	2			CustomerName(PK)	Customer
Employee Name	2	JobNumber(PK)(FK)	JobNumber(PK)(FK)	Customer Address	
Worker Type	2	Employee No(PK)	Employee No(PK)(FK)		
WorkerTypeHourlyRate	2	Employee Name	HoursOnJob	JobNumber(PK)(FK)	WorkerJob
HoursOnJob	2	Worker Type	TotalForJob	Employee No(PK)(FK)	
TotalForJob	2	WorkerTypeHourlyRate		HoursOnJob	
		HoursOnJob	EmployeeNo(PK)	TotalForJob	

		TotalForJob	Worker Type		
			WorkerTypeHourly Rate	EmployeeNo(PK)	Worker
				Worker Type(FK)	
				WorkerType(PK)	WorkerType
				WorkerTypeHourly Rate	

This ER diagram relates to the above normalisation:



2.7 Tutorial Notes

The time allowance for tutorials in this topic is 2 hours.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

For the presentations, you may wish to have different groups presenting on different areas for example some students could present their section on BCNF, the next group on 4NF etc. It could also be presented to a group of peers rather than the group as a whole.

Exercise 1:

In small groups, discuss your findings from Private Study Exercise 1 and develop a group presentation that gives a definition and example of each of the higher normal forms asked for in the exercise.

Each group should present their ideas and the information should be discussed as a whole class group.

Suggested Answer:

Please see private study exercises for this topic.

Exercise 2:

Discuss the following questions in small groups and feed back to the class as a whole:

a. What is the purpose of normalisation?
b. Explain the concept of functional dependency
c. Give a definition of normalisation up to 3rd normal form

Suggested Answer:

a. Normalisation aims to overcome potential anomalies that can occur when data is replicated. Normalisation aims to eliminate replication.

b. Functional dependency represents a relationship between attributes such that if the value of one attribute is known the value of the other attribute is known. It is not reversible.

c. Definitions of each of the normal forms:

   1st normal form aims to remove repeating groups.

   2nd normal form: removed an partial key dependencies

   3rd normal form: remove any non-key dependencies.

Exercise 3:

Normalise the following invoice from Marlowe Interiors shown below and draw the resulting ER model.

Marlowe Interiors

1 Newington Green Road
London
2. N1 TYY
020 7888 1234

Job: Kitchen Makeover
Customer: Ivan Jones, 2 Digby Mansions. Highbury Park
Area: London North
Date: 02/03/00
Parts
1 sink, tin @ 130.00 including VAT.
1 u-pipe @ 20.00 ditto
3 x assorted plumbing fittings @ 33.00 total including VAT
1 thermostat @ 100.00 including VAT
Labour
Plumber 3 hours £150
Labourer 3 hours £60
Electrician 1 hour £50 (to fit thermostat)

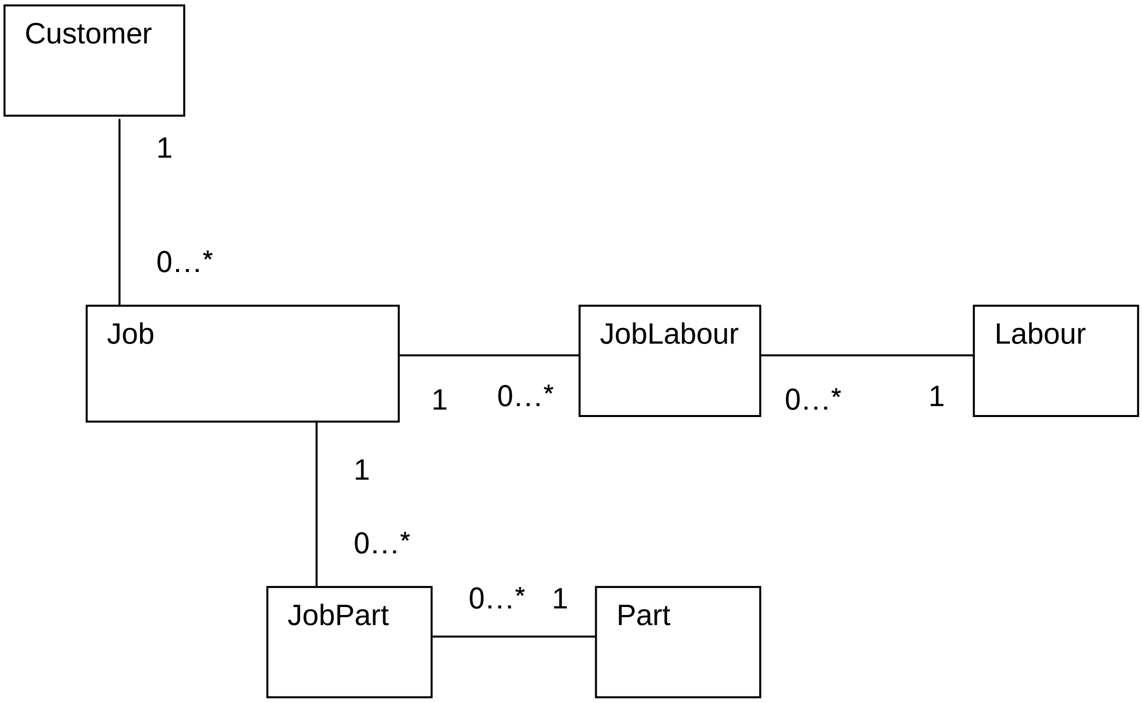
TOTAL PARTS	*£283.00*
TOTAL LABOUR	*£260.00*
TOTAL	*£543.00*

Suggested Answer:

UNF		1st Normal Form	Second Normal Form	Third Normal Form	Entities
Job	1	JobID(PK)	JobID(PK)	JobID(PK)	Job
CustomerName	1	Job	Job	Job	
CustomerAddress	1	CustomerName	CustomerName	CustomerName(FK)	
JobArea	1	CustomerAddress	CustomerAddress	JobArea	
Date	1	JobArea	JobArea	Date	
TotalParts	1	Date	Date	TotalParts	
TotalLabour	1	TotalParts	TotalParts	TotalLabour	
JobTotal	1	TotalLabour	TotalLabour	JobTotal	
		JobTotal	JobTotal		
Part	2			CustomerName(PK)	Customer
PartCost	2	JobID(PK)(FK)	JobID(PK)(FK)	CustomerAddress	
PartQuantity	2	Part(PK)	Part(PK)(FK)		
Labour	2	PartCost	PartQuantity	JobID(PK)(FK)	JobPart
LabourCost	2	PartQuantity	Labour(FK)	Part(PK)(FK)	
LabourQuanity	2	Labour	LabourQuanitity	PartQuantity	
		LabourCost			

		LabourQuanity	Part(PK)	JobID(PK)(FK)	JobLabour
			PartCost	Labour(PK)(FK)	
				LabourQuanitity	
				Part(PK)	Part
			Labour(PK)	PartCost	
			LabourCost		
				Labour(PK)	Labour
				LabourCost	

ER Diagram based on paper invoice:



Topic 3: Enhancing Design 2

3.1 Learning Objectives

This topic provides an overview of logical design. On completion of the topic, students will be able to:

- List the steps in logical design;
- Give an account of the main activities that make up logical design;
- Check the integrity constraints on their data model.

3.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practise the skills during the private study, laboratory and tutorial sessions.

3.3 Timings

Lectures:	2 hours
Laboratory Sessions:	1 hour
Tutorials:	2 hours
Private Study:	7.5 hours

3.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Deriving a set of relations from a conceptual data model
- Validating relations using normalisation
- Integrity constraints on tables

3.4.1 Guidance on the Use of the Slides

Slides 2-3: Note that here we are moving firmly from conceptual design to logical design. The tasks that will now be looked at will be using both the 'top down' approach of Entity Relationship Modelling and the 'bottom up' approach of Normalisation.

Slide 4: These are the two major steps. They are divided into a series of sub-steps that are discussed in this lecture.

Slide 5: It is important to note that throughout the discussion of both logical and physical design it is presented as though it were a series of steps that are carried out in a linear fashion, one after the other, with the output of one step being fed into the next step. This gives the impression of an ongoing process with very little room for reflection on what has happened. In the past, development methodologies have tended to be of this sort. The students will have come across this in discussions of systems analysis and development with the 'waterfall' model. More contemporary approaches to development are more likely to involve an iterative approach.

Ask the students if they understand what 'iterative' means. It means doing something more than once, repeating it.

So, in terms of database development it means that the steps examined here could involve revisiting, re-examining, and revision. One of the most important aspects of this to stress is user involvement. Users will look at the work produced by the database developer and feedback to him or her. For example the developer will produce ER models and the user will make comments on these. It is not unusual for the users to become quite adept at some of the tools and techniques that database developers use and make significant contributions to the development. This is often forgotten during discussions of development.

Slide 6 - 7: Step One involves creation of the key objects needed for understanding the data of any potential system. Work will have been done during systems analysis and conceptual design that has resulted in a mass of material relating to current processes and data. There will be initial ideas for what entities should exist in the new system. Step One of logical design will re-visit this material in a more rigorous way using some of the techniques of database development.

Slide 8: The starting point should always be the documentation gathered from discussions with users. This could be, at this stage, formal documentation such as a requirements catalogue; there will also be transcripts of interviews, the documents used in the daily running of the business (receipts, invoices etc.). Because, as has been stressed, this is an iterative process it can be useful to go back to users at various points for clarification about issues.

Looking for nouns and objects that exist in their own right - customers, for example, are real things that exist in the world. If an organisation has customers then it is likely that ‘Customer’ will be an entity.

With regard to the identification of entities there are a number of issues that might arise. Users within an organisation have their own jargon, the way they refer to what they do. This might not correspond to the everyday use of a given term. For example on a project for a regulatory body dealing with complaints every case they dealt with was not called a ‘Complaint’ or ‘Case’ but a ‘Matter’.

Beware of synonyms and homonyms. Synonyms are two different words with the same meaning. Homonyms are the same word that has a different meaning.

Examples of Synonyms:

- ‘Customer’, ‘Client’
- ‘Student’,’Pupil’,’Scholar’

Examples of Homonyms:

- Book (can be a book that you read or the verb to make a booking)
- Aisle and Isle – one is a walkway one is an island (Note that homonyms don’t have to be spelt the same).

Slides 9–11: Ask students to read the transcript of the interview on Slide 9 and attempt to identify entities. The process here is examining the nouns that occur in the transcript of an interview.

Slide 12: Gives a preliminary assessment of the discovered ‘entities’.

Slides 13–15: Sometimes identifying relationships is easier by just drawing an ER diagram and attempting to link the entities. By using these tables to cross reference entities to assess whether there is a relationship between them, the additional task of performing a quality check is also undertaken.

Ask the students if they have come across the term quality check. Basically the checking of all factors involved in any process or task.

In a large system with lots of entities there could be potentially many relationships. The task of cross referring every entity to every other entity would become too large to complete effectively. However this task could be aided by the use of a CASE tool.

Ask the students if they remember what a CASE tool is. Computer Aided Software Engineering. A CASE tool would have an entry where all the entities would be listed and a check could be made of all the relationships between them.

Slide 16: The students should think about what qualities a particular entity has that the users would be interested in recording In the case of customer it would be things like their name, address, telephone, email etc. It is worth noting that there is no scientific principle that can be applied to determine what is an attribute and what is an entity. For example if we were to record multiple addresses for one customer this might then mean that 'Address' became a separate entity with a one-to-many relationship to Customer.

Slide 17: The students should be asked whether they understand the concept of a domain. A domain is a set of valid values from which an attribute draws. This is related to the concept of a data type which is the format of the column in terms of whether it is a number or character string. A domain is a more strict definition of the values of a column. For example if we wanted to record the potential values of a six-sided die. The data would be integer, the domain would be the set of integers from 1 through to 6.

Later in the module we will see the various ways in which domains can be enforced on the database.

Slides 18–21: Primary Key and Foreign Key have been discussed - ask the students to recap on what they are.

- *Super Key* - An attribute or set of attributes that uniquely identifies a tuple.

- *Candidate Key* - A candidate key should be a super key. However ALL the attributes of this super key must be necessary to uniquely identify it: i.e. there should be no superfluous attributes that are part of the key. It should not be the case that any subset of the attributes that go to make up this key should qualify as a super key; ALL the attributes are necessary.

The primary key will be chosen from the candidate keys.

Assume the customer can only make one purchase a day.

- Option 1: The Date and Time are not both necessary because the customer won't have the same date twice.

- Option 3: This isn't really even a candidate key because there is the slight possibility that the customer could make an order on a different day but at the same time therefore violating the primary key.

- Option 2: The best choice here.

Slide 22: The next step is to check for redundancy. Ask the students if they know what redundancy means. Something is redundant if it is not needed. It may be the case that during the process of design carried out so far that structures (entities, attributes, relationships) may have been created that are not really necessary.

- *Re-examine one-to-one relationships*: They may have come about because of the use of synonyms and actually refer to one thing.

- *Time element*: if there is a relationship that exists and that has a time element make sure it is not lost when a situation changes. For example if someone had qualifications that were based on where they worked we need to ask what happens to those qualifications if they move job. Do they go with the person, if so how do we model it?

- *Remove redundant relationships:* A relationship is redundant if the same information can be found via some other relationships.

Slide 23: This example is based on one in Connolly and Begg. We don’t need to keep the ‘Rents’ relationship because it is possible to connect customers with the videos they rent via the Rental entity.

Slides 24 – 25: These slides give an example of some transactions. They should be modelled on a CRUD table as shown. More detailed documentation of transactions will be shown as an aspect of physical design but any information that has been gathered at this point should be recorded. This might include numbers of transactions happening in a particular time period, data that is affected etc.

Slide 26: Remember this is an iterative process. The users need to be involved in looking at the model so far in making comments, suggesting changes etc. This diagram shows an example of a generic iterative design methodology.

Slides 27–29: Describes the process of moving from entities to tables.

Slides 30–31: These slides look at transactions. If CRUD matrices have been constructed during logical design, then these should be checked for their fit to the new table structures. Where there have been changes from entity to table (such as resolution of many-to-many relationships), there will new table structures that are affected by the transactions and these will need to be checked. This checking process will need to be iterative as changes that affect the database take place.

Slide 32: Business rules are the methods, approaches and indeed ‘rules’ that govern the way an organisation does business. Often these can be unwritten and can be part of long engrained processes or part of the organisational culture, the way things are done.

Something like the rule ‘A customer order can only have up to 10 items on it’ is an example of a business rule.

These should be documented at this point. They may become constraints on the database, that is, built-in restrictions that govern how the database operates.

Slide 33: Once again stress the importance of iteration and user involvement.

Slide 34: This slide gives the learning outcomes for this topic; ask students whether they have been met.

Slide 35: This slide gives a list of references which students may find useful.

3.5 Laboratory Sessions

The laboratory time allocation for this topic is 1 hour.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

These exercises use the students and courses tables created in the laboratory sessions for Topic 1.

Exercise 1: Creating a table from another table and further queries

1. Create a table called student_types by selecting the distinct values of the student _type column from the students table.

2. Add a column to the student_types table to record the fees that will be charged.

3. Make the student_type attribute of student_types the primary key. You should use the alter table command

4. Use the alter table command to make the student_type attribute on students a foreign key to the new student_types table

5. Update the student_types table to set the fees. Home students will pay 5000, overseas students will pay 10000

6. Show each student's first and last names and the fees that they are paying.

Suggested Answer:

1.
```
create table student_types as
(select distinct student_type from students);
```

2.
```
alter table student_types
add fees float;
```

3.
```
alter table student_types
add primary key (student_type)
```

4.
```
alter table students
add foreign key (student_type) references student_types(student_type);
```

Note: This syntax will vary depending on vendor. The above works for MySQL and SQL Server. Oracle will use the Add Constraint clause.

5.

```
update student_types
set fees = 5000
where student_type = 'Home Student';
```

6.

```
select first_name, last_name, fees
from students, student_types
where students.student_type = student_types.student_type
```

3.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Read all the relevant material for this topic. This should including the slides and slide notes along with the sections from at least one relevant textbook.

Ensure you have read and understood each of the steps in logical design that have been identified in the lecture. Prepare a presentation/discussion that outlines each of the steps of logical design. This could be either a PowerPoint presentation or discussion document.

Investigate the optional step (1.6) Specialise/Generalise in extra detail. This step relates to concepts in the Enhanced Entity Diagramming approach. Investigate this and discuss how this might be applied to the development process.

Be prepared to present this during the tutorial session.

Suggested Answer:

For each of the steps in logical design, as discussed in the slides and lecture notes, the students should be able to state what the purpose of that step is and what the main activities are of that step.

For the optional step (1.6) Specialise/Generalise:

- They should investigate the concepts in the Enhanced Entity Diagramming approach that this relates to. Entities can be classified as super and sub types. For example, using the Student table that has been used in the SQL sessions. This could have been modelled as a super type ('Student') that has two sub-types ('Overseas Student' and 'Home Student').
- The supertype would have all the attributes that are shared by all students (StudentID, First_name etc.) The sub-types would have only those entities that pertain to them. For example it might make sense to keep information about the Home Country for overseas students but not for Home Students. For Home Students we might keep their Local Authority as an attribute but this would not e make sense for Overseas students.
- During this stage the nature of the relationship between the super and sub-type would need to be investigated.
    - Participation constraint: Determines whether every member of a super class must be a member of a sub-class. In the example above then yes they do since a student must be home or overseas.
    - The Disjoint constraint: can a member of a super-class be a member of more than one sub-class? In the example above it is no since a student cannot be a home AND overseas student.

- During this optional stage such super and sub types are identified for the whole entity diagram. The decisions need to made as to how the super and sub types will be represented as tables in the database. The decision as to how they should be implemented will depend on the nature of disjoint and participation constraints. Briefly if a super-class must be a member of a sub-class then a single table is likely. Other considerations are:

    - The number of attributes a sub-class has that are separate from its super-type. The more attributes the more likely it is that the sub-type should be implemented as a separate table.

    - Does the sub-type have relationships of its own with other entities that the super-type does not participate in? If so then, again, it is more likely that it should be implemented as a separate table.

Exercise 2:

Refer back to the Marlowe Interiors case study that you have worked with in the private study exercises in previous topics.

Draw the full entity relationship model from previous partial models that have resulted from normalisation in Topic 2. Be prepared to discuss this in the tutorial session.

You should incorporate the additional materials shown below:

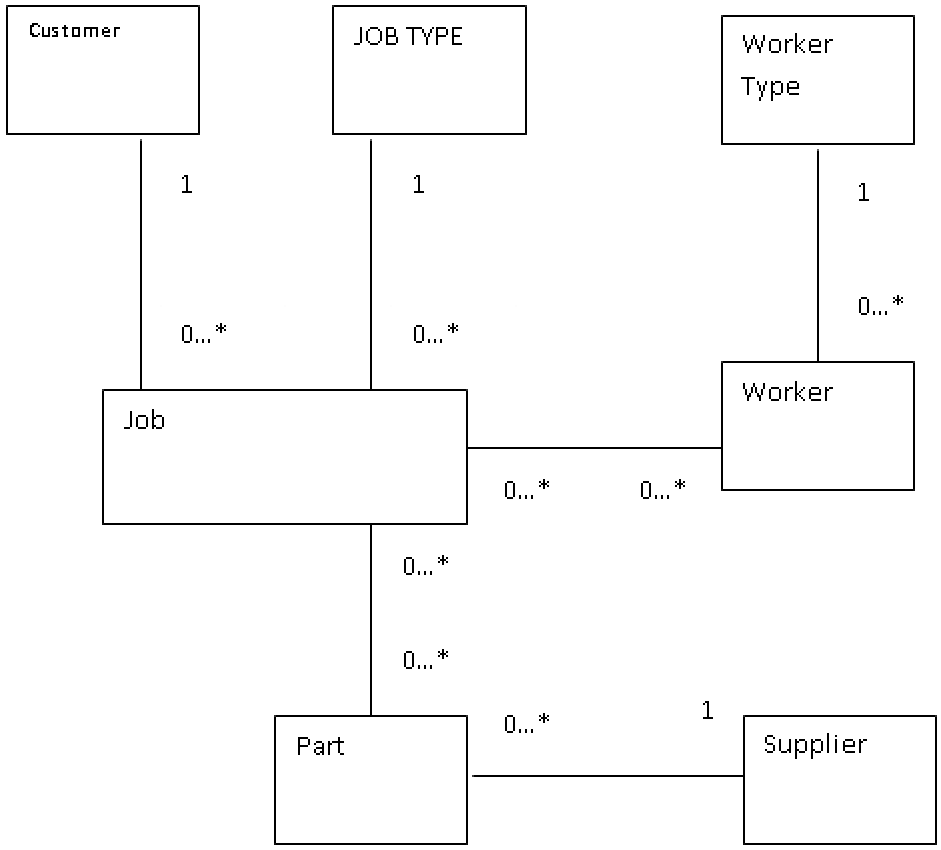
1st Document
Rates of pay for various types of worker

7. Type	8. Rate per Hour
Plumber	£50
Labourer	£20
Qualified Builder	£55
Interior Designer	£39
Electrician	£50

2nd Document
Example of a manual customer record kept on index files in the filing cabinet

Name: Jane MacDonald
Address: 23 Essex Road, N1 6YY
Telephone: 070 222 3333 or 0603 99933 (mobile) or 0208 99999 (work)
E-mail: janem@videotec.co.uk also jane@mc.eagle

Suggested Answer:



Note: Many-to-many relationships have not yet been resolved.

The two previous normalisations (the invoice and the job sheet) produced partial models of the system. These have been merged here and the additional material from the customer list and worker type added. Thus the rate of pay is a feature of the type of worker not the worker him or herself. Thus the entity is added.

3.7 Tutorial Notes

The time allowance for tutorials in this topic is 2 hours.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

Exercise 1:

In small groups, discuss your findings from Private Study Exercise 1 and develop a group presentation that gives an overview of the steps of logical design and in particular the 'specialise/generalise' step.

Each group should present their ideas and the information should be discussed as a whole class group.

Suggested Answer:

See Private Study Exercises.

Exercise 2:

Discuss Private Study Exercise 2 as a whole class group. How did you go about creating the entity relationship model? Did everyone come up with the same model?

Suggested Answer:

See Private Study Exercises.

Exercise 3:

Discuss the following as a whole-class group:

a. What is meant by an iterative approach to the development of databases?
b. Why is it important to involve users in the development of a database?
c. What do you understand by the concept of a candidate key?
d. What is meant by the concept of data redundancy?

Suggested Answers:

a. Iterative approaches involve going over steps. This means that work can be re-done and improved.
b. User involvement is important. In iterative approaches it means that users can give feedback on parts of the development in order to improve it. It is also important as a way of giving the users a stake in the system and a sense of ownership.
c. A candidate key is any group of attributes that is capable of uniquely identifying a row. A primary key is chosen from the candidate keys.

d. Redundancy is where relationships exist that can be derived from other sources. As well as being needless it can lead to conflicts of data.

Exercise 4:

In small groups go through each of the steps of logical design and suggest how these might be applied to the development of the system for the Marlowe Interiors.

Be prepared to discuss this as a whole-class group.

Suggested Answer:

Students should be encouraged to relate the tasks they have been involved in in the practical exercises with those they have studied in the lectures.

Step 1. Create and check ER model

Students will have had two approaches. The 'top down' approach of ER modelling. This has involved identifying entities from nouns etc. and drawing an initial ER diagram. The 'bottom up' approach of normalisation. Normalisation is both a check on the ER model and a means of discovering entities itself. Attributes are identified in a similar way. We might think of a 'top down' approach as investigating what the users are 'saying' about their data (such as in interview transcripts or scenarios). The 'bottom up' approach is looking at examples of data such as invoices, receipts etc.

The students will have been performing all these activities during the Marlowe Interiors development. The stepped approach outlined in the lecture, with the various sub-steps systematises this.

Step Two Map ER model to tables

Mapping the entities of the system requires the move from the final ER model to the table structures. Normalisation is part of this process too. Making sure that the new table structures fit the relational model. Names of tables and columns should be decided upon as appropriate. The student will be doing this as they implement the Marlowe Interiors database.

Topic 4: Data Retrieval 1

4.1 Learning Objectives

This topic provides an overview of the basic aspects of data retrieval. On completion of the topic, students will be able to:

- Implement more complex relationships;
- Describe how to retrieve data from one or more tables using SQL;
- Recognise and identify different data-types in SQL.

4.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practise the skills during the private study, laboratory and tutorial sessions.

4.3 Timings

Lectures:	2 hours
Laboratory Sessions:	1 hour
Tutorials:	2 hours
Private Study:	7.5 hours

4.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Table and view structure in a relational database
- Data types
- Null values
- Retrieving data using SQL

4.4.1 Guidance on the Use of the Slides

Slides 2–3: This topic will necessarily recap on some material covered on more general database modules. It will, however, examine it at a greater level of detail.

Slide 4: In the conceptual model as represented in the high level entity relationship diagram and/or catalogue of entities discovered during earlier stages of the development process, the relationship between entities was represented in a number of different ways as listed here. Tables to be created from these entities and relationships will now be examined.

Slide 5: In one-to-many relationships a candidate key (usually the primary key) becomes a foreign key at the many end of the relationship as shown here. It should be noted that this is also the way in which many-to-many relationships are resolved. In this example the many-to-many relationship would be between module and student: a student takes many modules a module is taken by many students. The creation of an intersection entity (Student on module) resolves it. In this way the relationships can be represented as tables.

Slides 6-7: A recursive relationship is where an entity has a relationship to itself. In this case a student has mentor who is also a student. A mentor can look after many students. This is represented in the table as shown with a ‘mentor id’ being added to the table; the mentor id is in fact a foreign key to the student id on the student table.

Slide 8: Resolving a one-to-one relationship usually depends on the optionality of that relationship as shown in these examples. But it is important to remember that the designer should always take account of the semantics of the relationship, that is the meaning of the relationship to the organisation or business.

Slide 9: In this case a student must have a portfolio and a portfolio must belong to a student. In this case there is a case for merging the two entities into a single table. Here the portfolio attributes become attributes on the student table.

Slide 10: Here it is the case that a student MIGHT have a portfolio but that a portfolio MUST belong to a student. The attributes for each of these two entities are kept separate and two tables (STUDENTS and PORTFOLIOS) are implemented.

Slide 11: Here the relationship is optional at both ends. A student MAY have a Studio, a Studio MAY be allocated to one student. The entities are kept separate and implemented as two separate tables. The decision as to where to place the foreign key depends on what makes more sense in terms of how to represent this data. In this case it would be sensible to place the foreign key on the studio to represent allocation of a room because Studios would be allocated but it might not be the case that most students would have a studio.

Slides 12-13: When entities are implemented, SQL is usually used to create them on a database. It is worth noting that there are other ways of carrying out the creation of database structures such as tables. For example MS Access uses a GUI to allow users to create tables; there are also CASE tools that use similar GUIs. However all database professionals need to understand how SQL creates database structures. It is usually the case that, for example a CASE tool, is providing a GUI front end and writing SQL create scripts in the background.

SQL is used also, and often primarily, as a query language.

Slides 14–16: Details of the main features of SQL DML.

Slide 15: ***Activity:*** Ask the students if they know what the main parts of an SQL select statement is doing.

Slide 16: ***Answer:*** The main parts of an SQL query are labelled here. They are comprised by the key words: SELECT, FROM, WHERE (plus additional 'Where' clauses added by the use of the keyword AND), GROUP BY, HAVING, ORDER BY.

Slides 17-18: Examples of the use of literals in SQL

Slide 19: This slide shows examples of selecting all of the columns in a table and a sub-set of them.

Slide 20 -21 : Example of order by clause. Note that, in accordance with the relational model, there is not theoretical ordering of tuples in a relation. In practice the underlying structure of a table will store rows in a particular order but this is at a level that should not be the concern of the developer in terms of design of applications such as queries. If there is the need to represent data in a particular order then the order by clause is used.

Note that the default is to order in the ascendant. This can be stated explicitly by using the ASC modifier. To order in the descendent the DESC keyword is used.

Slides 22-23: There are a number of aggregate functions that are available in SQL. Ask the students if they can identify the use for each of those listed in Slide 22. Slide 23 gives the answers.

Slides 24-25: Example of an aggregate function. Aggregate functions must be used with the GROUP BY clause since they are, by their nature, going to give a result that is valid for a whole group of rows rather than for individual rows. So, in this example, the Count is going to tally the number of members of each branch that has more than one member of staff.

Slide 26: The HAVING clause modifies a GROUP BY clause. Here the same query is shown the HAVING clause is where it determines that only those groups with more than one member of staff will be retrieved.

Slide 27: More complex queries are formed by the use of sub-queries. Sub-queries are queries that are nested within other queries. Note that only rows from the outer (main) query can be shown in the result.

Slide 28: It is more usual to use joins than sub-queries. With a join, data can be shown in the result from any of the tables used in the join.

Slide 29: This slide shows a list of common errors that are encountered when writing SQL select statements.

Slide 30: ***Activity:*** Ask the students what a data-type is. Ask the students what a domain is. Can the students give a list of data-types that are available in SQL?

Answers: A data-type is the most basic way of defining the valid values available for a column, for example whether it is a number or character. A more restrictive way of limiting the available values is to define a domain. A domain specifies a set number of values that are valid for a column (although in practice this could be a very large number). For example the values of a column primary_colours could be defined as belonging to a domain that contained only the values: Red, Blue, Green.

Slides 31-34: This slide recaps some of the basic data-types in SQL. Note that different vendors and versions of SQL have implemented many of their own data-types. This discussion limits itself to standard SQL.

Slides 31-32: String data-types. Note: bit allows values of 1 or 0.

Difference between Char and Varchar. Char pads out unused length with blank spaces, Varchar limits the length to the actual characters input.

Slides 33-34: Date and time are fairly self-explanatory but it is worth noting that formats for these vary depending on vendor. Timestamp stores a combined date and time. Interval represents a period of time.

Slide 35: Ask the students to recall what the definition of a null value is. What could a null in a database represent? Examples are: zero; a blank space; an unknown quantity. Because, fundamentally it represents ‘something not stored in this database’ and therefore unknown, it is not true that NULL = NULL. Since each of the nulls on either side of the equals sign could represent very different unknown values.

Slide 36: This slide gives the learning outcomes for this topic; ask students whether they have been met.

Slide 37: This slide gives a list of references which students may find useful.

4.5 Laboratory Sessions

The laboratory time allocation for this topic is 1 hour.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

These exercises use the Students and Courses tables created in the laboratory session in Topic 1.

Exercise 1:

Many important queries in a database involve using the aggregation functions COUNT, MIN, MAX, SUM and AVG

- COUNT counts the number of times something occurs in a database table, the number of rows that meet a particular condition.
- MIN finds the minimum or lowest occurrence of an attribute in a database table.
- MAX finds the highest occurrence of an attribute in a database table.
- AVG finds the mean average of an attribute in a database table.
- SUM finds the totals of all the values of an attribute in a database table

It is also useful to be able to use the aggregation functions with the group by and having clauses.

For example to show the number of students on each course:

```
select course_name, count(student_id)
from courses, students
where students.course_id = courses.course_id
group by course_name
```

We could modify this to shown only those courses with more than 1 student by using the having clause:

```
select course_name, count(student_id)
from courses, students
where students.course_id = courses.course_id
group by course_name
having count(student_id) > 1
```

Complete the following:

1. Show a count of students on each course and group by the course id
2. Using a sub-query show the first name, last name and age of the oldest student
3. Using a sub-query show the first name of the oldest overseas student
4. Write a script to count the number of students that are 30 years or older
5. Select the average age for students on a course. Show the course id and course name with this.

6. What is the total amount of fees being paid by all the students
7. What is the total amount of fees being paid by overseas students
8. Write a query that shows which course is the most profitable.
9. Modify the above query and use the ‘having’
10. What is the average amount of fees paid by people over the age of 20?
11. What fee is paid by the youngest student?

Solutions

1.

```
select course_id, count(course_id)
from students
group by course_id;
```

2.

```
select first_name, last_name, age
from students
where age = (select max(age) from students);
```

3.

```
select first_name
from students
where age = (select max(age) from students where student_type = 'Overseas Student');
```

Note: This query will return two rows as the two students have the same age.

4.

```
select count(student_id)
from students
where age > 29;
```

5.

```
select courses.course_id, course_name, avg(age)
from students, courses
where students.course_id = courses.course_id
group by course_name;
```

6.

```
select sum(fees) from
student_types, students
where students.student_type = student_types.student_type;
```

7.

```
select sum(fees) from
student_types, students
where students.student_type = student_types.student_type
and students.student_type = 'Overseas Student';
```

8.

There are various ways of doing this. This way shows each course with the amount of revenue it brings in. It is ordered with the most profitable at the top.

```
select courses.course_id, courses.course_name, sum(fees) from
student_types, students, courses
where students.student_type = student_types.student_type
and students.course_id = courses.course_id
group by courses.course_id
order by sum(fees) desc;
```

9.

```
select courses.course_id, courses.course_name, sum(fees) from
student_types, students, courses
where students.student_type = student_types.student_type
and students.course_id = courses.course_id
group by courses.course_id
having sum(fees) < 10000
order by sum(fees) desc;
```

10.

```
select avg(fees)
from student_types, students
where students.student_type = student_types.student_type
and students.age > 20
```

11.

```
select fees
from student_types, students
where students.student_type = student_types.student_type
and students.age = (select min(age) from students)
```

4.6 Private Study

The time allocation for private study in this topic is expected to be 7.5hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Read all the relevant material for this topic. This should including the slides and slide notes along with the sections from at least one relevant textbook.

You should prepare a PowerPoint presentation or discussion document that examines the following issues:

- The Purpose of SQL
- The difference between Data Definition Language (DDL) and Data Manipulation Language (DML)
- Parts of the Data Manipulation Language
- The Structure of a Select statement
- The function of the aggregate functions

Be prepared to present and discuss your ideas in the tutorial session.

Suggested Answer:

Students should demonstrate a clear understanding of the parts of SQL.

The purpose of SQL is embodied in the Data Definition Language (DDL) and Data Manipulation Language (DML) and their dual role. DDL is about creating data structures such as tables. DML is about performing operations on those structures such as queries and updates.

The structure of a select statement as outlined in Slide 16. The student should clearly demonstrate an understanding of each of these parts.

The various aggregate functions have been used in the SQL session for this topic. The students should demonstrate an understanding of their purpose. These are functions that apply to groups of data rather than single rows. Thus it makes sense that they are used with the GROUP BY clause since they are about examining multiple rows of data and making some kind of calculation on them depending on the aggregation function being used.

Exercise 2:

You have been asked to develop a further set of possible transactions for the Marlowe Interior system. List and give details of possible transactions that would be likely for this system.

You should think about aspects of the business that will need to be catered for by the new system. Marlowe Interiors should be able to keep data on their customers and add new customers if they need to. Customers' details might also change. The main part of their business involves keeping track of jobs, who is working on them and what parts are used. With all parts of their business there might be new data such as new workers, parts of suppliers. Marlowe Interiors also needs to generate various paper-based reports such as job sheets and invoices.

Think about these aspects of the business and others you have identified. While looking at the ER diagram, create a list of transactions that support these activities.

Suggested Answer:

Some examples of possible transactions are:

- Create new customers
- Set up a new job for a customer
- Assign workers to jobs
- Assign parts or materials to jobs
- Set up new parts
- Create a new supplier
- Create a new job type
- Assign workers to a worker type
- Generate an invoice for a customer
- Generate a report on jobs for a customer
- Delete obsolete Suppliers

4.7 Tutorial Notes

The time allowance for tutorials in this topic is 2 hours.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

Exercise 1:

In small groups, discuss your findings from Private Study Exercise 1 and develop a group presentation that gives an overview of the issues mentioned in the exercise.

Each group should present their ideas and the information should be discussed as a whole class group.

Suggested Answer:

See Private Study Exercises.

Exercise 2:

Discuss Private Study Exercise 2 as a whole class group.

Suggested Answer:

See Private Study Exercises.

Exercise 3:

a. Outline the ways in which one-to-one relationships can be represented in a database.
b. Explain what is meant by the term 'recursive relationship.'
c. Why is it important to resolve many-to-many relationships?

Tutorial Answers

a. This is discussed in the lecture. The options are shown in the slides 7 to 10 and depend on the type of optionality.

b. Recursive relationships are where an entity has a relationship to itself.

c. It is important because many-to-many relationships cannot be represented in the database since they could lead to data that is in contradiction. For example in the example shown in Slide 5 it could mean that a student been assigned to a module that was in contradiction to whatever module that student is assigned to.

Exercise 4:

Define a CRUD matrix for the new transactions that you have identified for Marlowe Interiors.

Define data types and domains for Marlowe Interiors system.

Suggested Answer:

The CRUD matrix will depend on the transaction chosen.

Below is an example of two transactions from the list given above.

Transaction /Table	Customer	Job Type	Job	Job part	Worker Job	Worker	Worker type	Part	Supplier
Create a new customer	C								
Generate an Invoice for a customer	R	R	R	R	R	R	R	R	

Data types and domains should be specified for the particular attributes. Students should be encouraged to think about what the range of valid values for an attribute is and match this accordingly.

One potential pit-fall is the use of letters and such in code. This means that it will be difficult to automatically generate these if that is what is required. Number data types should be used where mathematical operations need to used.

Topic 5: Data Retrieval 2

5.1 Learning Objectives

This topic provides an overview of data retrieval operates in more complex situations. On completion of the topic, students will be able to:

- Outline the concept of referential integrity and say why it is important in a relational database;
- Understand how to retrieve data from one or more tables using join;
- Understand how to retrieve data from one or more tables using sub-queries.

5.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practise the skills during the private study, laboratory and tutorial sessions.

5.3 Timings

Lectures:	2 hours
Laboratory Sessions:	1 hour
Tutorials:	2 hours
Private Study:	7.5 hours

5.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Referential integrity in relational databases
- Types of joins
- Retrieving data using joins
- Retrieving data using sub-queries

5.4.1 Guidance on the Use of the Slides

Slides 2–3: This topic is a continuation from the last topic and expands and adds to a number of concepts covered there.

Slide 4: This slide shows a summary of the concept of referential integrity. Ask the students why this is important? Think about what would happen if it wasn't the case i.e. if foreign keys were allowed to be unmatched. This would lead to being unable to join tables where necessary in queries. Data would be meaningless in the sense that essential pieces of data would not exist. Why should it be a candidate key (usually the primary key) that a foreign key is linked to? Because otherwise when a join between tables was made using the foreign key then if the foreign key linked to an attribute that was not a candidate key then the join could result in multiple values being retrieved where only one value should be retrieved.

Slide 5–6: These slides demonstrate the way in which primary keys and foreign keys link. Here the tables show data about students, module and the module that students take. Confirm that the students understand these basic concepts. For example they could be asked what modules a student is taking.

Slides 7–8: Here we recap on parts of the select statement. Ask the students to identify the function of each of the clauses of the select statement.

Slide 9 - 10: Joining over two tables. Slide 9 shows the creation of two tables.

Activity: Ask the students how they are linked.

Answer: They are linked by the foreign key on workers which shows which department a worker belongs to. When it comes to joining the two tables in an SQL query the foreign key is used.

Activity: Ask the students what would happen if the WHERE clause was left out of this query.

Answer: What would be retrieved would be every worker shown against every department regardless of whether they belonged to that department. This is known as the Cartesian product and is meaningless in this context. Therefore we join the tables using the foreign key as shown.

Activity: What is the result of this query?

Answer: It shows the department name, first name, last name and job title for each worker. The department shown will be the department the worker is in.

Slide 11: There are different sorts of join available in SQL. Explain each of these, and how they differ, to students.

Slide 12: Another example of a simple join on two tables.

Slide 13: Here is an example of joining more than two tables. Note the use of the AND keyword, which operates as an additional WHERE keyword when additional tables are added to the join. Each table must be joined to the relevant other table where they are linked by a foreign key. Obviously a clear understanding of the database structure is required here.

Note that when we start to join multiple tables in a database there is a potential for a performance problem. Every time a new table is added to the where clause there is a potential for making the query retrieve results more slowly. This is especially true in a database that has lots of rows on each of the tables. There are ways of dealing with this when it comes to physical design and implementation, for example the use of views and de-normalisation. These subjects will be examined in a later topic.

Slide 14: Outer joins: note that outer joins are implemented slightly differently in different vendors' versions of SQL.

Slides 15–17: These show the results of a standard join where there is a foreign key that is NULL. Arif Rahman has not been allocated to a department, so when the query is made then Arif Rahman is not shown in the result. It may be the case that this is entirely valid, that this is what is required from the query to show employees in the list only if they have been allocated to a department. However if we want a query that shows employees regardless of whether they have been allocated to a department or not then the use of an outer join is required.

Slide 18: There are different types of outer join available. Ensure that students understand each type.

Slide 19: This slide shows the result of a left outer join on the same tables shown earlier (in Slide 15). A left outer join retrieves all the rows from the first (left) table that are unmatched in the second (right) table. So all the employees are shown even Arif Rahman (ID 5) who has not yet been allocated to a department.

Slide 20: This slide shows the result of a right outer join. A right outer join shows all the results from the second (right) table that have unmatched rows in the first (left) table. In this case the department no 33 (Chemistry) does not yet have any employees allocated to it but is still shown in the result. Note that employee 5 is not shown.

Slide 21: This slide shows the result of a full outer join. A full outer join shows all unmatched rows in the query result set. In this case both employees that do not have a department and departments that do not have employees are shown.

Slide 22: As well as joins it is possible to retrieve data where rows are reliant on more than one table by the use of a sub-query. The example here shows retrieval of department information where we want departments that contain workers that are the maximum age of employees in our organisation. A sub-query is useful where we know how to search for a value but do not know what that value is. In this example we know how to search for the maximum age of our employees even if we don't know what it is.

Activity: Ask the students how we would write this query if we knew that the maximum age of all our employees was 45 for example.

Answer: We could just put the 45 into the query and do away with the sub-query altogether.

Sub-queries can use all the logical operators available in SQL such as =,>, < etc.

Slide 23: In some case the inner query is dependent on data retrieved in the outer query. For each row processed by the inner query the outer query must be processed as well. In this example we want to get a product name for products that are on the orders table. The inner query selects the product id from the orders table and the WHERE clause in the inner query joins the orders table with the products table that is in the outer query.

Slide 24: This slide gives the learning outcomes for this topic; ask students whether they have been met.

Slide 25: This slide gives a list of references which students may find useful.

5.5 Laboratory Sessions

The laboratory time allocation for this topic is 1 hour.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

These exercises use the Students and Courses tables that were created in the laboratory session for Topic 1. Some of the queries in this section are fairly complex and may require close guidance and support if students are to be able to construct them.

Exercise 1:

To investigate some of the joins we need to insert some new data.

1. Insert a new student into the students table.

ID	6
First_name	Arno
Last_name	Laski
Student_type	Overseas Student
Course_id	NULL
Age	25

Note that you will need to leave the course_id null.

2. Insert the following new course: course id = 4, course_name = 'Art'

Suggested Answer:

1.
insert into students (student_id, first_name, last_name, student_type, age)
values (6,'Arno','Laski','Overseas Student',25)

2**.**
insert into courses values (4,'Art');

Exercise 2:

Execute the following query:

```
select course_name, student_id, first_name
from students, courses
where students.course_id = courses.course_id;
```

Note that it does not bring back any of the new data that you have entered. This is because the basic join used here excludes any unmatched columns. To retrieve the unmatched columns then an outer join should be used.

1. Re-write the above query so that it shows even students who are not yet allocated to a course.
2. Write a query that retrieves the course_name and the students' first and last names and includes courses that do not yet have any students allocated to them.
3. To get both the courses without students and the students without courses requires the use of a full outer join. However this features is not always supported in implementations of SQL. One way of doing it is to use the UNION operator to join the two previously written queries. Write the query using the method available from your particular version of SQL.
4. Select all the students who are over the age of 20. Include those students who are not yet allocated a department
5. Now select all the students who are over 20 who are paying more than 6000 in fees. This will require the addition of a sub-query to the query in the previous question.
6. Select the course name and average age of students on that course. Include students who do not have a course.

Suggested Answer:

1.

```
select course_name, student_id, first_name
from students
left join courses on (students.course_id = courses.course_id);
```

Note: Syntax can differ between vendors. Above is from MySQL.

2.

```
select course_name, first_name, last_name
from students
right join courses on (students.course_id = courses.course_id);
```

3.

The MySQL version is:

```
select course_name, first_name, last_name
from students
left join courses on (students.course_id = courses.course_id)
union
```

```
select course_name, first_name, last_name
from students
right join courses on (students.course_id = courses.course_id)
```

4.

```
select course_name, student_id, first_name, age
from students
left join courses on (students.course_id = courses.course_id )
where students.age > 20
```

Note: If the query is constructed slightly wrongly as below it produces very strange results:

```
select course_name, student_id, first_name, age
from students
left join courses on (students.course_id = courses.course_id and students.age > 20)
```

5.

```
select course_name, student_id, first_name, age
from students
left join courses on (students.course_id = courses.course_id )
where students.age > 20
and student_type in (select student_type from student_types where fees > 6000)
```

6.

```
select course_name, avg(age)
from students
left join courses on (students.course_id = courses.course_id )
group by course_name
```

5.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Read all the relevant material for this topic. This should including the slides and slide notes along with the sections from at least one relevant textbook.

You should prepare a PowerPoint presentation or discussion document that examines the following topics with the use of examples:

- The necessity of joining tables.
- Simple joins
- Multi-table joins
- Types of Outer Joins

Be prepared to discuss this in the tutorial session.

Suggested Answer:

The students should demonstrate an understanding of the central aspect of joining tables in SQL. Joining tables means that data can be presented in a relational way and without needless duplication. It is a way of answering a user enquiry about their own data as embodied in a Select statement. If joins are not used and data is selected from more than one table then the result is a Cartesian product with duplicated data.

Simple joins are between two tables. The student should provide examples.

Multi-table joins are between more than two tables. The student should provide examples.

Outer joins. These are ways of joining tables so that data that is only represented in one table in the join is also included in the result set. The various types of outer join discussed in the lecture should be identified and understood.

Exercise 2:

In a fully developed database system, data is often accessed and manipulated through programs known as applications. These might be reports, screen based forms, web-pages or other types of tool.

Using the resources available to you, investigate the types of applications available for a database system. Prepare a discussion document about them.

Be prepared to discuss this in the tutorial session.

Suggested Answer:

A key aspect of this discussion is that students:

- Understand the distinction between the DBMS and the applications that access it
- Can identify the different sorts of applications available

Applications embody the *process* part of a system as opposed to the *data* of the system that is embodied in the database. As well as keeping data, database systems do something with that data in the sense that they use it to perform their business tasks. So, a Student Records System does not just keep data about students it enrols them on courses, generates reports about them, records grades etc. The applications here will alter the data in some way. So enrolling students on a course will mean creating a new row on a table that links students with courses.

The link to transactions should be clear here.

The different sorts of application available are listed above. The key distinction should be made between applications that can alter data (forms, web pages, batch processes) and applications that retrieve data (reports, either screen based or paper based).

Additional points:

- There are certain types of transactions that break down the strict barrier between applications and the database. Database triggers, for example, perform processing but are stored in the database.
- Also, some DBMS products (notably MS Access) come bundled with lots of application development software able to create form and reports etc.

Exercise 3:

Using Web resources, investigate the typical application software available for one of the following DBMS products and create a document that details this:

- Oracle
- MySQL
- MS SQL-Server

The format of your document is up to you, it could be a text based document, mind-map or other illustration.

Be prepared to discuss this in the tutorial session.

Suggested Answer:

My SQL

Works with PHP and Apache server so is used extensively for the development of web-sties that need database connectivity. It also has Application Programming Interfaces for number of programming language such as Visual Studio and Java which allow development of applications such as forms and reports.

MS SQL Server

Integrates well with many MS products. Much development of applications is carried out using Visual Studio and its supported language such as Visual Basic and C++.

Oracle

Has a whole range of native development tools such as Oracle Forms and Reports. It also now has support for Java development.

5.7 Tutorial Notes

The time allowance for tutorials in this topic is 2 hours.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

Exercise 1:

In small groups, discuss your findings from Private Study Exercise 1 and develop a group presentation that gives an overview of the topics mentioned in the exercise.

Each group should present their ideas and the information should be discussed as a whole class group.

Suggested Answer:

See Private Study Exercises.

Exercise 2:

In small groups, discuss your findings from Private Study Exercise 2 and develop a group presentation that gives an overview of the types of applications available for database system.

Each group should present their ideas and the information should be discussed as a whole class group.

Suggested Answer:

See Private Study Exercises.

Exercise 3:

In small groups, discuss your findings from Private Study Exercise 3 and develop a group presentation that gives an overview of the types of applications available for database system.

Each group should present their ideas and the information should be discussed as a whole class group.

Suggested Answer:

See Private Study Exercises.

Exercise 4:

a. Give a definition of referential integrity and say why it is important.
b. What role do foreign keys play in the construction of queries?
c. What is the difference between a sub-query and a join?
d. What is a correlated sub-query?

Suggested Answer:

a. A foreign key must refer to a candidate key (usually a primary key) elsewhere in the database or must be null.
b. Foreign keys are used to join tables so that queries do not perform Cartesian product joins.
c. A sub-query is a query nested within another query. A join select rows from multiple tables.
d. A correlated sub-query is a query when the inner query references an attribute in the outer query.

Exercise 5:

Define the joins and scripts for all transactions that have been defined for Marlowe Interiors.

Suggested Answer:

The scripts will depend on the transactions designed. Relevant examples can be found in the solutions to some of the SQL lab exercises For example if the transactions defined for Marlowe Interiors require being able to insert a new customer (as it should) then this can be extrapolated from already written scripts for inserting a customer as shown in the laboratory session for this topic where an insert is written for students.

It should be pointed out that some of the more complex transactions, such as the production of an invoice, might involve a level of SQL that students will only attain towards the end of the module. Nevertheless students should be thinking about what tables and attributes would need to be involved in this transaction and how they will be joined by foreign keys.

Topic 6: Physical Design 1

6.1 Learning Objectives

This topic provides an overview of physical design. On completion of the topic, students will be able to:

- Understand the purpose of physical design;
- Map a logical database design to a physical database design;
- Design tables for the chosen database product.

6.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practise the skills during the private study, laboratory and tutorial sessions.

6.3 Timings

Lectures:	2 hours
Laboratory Sessions:	1 hour
Tutorials:	2 hours
Private Study:	7.5 hours

6.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- The purpose of physical design;
- Mapping the logical database design to a physical database design;
- Designing tables for the target DBMS;
- Creating tables using SQL.

6.4.1 Guidance on the Use of the Slides

Slides 2–3: This topic serves to introduce the purpose of physical design and to help understand the movement from logical to physical design.

Slides 4–5: Ask the students if they can recall the main aspects of each of the specific design phases that have been introduced.

Slide 6: The main purpose of conceptual design is stated here. Emphasise that conceptual design is, in a sense, a ‘pure’ investigation into the data needs of an organisation and/or system. This entails the identification of entities, attributes etc. but independently of any particular model. So the investigation here does not even assume the relational model.

Slide 7: The main purpose of logical design is stated here. The data is now designed in terms of the chosen model. In the case of this course, that is the relational model with all that this involves in terms of activities like normalisation. This is the activity most commonly thought of when we have been talking about ‘database design’ up to this point.

Slides 8-9: The main purpose of physical design is stated here. The database is designed with the target Database Management System (DBMS) in mind. Therefore it entails knowledge of the chosen DBMS whether it is Oracle, MySQL, SQL Server etc. It is here that a multi-disciplinary approach might be needed in terms of who is involved with the physical design. It might be the case that in-house or available experts in the chosen DBMS might be Database Administrators as well as Database Developers. It is also worth noting at this point that the features of a chosen DBMS (as outlined very broadly in Slide 10) are often undergoing a process of change. New features get added and older features become obsolete. Therefore part of the task of a database professional is to keep up to date with the evolution of the DBMS. This will involve consulting vendor’s websites, press releases, technical papers and attending conferences. Such self-development, although essential, is by no means easy since it is time consuming and often seen as not the main task of the database professional.

Slide 11: Some points about how to choose a vendor. Often though there is little choice in the sense that an organisation might have committed to a particular technology because of systems that have been built earlier.

Slide 12: The steps outlined here continue the steps that began with the logical design as outlined in the approach taken in Connolly and Begg [Database Solutions]. Note however that the steps will not be taken in the same order. These steps provide a fairly good working taxonomy (classification) of the various tasks undertaken but it is often easier to understand them by choosing a slightly different grouping of them as has been done in this module.

Summary of Step 3: The aim of step 3 is to convert the outcome of the logical model into designs that fit with the chosen vendor's DBMS. Base tables will come from the entities that have been modelled in the ER model and as the result of normalisation.

Derived data will be covered in Topic 7. Derived data is data that has its source in other columns in the database. For example the total for an order can be derived from all the sub-totals of the items that go to make up that order. Decisions need to be taken as to whether such a total should be stored on that database and worked out from the sub-totals or whether it can be calculated at run time for example in an application such as a report.

Business rules are covered in Topic 8. Business rules are rules about the way an organisation goes about its operations. Many such rules can be represented in the normal constraints that exist in a database for example referential integrity and domains. However there may be a number of rules of a type that require more complex representation. For example if we have and order in a sales system that can only have up to 10 items. There are different ways of representing this.

Slide 13: Step 4: This step is concerned with analysing the file structures and choosing how they will be organised. It will be covered in detail in Topic 9.

Activity: Ask the students if they understand what is meant by the term transaction.

Answer: A transaction is a logical unit of work; although it way contain a number of operations they should all be grouped such that if one operation fails the whole transaction fails. Transactions were investigated as part of logical design but now they are investigated further and to a greater degree of detail with the target DBMS in mind.

Indexes are ways of improving the performance of queries on a database. They can make searches quicker but there is usually a trade-off. The analogy to an index in a book is useful. Finding something in a book is much easier if there is an index rather than having to go through the whole book from start to finish looking for what you want.

Slide 14: Step 5 to Step 8: These steps cover a number of specific tasks which should be carried out as part of physical design. All these steps are covered in the various topics as specified.

- *User views*: Views are dynamic structures that are the result of a query on the database. They are a way of permanently storing the result of a query so that it responds to any changes in the database. They can improve performance, usability and security.

- *Security mechanisms*: These will be aspects such as who has access to which part of the database. This will involve the design of user roles and permissions. These can be enforced using SQL.

- *Controlled redundancy:* Redundancy means storing data or relationships between data that exist elsewhere. This can sometimes improve performance.

- *Monitor and tune the operational system*: This involves knowledge of the DBMS and how to use it to make it work more efficiently. The managing and tuning of the operational systems is not covered specifically. Tuning the database refers to manipulating the set up and operational parameters of a particular database product. This could refer to things like the way queries are performed. With any database product the number of parameters would be very large (in the hundreds). Knowing the details of the particular product is a highly skilled role within the database world usually the province of a Database Administrator (DBA). As a database developer it is important to liase with the DBA during development to ensure that the design of the physical database takes advantage of the features of the database product.

Slide 15: Collate information about tables and columns that has been collated during logical design: This will be in the data dictionary and the entity relationship diagram. Note that at this point it is more proper to start talking about tables and columns rather than entities and attributes. When specifying attributes, domains will be defined. If new domains are developed at this point then these should be documented also.

Slides 16–17: The outcome of logical design will be a set of entities as documented in the entity relationship diagram. This example is of the Art Supplies database discussed in an earlier topic. A customer has ordered a number of items; items are supplied by a supplier. Slide 16 shows the attributes defined in logical design with the primary keys specified.

Slides 18-19: Record the physical design structures for these tables: Shown here is the example of the Customer table using DBDL (Database Design Language) which is closer to the eventual DDL (Data Definition Language) embodied in SQL.

This is an initial definition. Further aspects will become apparent as the physical design profgresses.

For each table:

- Table name
- List of columns
- Primary key and foreign keys. Any alternate keys
- Referential integrity constraints for each of the foreign keys that have been identified

For each column

- The domain of that column including the data-type, length and any additional constraints that apply to that column
- A default value for the column
- Whether the column is derived and if it is derived then how it is computed

- Whether or not the column can contain null values

Ask the students if they can write the SQL to define this structure.

Slides 20-21: This slide shows data definition in SQL. Writing these sorts of data definitions should be something that the students should become familiar with. These will be practised during the workshops.

Slide 22: It is worth noting that in many large projects a CASE tool might be used. This could generate SQL scripts from entity and relationship diagrams that have been defined in the CASE tool repository.

Slide 23: Note that different vendors implement SQL DDL (Data Definition Language) in slightly different ways. Here is an example of two ways in which primary keys can be defined. It is also worth mentioning that in some vendors implementations it is perfectly possible to define a table WITHOUT a primary key.

Activity: Ask the students why this is a problem.

Answer: For one thing, it completely violates the relational model. It means that duplicate rows are possible.

Slide 24: An example of defining a foreign key. Foreign keys in the physical design embody referential integrity defined in the logical model. As was discussed in the logical design a relationship ”many” will become a foreign key. For example with the courses and student database used in the SQL sessions: a course can have many students so the primary key of ’Course’ becomes a foreign key on the Students table. Foreign keys are defined as shown in the example here.

Once again there is nothing to stop a table being created without the foreign keys being defined. These can be added later. As discussed below.

Slide 25: An example of explicitly creating a domain. Note that there are a number of ways of enforcing domains as well as this explicit way. These will be discussed in a later topic.

Slide 26: This is an example of altering a table after it has been created.

Activity: Ask the students what they think the advantages might be of being able to define foreign key constraints on tables after the tables have been created.

Answer: It means that if the data model changes and a new relationship between tables is established then these can be added to the physical design and the physical implementation. Remember what was said much earlier about the design process: that it is an iterative process, whereby changes might happen at almost any stage in the process. Note different vendors have their own differences in syntax.

Slide 27: An example of adding a column later.

Slide 28: There are in fact many ways of modifying structures once they have been created in SQL. Stress that this allows the developer the freedom to make changes and so allows a more iterative approach to development.

Slide 29: This slide gives the learning outcomes for this topic; ask students whether they have been met.

Slide 30: This slide gives a list of references which students may find useful.

6.5 Laboratory Sessions

The laboratory time allocation for this topic is 1 hour.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

You will need to monitor students carefully during this exercise and provide guidance if they are unsure about what is required.

Please note that you may also need to give students some guidance on what is required for Private Study Exercise 2 during the laboratory session.

Exercise 1:

You should create the tables that you have developed as part of the Marlowe Interiors example you have been working on in the previous tutorial.

Suggested Answer:

The exact structure of the tables will vary.

Foreign key constraints can be defined as part of the table definition or using the alter table command. If the student is not sure where and how to enforce the foreign key constraints they can wait to a later workshop but should be aware of diligence needed when inserting data.

Below is an example:

```
create table customer s
(Customer_id int not null auto_increment primary key,
Customer_First_name varchar(30) not null,
Customer_Last_name varchar(30) not null,
Address_line1 varchar(30),
Address_Line2 varchar(30),
Postcode varchar(10));

create table job_types
(job_type_code varchar(6) not null primary key,
job_type_description varchar(30))

create table workers
(Worker_id int not null auto_increment primary key,
 Worker_First_name varchar(30) not null,
 Worker_Last_name varchar(30) not null,
 Address_line1 varchar(30),
 Address_Line2 varchar(30),
 Postcode varchar(10),
 Worker_type_code varchar(6));

create table jobs
(job_id int not null auto_increment primary key,
start_date date,
end_date date,
customer_id int,
job_type varchar(6))

alter table jobs
add foreign key (customer_id) references customers(customer_id);

alter table jobs
add foreign key (job_type_code) references job_type(job_type_code);

create table worker_types
(worker_type_code varchar(6) not null primary key,
worker_type_description varchar(30),
hourly_rate float))

create table workers
(Worker_id int not null auto_increment primary key,
 Worker_First_name varchar(30) not null,
 Worker_Last_name varchar(30) not null,
 Address_line1 varchar(30),
 Address_Line2 varchar(30),
 Postcode varchar(10),
 Worker_type_code varchar(6));

alter table workers
add foreign key (worker_type_code) references worker_types(worker_type_code);

create table worker_jobs
(worker_id int not null,
job_id int not null,
start_date date not null,
end_date date,
primary key (worker_id, job_id, start_date))
```

Note: Different syntax for multi-column primary key

```
alter table worker_jobs
add foreign key (worker_id) references workers(worker_id);

alter table worker_jobs
add foreign key (job_id) references jobs(job_id);
```

```
create table suppliers
(supplier_id int not null auto_increment primary key,
 Supplier_name varchar(30) not null,
 Address_line1 varchar(30),
 Address_Line2 varchar(30),
 Postcode varchar(10));


create table parts
(part_id int not null auto_increment primary key,
 part_name varchar(30) not null,
price float,
 supplier_id int)

alter table parts
add foreign key (supplier_id) references suppliers(supplier_id);

create table job_parts
(job_id int,
part_id int,
quantity int,
primary key (job_id, part_id));

alter table job_parts
add foreign key (job_id) references jobs(job_id);

alter table job_parts
add foreign key (part_id) references parts(part_id);
```

6.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Read all the relevant material for this topic. This should including the slides and slide notes along with the sections from at least one relevant textbook.

You should prepare a PowerPoint presentation or discussion document that gives an overview of the following topics:

- The Purpose of Physical Design
- The Knowledge needed of chosen DBMS
- Steps 3 to 8 of design process (i.e. the steps of Physical Design).

Suggested Answer:

The Purpose of Physical Design

The students should demonstrate a clear understanding of the purpose of physical design in terms of recognise that knowing the target DBMS will impact the design process. Whereas in logical design (where the data model is taken into account but not the chosen DBMS) the process of design can proceed in a 'pure' sense in conformity with relational (or other) theory; in physical design the possibilities, limitations and constraints of the physical system must be taken into account. This requires a knowledge of the chosen DBMS.

Knowledge of the Chosen DBMS

A full knowledge of the chosen DBMS is required. How close a fit to the chosen data model exists i.e. if it is a relational DBMS how much of the relational model is implemented in the DBMS? What 'flavour' of SQL is supported? How close does it conform to standard SQL? What applications fit easily with the DBMS? For example Oracle has its own application suite and is also integrated with Java. MySQL meshes well with PHP. SQL Server meshes well with VB.NET. What are the parameters that can be 'tuned' with the particular database? There will usually be lots of different parameters for example concerning the type of query optimisation (which works out how to perform queries in the best way)

Steps 3 to 8 of design process

For each of these steps the students should know the purpose of the steps and the main activities of each of these steps. This is outlined in the slides and slide commentary.

Exercise 2:

For each of the 3 areas discussed above, devise 5 quiz questions (so you should have a total of 15 questions), along with the answers to these questions. These will be used in the tutorial to conduct a quiz based on the topics studied.

Suggested Answer:

Students should be told to develop questions that reflect the knowledge of the topic at the level of the lecture and activity above. It would be a good idea if the questions and answers were prepared on index cards or sent to the lecturer by email or through an eLearning tool. You can then prepare the quiz based on the questions the students have devised, selecting appropriate questions and eliminating questions that are too similar.

Exercise 3:

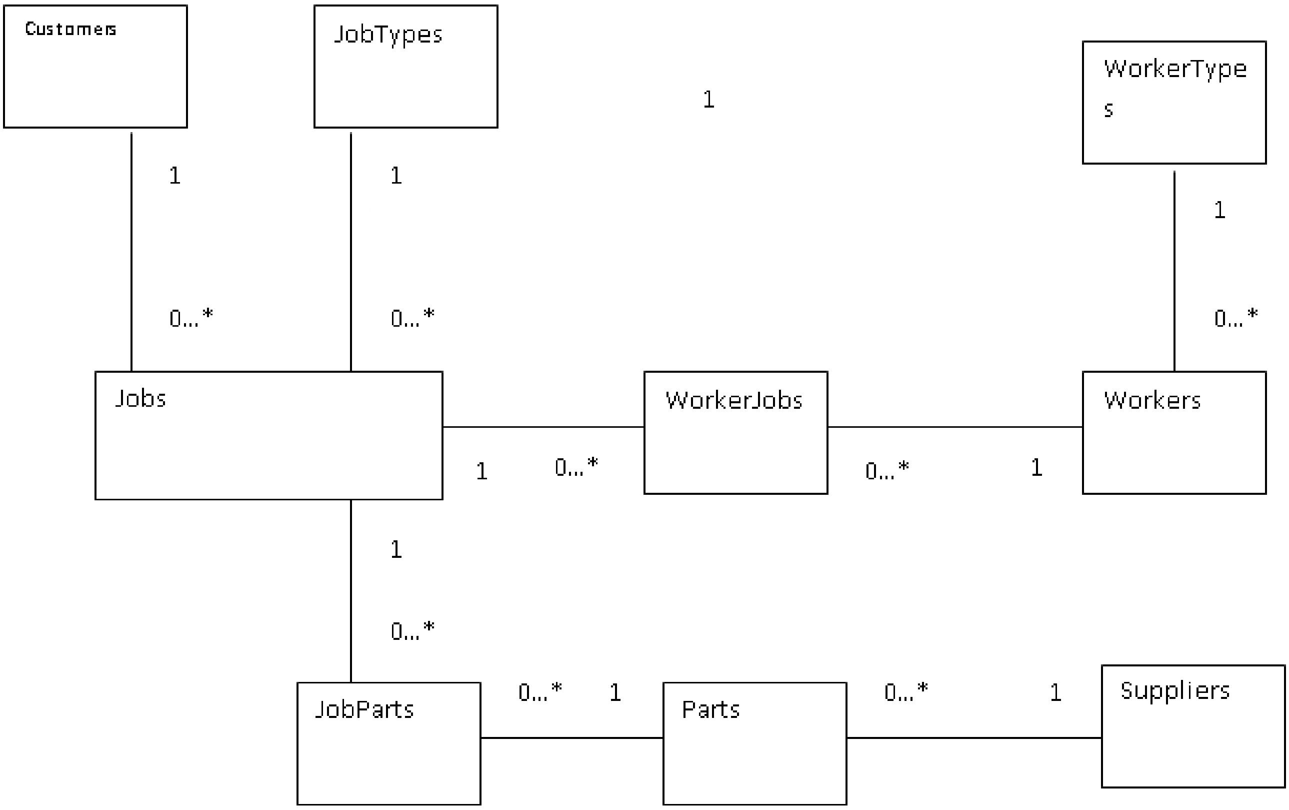
Provide a documented mapping of the logical database design to the physical design for Marlowe interiors.

All tables and possible domains should now be defined:

a. Produce a table design document. This could be a modified ER document.
b. Produce a document of each table in the format shown in the slides.

Suggested Answer:

Each table should have a set of suitable attributes in the format shown in the slides.



6.7 Tutorial Notes

The time allowance for tutorials in this topic is 2 hours.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

You may prefer to run a team quiz instead of Exercise 2 below if you have been able to collect the questions in advance.

Exercise 1:

In small groups, discuss your findings from Private Study Exercise 1 and develop a group presentation that gives an overview of the topics mentioned in the exercise.

Each group should present their ideas and the information should be discussed as a whole class group.

Suggested Answer:

See Private Study Exercises.

Exercise 2:

In small groups, using the questions that you wrote for Private Study Exercise 2, select the five best questions under each heading from your group. Each group should take it in turns to ask the rest of the class their questions.

Exercise 3:

In small groups, discuss your findings from Private Study Exercise 3.

Each group should present their ideas and the information should be discussed as a whole class group.

Suggested Answer:

See Private Study Exercises.

Exercise 4:

a. What is the difference between a data type and a domain?
b. Why is it important to have a knowledge of your chosen DBMS when performing physical design?
c. How can CASE tools aid the physical design process?
d. Why is it useful to be able to alter a table using SQL after it has been created?

Suggested Answers:

a. A domain is a more restricted limiting of allowable values than a data-type.
b. It is important because: the developer will need to implement their designs in the actual database product; different vendors implement particular structures in different ways.
c. Automate aspects of the design and development process.
d. To rectify mistakes; to allow an iterative design process to take place that means changes can be made in response to user feedback.

Exercise 5:

Write the SQL create scripts for each of the tables in Marlowe Interiors.

Suggested Answer:

Examples of the scripts can be found in the SQL laboratory exercise answers for this topic. Students own scripts will depend upon what attributes they have defined. These scripts can be used as a guide.

Topic 7: Physical Design 2

7.1 Learning Objectives

This topic provides an overview of derived data. On completion of the topic, students will be able to:

- Understand the concept of derived data
- Design a representation of derived data
- Recognise the trade-offs between different ways of implementing derived data

7.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practise the skills during the private study, laboratory and tutorial sessions.

7.3 Timings

Lectures:	2 hours
Laboratory Sessions:	1 hour
Tutorials:	2 hours
Private Study:	7.5 hours

7.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- The concept of derived data
- Designing a representation of derived data

7.4.1 Guidance on the Use of the Slides

Slides 2–3: This topic looks at the concept of derived data. ‘Derived’ means obtained from a source. So derived data is data that is derived from a source elsewhere in the database.

Slide 4: Specifically derived data is defined as a column whose value is derived from the value of one or more other columns in the database. These might be columns within the same table or columns from one or more other tables.

Slide 5: This table represents a summary of academic classes in a college. Each class has a topic, a room, and the number of students in that class.

Activity: Ask the students which of these columns might be derived from data elsewhere in the database? How might it be derived?

Answer: It is likely that the NoOfStudents column is derived from elsewhere. Students will be allocated to academic classes and this would be stored. By counting up the number of students that are listed against a particular class then the NoOfStudents in that class is calculated.

Activity: What might be the problem if the users were simply allowed to enter into the database a value for NoOfStudents?

Answer: There is then no way to guarantee that the value of NoOfStudents in the AcademicClasses table actually matches the number of students who are in that class. The data risks compromising the database’s integrity.

Slides 6-7: Derived data can sometimes be seen as the responsibility of application design rather than database design. For example the total number of students in a class might be seen as being part of a report on the database rather than something in the database itself. In this way sometimes derived data is not always represented in the data model but rather becomes part of application design.

Where there is a danger that the derived data will not be properly documented and therefore ‘lost’ as the design of the database progresses then it is better to document it.

The other danger with derived data is that columns that are actually derived are taken as being intrinsic parts of the database i.e. as non-derived fields. So NoOfStudents would be a normal field which allowed input and therefore all the problems of risking the integrity of the database would follow.

Slides 8-9: Shows a representation of the derived column of number of students on the AcademicClass table. The method by which the data can be derived has been added here as pseudo-code.

Activity: Ask the students if they know what pseudo-code is?

Answer: High level representation of an operation in a computer system that is designed to be read and understood by people rather than machines. It can omit much of the formal syntax of the programming language in which it will eventually be implemented.

Activity: Ask the students what the COUNT function is doing here.

Answer: The COUNT function is counting the number of students that belong to a particular class. The join is using the Topic column which is the FK on student and the PK on AcademicClass. In this way the calculation for noOfStudents is correctly linked to what class they are in.

Slide 10: ***Activity:*** Ask the students to think about organisations and businesses. They might be familiar with some that they have worked in as well as colleges, supermarkets, retailers. What documents or applications in these businesses might have data on them that is derived data?

Slide 11: ***Answer:*** It is worth pausing to think about this. Something like an invoice in an organisation usually contains lots of derived data such as totals for the number of products ordered. When looking at documents in an organisation such as an invoice the temptation is to model them as entities during logical design but they often contain much derived data and can even be implemented as applications. When looking at documents during normalisation we sometimes come across derived data.

Slide 12: At this point it should be remembered that database design is an iterative process and looking at, and for, derived columns and thinking about how they are derived is often a good way of validating aspects of our database.

Slide 13 -14: Slide 13 shows the original Art Supplies shop customer order form. Slide 14 shows fields that have been added to make this a more usable form. There is now an ’order id’ which is used to uniquely identify each order. For each order line there is a sub-total. The sub-total is calculated by multiplying the item price by the number of items ordered (the quantity). There is also a total field which is used to add up the total cost of the items ordered.

Slides 15-16: It is assumed that a customer might make many orders over time. Therefore it makes much more sense to organise the items that a customer has ordered into a separate entity Order. The modified ERD is shown in Slide 16.

Slide 17: To derive the totals for each item ordered then the price needs to be retrieved from the Item table and multiplied by the number of Items ordered on the OrderItem table. Remember this is pseudo-code and not the physical implementation.

Slide 18: To derive the total for the order itself then we need to use the aggregated function SUM to add up the ItemTotals for each of the items that belong to that Order.

Slides 19–21: These slides show how this works physically on the database. Note that this could be a calculation that is performed in an INSERT statement or might be part of the table definition in the CREATE TABLE statement.

Slide 22: Using derived data has an overhead on the database. Wherever the calculation is done, whether in the insert statement or attached to the table definition or in a database trigger, it will still need to be performed when the relevant operation is carried out on the database.

There is also the possibility that data can become inconsistent. For example if a derived column also allows the users to update it manually.

Slide 23: As well as storing derived data in a column there is the possibility that it is embedded in an application and calculated at run-time.

Slides 24–26: This slide shows how a derived column will be calculated at run-time.

Slide 27: The list shown on this slide is the aggregate functions in SQL that can be used to calculate a derived column or during an application. Also the normal mathematical operators can be used.

Slide 28: This shows the standard SQL for the calculation of a derived column in an SQL statement. This column will be stored on the database. Whenever a new row is inserted into this table. Please note that different vendors implement this differently.

Slide 29: This shows a generated column that is being generated when data is inserted into the table. A table employee_yearly_earnings has two columns (employee_id, yearly_salary). The yearly salary is calculated as 12 times the (monthly) salary.

Slide 30: A decision will need to be made as to how derived data is implemented in the database. Complicated calculations that are performed many times at run-time could affect the performance of the database. If a derived attribute is hardly ever updated then there would not be much of a performance overhead in storing it as a column on the database. Otherwise it might be better to calculate it in an application as and when necessary.

Slide 31: Derived fields do not only have to be numerical. So, for example, selecting a particular item may mean that a certain colour is set as the colour of that item. Derived data like this is best handled in something like a database trigger, which is a piece of procedural logic stored on the database that operates (‘fires’) when a specific database operation such as an insert or update is performed.

Slide 32: This slide gives the learning outcomes for this topic; ask students whether they have been met.

Slides 33-34: These slides give a list of references which students may find useful.

7.5 Laboratory Sessions

The laboratory time allocation for this topic is 1 hour.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1: Inserting data into the Marlowe Interiors database.

Data should be inserted into the database for Marlowe Interiors. Referenced tables will need to be populated first. You should insert a representative sample of data for example about 10 customers, 10 jobs etc.

Please note that multiple values can be inserted in the format:

```
insert into courses values
(5,'Biology'),
(6,'Psycology'),
(7,'Music')
```

Where data is inserted into a table where there is an auto increment or sequence on the primary key, then the other rows should be specified and inserted and the primary key will be automatically generated.

```
insert into workers (worker_first_name, worker_last_name, worker_type_code)
values ('Fred','Smith','DEC')
```

Exercise 2: Practising comparison operators

SQL has a number of comparison operators that lets the user perform takes where a simple equality statement ("=") will not do.

- LIKE – where the condition is that the value is similar to the specified condition
- IN – where a column value is equal to any of the values that are set out in a list supplied in the query
- BEWEEEN … AND – specifies a range of values
- IS NULL – specifies that a condition where the rows retrieve has a null value in that column

Run this example against the student table you created in the first workshop.

```
select * from students
where first_name like 'D%'
```

This example retrieves the student details where a first name begins with the letter 'D'. Note the use of the wild card operator '%': this is used to mean any value.

See what happens If you try to do the following using the equality operator:

```
select * from students
where first_name = 'D%'
```

Now use the tables from the first workshop to complete the following exercises.

Exercise 3:

1. Select all the students whose last name begins with ‘S’.
2. Select all the students whose last name begins with ‘S’ and whose first name begins with ‘H’
3. The format of the BETWEEN statement is of the form:

   Where <column_name> Between <first_value> And <second_value>

   Use this to select all the students who are between the ages of 25 and 30.
4. The format of the IS NULL statement is to simply write ‘is null’ in the where condition.

   Use this to select the first name and last name of all overseas students between 19 and 25 who do not have a course (i.e. their course_id will be null).
5. Use the IN operator to select course information from the courses table that you created in the lab session for Topic 1, but only for a set of specific courses.
6. Now, using this structure and joining with the student table, select the first and last names, and the course name, of all the students who are studying History or Geography.
7. Modify the last statement to select the first name and last name, and the course name for all students studying History and Geography whose first name starts with the letter ‘H’.

Suggested Answer:

1.

```
select * from students
where last_name like  'S%'
```

2.

```
select * from students
where last_name like 'S%'
and first_name like 'H%'
```

3.

```
select * from students
where age between 25 and 30
```

4.

```
select first_name, last_name
from students
where student_type = 'Overseas Student'
and age between 19 and 25
and course_id is null
```

5.

So to select the course_id and course_name from just the arts courses you write:

```
select * from courses
where course_name in ('Art','Music')
```

```
select first_name, last_name, course_name
from courses, students
where courses.course_id = students.course_id
and course_name in ('History','Geography')
```

6.

```
select first_name, last_name, course_name
from courses, students
where courses.course_id = students.course_id
and course_name in ('History','Geography')
and first_name like 'H%'
```

7.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Read all the relevant material for this topic. This should including the slides and slide notes along with the sections from at least one relevant textbook.

You should prepare a PowerPoint presentation or discussion document that gives an overview of the following topics:

- What is derived data?
- The type of documents and applications which use derived data.
- Storing derived data as a database field or view vs. deriving at runtime.

Be prepared to discuss this in the tutorial session.

Suggested Answer:

The students should demonstrate an understanding of the definitions given for derived data as outlined in the lecture slide and commentary. The vital concept here is that derived data is data that can be calculated or deduced from other data that already exists in the database. The classic example of this is an invoice. An invoice is usually a bill for some service or product that contains details of the service or product, customer information, itemised listings of the products or services provided and a final total. All these attributes can be derived from other attributes. When first modelling data, students often decide that 'Invoice' should be an entity but as its attributes are usually derivable then it is possible to that it can be implemented not as a table but via a view or report.

The trade-offs when storing derived data on the database as say a stored derived attribute are that on the one hand this can improve performance in terms of querying but can also slow performance when it comes to updating data.

Exercise 2:

Collect as many documents as you can that contain fields that show examples of derived data. These could be receipts, order forms, application forms, invoices or customer order forms.

For a selection of at least 3 of these documents you should identify:

- Derived attributes
- The source of the derived attribute (i.e. in what other attributes)

- The method that has been used to derive any derived attribute. (i.e. what operations are performed to come up with the result).

Suggested Answer:

The students should bring in something like a receipt or index. On a receipt, for example, each item purchased might be listed along with its price. The receipt itself need not be stored as a data structure in the database since the list of prices and item details would be stored as attributes on other tables.

Where something like VAT is added this is also calculable rather than stored since it would be done with a calculation based on the rate of tax against the cost of items.

Students should recognise that when it comes to developing their own systems that there will be processes such as the production of a receipt or invoice that involve derived data and so can be implemented in some of the ways discussed in the lecture.

7.7 Tutorial Notes

The time allowance for tutorials in this topic is 2 hours.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

At this stage of the module, you should also introduce the assessed assignment to students. Assignments for the relevant assessment cycle are available from the NCC Education Campus (http://campus.nccedu.com). You will need to ensure that each student has a copy of the assignment and understands the requirements. Assignments would normally be submitted for marking during Topic 9 or 10, depending on how much time you feel you need for marking.

Exercise 1:

In small groups, discuss your findings from Private Study Exercise 1 and develop a group presentation that gives an overview of the topics mentioned in the exercise.

Each group should present their ideas and the information should be discussed as a whole class group.

Suggested Answer:

See Private Study Exercises.

Exercise 2:

In small groups, discuss the documents that you found when you completed Private Study Exercise 2.

This should then be discussed as a whole-class group.

Suggested Answer:

See Private Study Exercises.

Exercise 3:

a. Give an explanation of derived data and explain the different ways it can be represented in a database.
b. What are trade-offs for the different ways of representing derived data?
c. What types of aggregate functions can be used in deriving data?
d. Explain how non-numeric derived data could be used in enforcing rule in a database.

Suggested Answer:

a. Derived data is any data the value of which can be derived by examining the value of some other data item. Usually some kind of operation will need to be performed on the source data to transform it into the correct derived data format. Derived data can be represented as a stored field in the database (either on a table or a view); or it can be derived at runtime in a calculation in an application such as a screen or report.

b. When storing in a table there is the additional costs of storage. There will be a new calculation every time a value in the source field is changed. There is also the possibility of data becoming inconsistent. When calculating at run-time it might mean joining multiple tables, which could have an impact on performance.
c. The list of standard SQL aggregate functions as shown in Slide 27.
d. The example in the slides is where a business rules specifies that one value depends on another in a way that is not numeric such as a part being of a standard colour, then this can be set.

Exercise 4:

Design and specify the requirements for at least two derived attributes in the Marlowe Interiors system.

Suggested Answer:

The examples could include the totals for labour, parts and final totals of the jobs. They can be added to the relevant tables for documentation purposes. The final derivations will most likely happen in an application such as a report.

Examples of how these will be derived are shown in the SQL sessions.

Topic 8: Physical Design 3

8.1 Learning Objectives

This topic provides an overview of database constraints. On completion of the topic, students will be able to:

- Define different types of constraints;
- Understand how to design and implement constraints on their chosen DBMS.

8.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practise the skills during the private study, laboratory and tutorial sessions.

8.3 Timings

Lectures:	2 hours
Laboratory Sessions:	1 hour
Tutorials:	2 hours
Private Study:	7.5 hours

8.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Defining integrity constraints on tables;
- Define integrity constraints for target DBMS;
- Applying integrity constraints;
- A look at some addition database structures;
- The importance of security.

8.4.1 Guidance on the Use of the Slides

Slides 2 -3: A ‘Constraint’ in everyday language is a restraint or constriction, a limitation on something. In database terms it is a rule that restricts the database from being in a disordered state. Some of this topic covers areas that have already been touched on when talking about various rules of the relational model such as entity and referential integrity.

Slide 4: This topic looks at the different sorts of constraints that exist and how they are designed physically for the target DBMS and examples of their implementation. For some of the constraint types there will alternative ways of implementing them.

Slide 5: ***Activity:*** Ask the students to recall the Entity integrity rule and the referential integrity rule.

Slide 6: ***Answer:*** The enforcement of the Referential integrity rule will be examined in some detail below.

How is the Entity integrity rule enforced? In the relational model if something is specified as a Primary Key then it cannot contain a NULL value. However the relational model is a theoretical construct that is implemented in different ways by different vendors. Defining something as a PRIMARY KEY on most vendor’s databases will enforce the rule that no part of it can be NULL. However, as previously noted, it is possible to create tables that do not have primary keys. If, for example in Oracle, we created a table and enforced the Primary Key with a Unique index this would not enforce the Entity integrity rule as it allows NULLS.

Activity: Ask the students: what is the purpose of the Entity Integrity rule?

Answer: It is there to guarantee uniqueness of the primary key since a NULL value might be the same as another NULL value (since null can mean unknown’.)

Slide 7: ***Activity:*** Ask the students to look at the Art Supplies data model. Where will the constraints that enforce the referential integrity rule have to go?

Answer: Wherever there is a foreign key then the referential integrity rule will need to be enforced. Foreign keys exist wherever there is a ‘many’ end to a relationship plus in selected other circumstances such as recursive relationships or one-to-one relationships.

In the case shown here Order would need to have a foreign key enforced to Customer. OrderItem would need to have a foreign key enforced to Order and Item. Item would need to have a foreign key enforced to Supplier.

Students should appreciate how we have moved from logical designs (entities and relationships) to physical table and integrity designs with our target database in mind so that we can specify in SQL how the constraints will be established.

Slide 8: This shows the Create script for the Customers table which does NOT have any foreign keys in it. Note that the primary key has been enforced here. The Customers table, because it will be referenced by other tables, is specified as a Source or Parent table to whatever table references it with a foreign key.

Slide 9-10: Examples of referential integrity constraints. Slide 9 shows the Orders table which references the Customers table, hence it can be designated as a Referencing or Child table. Note here the definition of the foreign key.

Slide 11: ***Activity:*** What happens if we delete an item from our Art Supply database?

Answer: There will be OrderItem records that reference these Items. If we delete the Item it might be the case that there is then a foreign key in the OrderItem table which refers to a row in the Items table that does not exist. We will be breaking the referential integrity constraint.

There are various options we can put in place to deal with this.

Slide 12: On this slide there is an example of a table defined with a foreign key and the propagation constraints specified. Propagation constraints define what will happen if the referenced column is altered in some way. In this case, rules are specified if a delete or update are made.

Slide 13: The options for propagation constraints are:

- **No action** means the record in the table with the foreign key is left as it is.

- **Cascade** means that any change (including a delete) is replicated in the table with the foreign key in it.

- **Set Default** means that the change in the parent table (in this case OrderItem) causes the record in the child table to be set to some sort of default. For example an Item could be deleted and in the OrderItem record the foreign key is set to some value such as ‘X’ which indicates that the record refers to an Item that no longer exists on the system.

- **Set Null**. Similar to Set Default except that the table with the foreign key has that foreign key set to Null.

Slide 14: Domains remember are the allowable values that are specified for particular columns. Some aspects of domains enforced through the user of data-types. However strictly speaking domains have a more limited range of values. Therefore they need to be defined more precisely. There are a number of options for enforcing domains on a database.

Slide 15: Check constraint: Defined in the Create Table statement. Used if the domain is unlikey to change.

Slide 16: Standardised versions of SQL allow the creation of structures in the database that are separate domains. Here is an example. Although this is fairly systematic, again it assumes that domains will not be subject to much change.

Slide 17: If the values of the domain are likely to be dynamic then it would be better to implement them as a look-up table. This will allow users to easily add, delete or update domains without having to rely on the expertise of a database professional. For example if the domain was recorded in a separate table then a form could be built which allows users to add, update or delete records in that table.

Slide 18: Recap on definition of business rules. It should be stressed that the 'real world' situation is vital in understanding business rules.

Slide 19: An example of a business rule in the Art Supplies database. The business rule states that no order must have more than 10 items. Note that there is no way of knowing this business rule without an investigation into the business. It might be the case that this rule is not even mentioned by the users until late into the design. This is the advantage of taking an iterative approach.

Slide 20: Shows how this business rule might be enforced in the SQL create table statement.

Slide 21: Another example.

Slide 22: More complex examples of business rules might need to be enforced by using database triggers. Not every vendor supports these. The trigger is created as a structure on the database. It is a piece of procedural code (programming) that is connected to another database structure, usually a table.

Activity: Ask the students if they understand the concept of procedural code in programming.

Answer: An instruction to the computer (in this case the DBMS) to perform some operation in accordance with the logical instructions that are specified in the code.

The definition of a trigger states under what circumstances it should come into operation (when it should 'fire').

Slide 23: This shows an example of a trigger. This trigger checks to see if an order has reached 10 OrderItems. If it has then an error is raised.

The advantage of triggers is that it is possible to use the full range of instructions available in a third-generation programming language. So for example there are LOOPS and IF statements in triggers. Some vendors develop their own languages for their database triggers. Oracle has PL/SQL.

Slide 24: There are additional database structures that the database developer needs to be aware of.

Slide 25: A view is a stored query result that is represented as a virtual table. It is a way of showing commonly occurring queries. It is a way, also, of allowing a user to see those parts of the database that they need to see in the way that they need to see it.

Slide 26: This shows the SQL for creating a view.

Activity: Ask the students if they can guess what the purpose of this view will be. What data do they think it will hold?

Answer: This will show a summary of the Order information and the Items that belong to that Order. Note the need to join the tables in the query.

Slide 27: Indexes. There will be more on indexes in Topic 9.

Slide 28: Sequences are useful for generating values for unique identifiers (primary keys for example). Sequences are created on the database. The default is that they start at one and increment. However it is possible to start them at a higher number.

Sequences are implemented in different ways by different vendors. Oracle calls them ’sequences’. In MySQL a column can be defined as auto_increment, which means it will select from an inbuilt sequence defined for the column that the developer need not define themselves.

Slide 29: ***Activity:*** Ask the students why they think security is important in a database system.

Answer: The students should have a basic understanding of issues related to security of both systems and databases. Data is vital to the basic operations of many organisations and the loss of it could mean loss of business, lots of confidence and/or credibility.

System security relates to the security of systems as a whole. Systems compromise applications, networks etc. as well as the databases that they access.

Database security explicitly relates to a database.

Slide 30: With regards to access to database systems it is important to recognise that not every user is the same. Users will need to access different parts of a system in different ways depending on their job role within the organisation and/or the level of authority that they hold.

Slides 31-33: SQL has facilities for the management of roles. Examples are shown here.

Slide 34: This slide gives the learning outcomes for this topic; ask students whether they have been met.

Slide 35: This slide gives a list of references which students may find useful.

8.5 Laboratory Sessions

The laboratory time allocation for this topic is 1 hour.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

During this session, you will need to split the class up into small groups in order for them to complete Private Study Exercise 1 prior to the tutorial session.

Exercise 1:

1. Setting a date automatically and working out the length of a job. A date can be generated automatically. This is an example in from MySQL:

```
insert into jobs (start_date, job_type_code)
values (current_timestamp,'KIT')
```

Now set the end date to sometime in the future:

```
update jobs
set end_date = ('2012-7-04')
where job_id = 1
```

Using the datediff function in MySQL (or similar functions in other vendors versions of SQL) it is possible to derive the length of a job in a select statement:

```
select job_id, datediff (end_date, start_date) job_length
from jobs
where job_id = 1;
```

2. How much a job costs. The building blocks. Insert some values into the worker_jobs table.

```
Insert into worker_jobs values (1,1,'2010-01-01','2011-01-01')
```

The next query joins the worker_jobs table with the workers table and then the worker_type table.

It works out how many days have been spent on the job by doing the date difference function. It then multiplies this by the hourly rate and then by the number of hours spent per day on a job (in this case 12). It makes many assumptions (such as weekend working!) but it is the beginning of the building blocks for deriving data for producing a full invoice for a job, for example.

```
select worker_jobs.worker_id, workers.worker_type_code, worker_types.hourly_rate,
worker_jobs.start_date, worker_jobs.end_date, datediff(worker_jobs.end_date,
worker_jobs.start_date) days_on_job, datediff(worker_jobs.end_date,
worker_jobs.start_date)
```

```
* 12 * hourly_rate labour_cost
from worker_jobs, worker_types, workers
where worker_jobs.worker_id = workers.worker_id
and workers.worker_type_code = worker_types.worker_type_code;
```

Try to get this query working using the data and structures in your own version of the Marlowe Interiors database.

3. Attempt to write a similar query for working out the cost of parts used on a job.

Suggested Answer:

For part 3:

```
select job_parts.job_id, job_parts.part_id, job_parts.quantity, job_parts.quantity * parts.price
parts_cost
from job_parts, parts
where job_parts.part_id = parts.part_id
```

8.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Read all the relevant material for this topic. This should including the slides and slide notes along with the sections from at least one relevant textbook.

Your lecturer should have divided the class into small groups. In your groups, you should prepare quiz questions based on the topics listed below. You should also provide a set of answers. The questions and answers should be supplied to your lecturer prior to the tutorial session.

Try to aim for at least two questions per bullet point as outlined below:

a. Rules

- Entity Integrity
- Referential Integrity
- Propagation constraints
- Domain constraints
- Table constraints

b. Types of propagation constraint:

- No action
- Cascade
- Set Default
- Set Null

c. Ways of enforcing constraints.

- As rules on tables
- Using the alter table
- Check constraints
- Domains

Suggested Answer:

Each group should produce 2 questions per bullet point plus answers. These can be submitted to you by email or by an eLearning tool prior to the tutorial session when the quiz takes place.

8.7 Tutorial Notes

The time allowance for tutorials in this topic is 2 hours.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

You should also allow time during the tutorial to check that students are working on their assignments and answer any general questions on the expected scope of the work. You may also wish to remind them of the submission deadline and documentation requirements.

Exercise 1

Your lecturer will run a class quiz based on the questions produced as part of Private Study Exercise 1.

Suggested Answer:

See Private Study Exercises. You should choose appropriate questions and answers to compile the quiz. All the topics listed are covered in detail in the lecture notes.

Exercise 2:

a. What are the different ways of enforcing a domain constraint in a database management system?
b. What are the ways a business rule can be enforced in a database management system?
c. What are database triggers?
d. What is a sequence in a database system and what is it used for?

Suggested Answer:

a. Check constraint; separate domain structure; as a foreign key to a separate table.
b. Many ways. Built into the structure of the database through the normal constraints e.g. a business rule like 'Every person must have an address' would mean that address fields are specified as not null. A rule like 'A student must be enrolled on a course' would be a non-null referential integrity constraint (foreign key). More complex business rules can be enforced with check constraints as shown in the example where no order has more than 10 items. Applications can also be used to enforce business rules through the logic of whatever programming language there are built in. Database triggers can also be used to enforce business rules.
c. Database triggers are pieces of procedural logic that are attached to database objects that operate ('fire') upon some event happening such as a new row being inserted into the database.
d. A sequence is a structure that generates a number one after the other. It is used to populate fields such serial numbers.

Exercise 3:

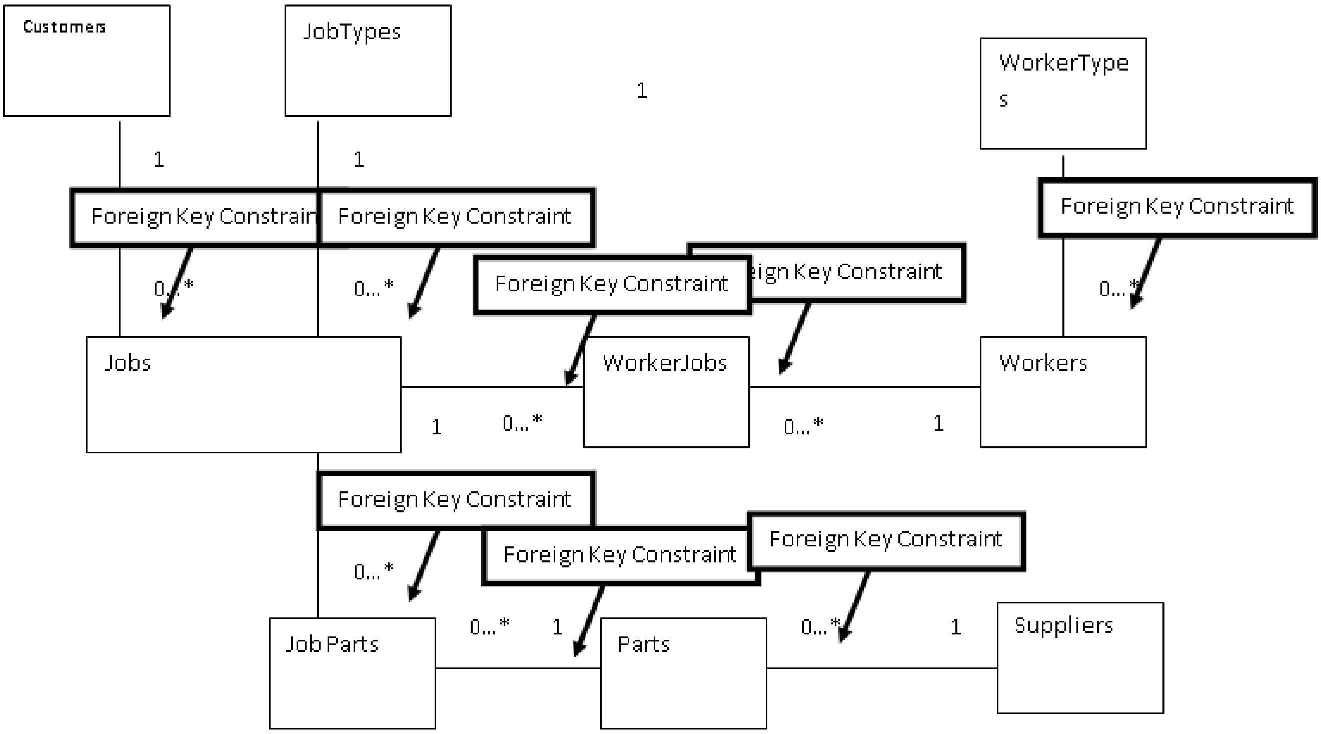
Define all the constraints for the Marlowe Interiors system.

Suggested Answer:

Any particular business rules should be documented. Foreign key constraints should be defined if they have not already been done.

Constraints in Marlowe Interiors:

- Defining constraints. All entities will have a Primary Key constraint. Foreign key constraints will be enforced whenever there is a 'many' end to a relationship. Remember that constraints can be enforced when tables are created or afterwards with the 'alter table' command.
- See diagram below.



Exercise 4:

Outline which propagation rules would be appropriate for the Marlowe Interiors database.

Suggested Answer:

Most cases where there is a child record deletion to the referenced key would not be allowed. For example a Customer could not be deleted (so losing the primary key) where it is referenced on the Jobs table.

If the primary key is changed, then it would be usual to cascade the change so that the reference on the child record remains e.g. changing the Worker Type code would change it on the Workers table.

Topic 9: Physical Design 4

9.1 Learning Objectives

This topic provides an overview of transactions and related issues. On completion of the topic, students will be able to:

- Define transaction use
- Understand the concept of de-normalisation
- Understand the use of indexes
- Estimate the size of a database

9.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practise the skills during the private study, laboratory and tutorial sessions.

9.3 Timings

Lectures:	2 hours
Laboratory Sessions:	1 hour
Tutorials:	2 hours
Private Study:	7.5 hours

9.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Understanding transactions
- De-normalisation
- Improving performance using indexes
- Estimating the size of the database

9.4.1 Guidance on the Use of the Slides

Slides 2–3: Transactions have been defined in logical design. It is crucial to stress the importance of transactions in any understanding of databases. Databases are not just about what is stored in them but the operations that take place, what it is that the organisation for whom the database is designed, actually does.

Some early (and rather zealous) advocates of databases tended to argue that in understanding an organisation all that mattered was to understand the data that the organisation held. In this sense a hospital was not any different from a car manufacturer. However, a more balanced approach is to appreciate that the processes that happen within an organisation need to be taken into account when thinking about the design of the database itself.

This approach is often necessarily iterative in the sense that users refine their understanding of how they will use new systems as they are developed.

Transactions can affect the performance of a database and discussions of de-normalisation and indexing are relevant here.

Finally estimating the size of a database is also essential to understanding how well it will operate and how well it will perform various transactions.

Slide 4: It is necessary to make a distinction between transactions and operations. Operations are the basic units of action that take place with regard to database structures such as tables and columns. They can be classified using the types listed here.

Transactions are made up of one or more of these operations.

The concept of atomicity is important here. A transaction is a logical unit of work that should not be divisible into its constituent operations.

Slides 5 - 8: Many of these questions will have been answered as part of logical design. However it is important to stress the need to examine each transaction in turn with regards to the latest stage of physical design.

Slide 9: This slide once again refers to the Art Suppliers database as discussed in previous lectures.

Slide 10: A list of representative transactions that have been identified for the Art Suppliers database. Ask the students to jot this list down.

Slides 11-12: The students should recognise the CRUD matrix. Ask them to re-draw the CRUD matrix with the operations for the transactions in it.

Slide 13: It is possible and sometimes necessary to examine transactions at the attribute level. For example if there was a transaction to change the price of an Item this might affect only the price attribute on the Item table not all the other attributes.

Slide 14: This slide shows the likely occurrences of data or the likely number of rows. From the number of rows on some tables it is possible to work out the number of rows on other tables if an estimate can be made of the ratio between them. On this diagram this is shown as a number between and average (AVG) and a maximum (MAX). The numbers of rows on this diagram are estimated based on an average. So, Customer has 1000 rows. If, on average, there are 2 orders for each Customer then there will be on average (2 x 1000) 2000 rows on the order table. For each Order there is, on average, 2 items so (2 x 2000) there is on average 4000 items on the OrderItems table. Because OrderItems is an intersection table we now need to move from the opposite direction towards it. There are 10 suppliers and each supplies on average 4 different items. So, (4 x10) there are on average 40 items. We can work out the average number of times an Item is on the OrderItems table (4000 / 40) is 100.

These sorts of estimates become useful later when thinking about sizing the database.

They are useful here for examining how many rows a transaction will affect.

Slide 15: An example of Transaction map. The bold arrow indicates the tables that are affected by the transaction.

Slides 16-17: For each transaction a number of aspects need to be documented. These can be documented in the Transaction Analysis Form shown in Slide 17

Slide 18: Performance refers to the amount of time transactions take. As well as queries this can refer to transactions containing other operations such as updates, inserts and deletes. In the discussion that follows the focus will be on query response as the most common focus of performance and one which exemplifies the most important issues. Note that in the discussion on indexes that follows; queries and updates (including insertions and deletions) can be affected in different ways. Often an index that improves performance of a query can actually be slow to update.

Transactions will normally be measured against some benchmark agreed on with the users. As databases grow in size the number of rows which need to be sorted through increases and this has the potential to slow down the transaction. It is worth bearing in mind that a wait of 30 seconds will seem fairly long for the users. Think of the response time when using the web as analogous.

Slide 19: Indexes are a way of increasing the performance of a query. They can be thought of operating in the same way as an index in a book. If someone wants to find a particular topic in a book then one way of doing it would be to start on page one and turn each page until the desired topic is found. This is not a very efficient way to go about locating the topic unless the book has very few pages. Using a book index is usually more efficient since the index just contains the topic titles and page numbers and will be much shorter in page numbers than the whole book.

Indexes in databases operate in much the same way. They are separate files that contain location pointers to the actual rows in the database. Instead of a topic title like in a book index they usually contain one or more attributes that belong to a row. They could, for example, contain just customer numbers from a customer table, rather than all the data for the customers as the actual database contains.

It is worth remembering though that there is no natural order to rows in the relational model and while particular physical implementations of the model will have an internal structure this will not necessarily correspond to any meaningful order, such as alphabetical or sequential. The attribute that the index is constructed around is the search key or indexing field.

Slide 20: The columns most used in joins are chosen because these are most often used in queries in situations where performance can be affected. Remember that every time a table is joined to another in a query it takes more time to perform the query. If an index can be used in this situation the chances are that it will make the query run quicker.

Columns used for ordering can also be indexed to improve performance. In this case the ordering operation can use the index to do the sort rather than the rows in the database table.

Slide 21: A list of types of index is shown here. A database table can have:

- ONE Primary Index OR Clustering Index
- AND Several Secondary Indexes

Slide 22: The primary index here uses a key in the table that is used to do the ordering of that table. The key here must be unique.

Slides 23-24: Secondary indexes uses are listed here. Note that the improvement in the performance of queries can be offset by slowing the database operation when updates are carried out.

Slide 25: Features of clustering indexes are listed. They behave much as primary indexes but are built around fields that are not unique. If a clustering index was built around Customer Name then there could not be a unique index built around Customer ID on the same table. This would remove the need to have a secondary index on Customer Name.

Slide 26: SQL examples of creating indexes.

Slide 27: Use the index of a book again as an analogy.

Activity: Ask the students to imagine a book that allowed new pages to be added, and indeed, new topics to be added. What would need to be added to the index to keep it up to date?

Answer: Every time a new topic was added then a new entry would have to be made in the index. This is what also must happen to a database index every time a new record is added, changed or deleted from a table. Secondary indexes have a particularly marked effect because they are built around non-unique field and they are stored in an un-clustered way. So imagine a new customer is added then the reference to their row must be inserted into the secondary index table built around the Customer Name. Any additional secondary indexes will also have to have this operation occur.

Slides 28-29: De-normalisation is another way of improving performance. De-normalisation involves using replication in a controlled way. Attributes that do not belong to a particular table but are often retrieved with that table in queries are replicated on that that table.

There is an example here from the Art Suppliers scenario.

Slide 30: Looking again at the data model with numbers of rows on it gives us the starting point for sizing the database. Sizing is important for estimating the hardware, network traffic and performance of transactions. The initial estimate of the number of rows is shown.

Slide 31: A number of factors need to be taken into account when estimating size. A database is not a static repository of data but a dynamic store which is constantly changing and usually increasing in size. Part of the design process will involve asking questions about the nature of the business as it affects the size of the database. Particular organisations might have periods where there is rapid increase in the number of rows on the database. Ask the students if they can think of some examples. Possible answers: a college that enrols students every September; a holiday company that makes lots of bookings at certain times of the year etc.

The organisation should also have an archiving policy. Records that have been inactive for a certain amount of time might be deleted or more likely transferred to a copy of the database that is used for archiving purposes. Again, think of a college: after students have left then their records can be archived; they are no longer 'live' students in the college but their records should be available should they need them in the future. This archiving policy will have an effect on the growth rate of the database and should be factored in.

Slide 32: This topic brings to the end the set of topics that has focused on the design process. Topics 10 and 11 discuss particular areas around different sorts of database implementation namely distributed databases and data warehouses.

It is worth thinking finally about the development process and how it draws to a close.

There are still tasks that need to be taken account of.

- *Data loading:* How the mass of existing data gets into the databases is an issue that needs to be thought about. This might involve large numbers of workers sitting and typing in the data from paper files. They will need to have standards that tell them how the data should be structured and there will need to be quality control to make sure that mistakes in data input are kept to a minimum. Many databases today are not, in fact, a replacement of paper based filing systems. Rather they are a replacement for older electronic databases systems. In this case there will need to be programs or tools that can be used to extract the data from one system and put it into the next. Many vendors have developed their own tools to do this.

- *Maintenance:* The database will need to be kept in good working order. This will involve making sure the data is kept in a clean state and does not become corrupted or replicated. It will also mean making sure that the DBMS is working to the best of its ability. When new versions of the DBMS are released, the database will have to be integrated with these.

- *Expansion and Change:* The database will most likely get bigger from its initial size. This may have an impact on size and could require database tuning (resetting the parameters of the DBMS). There is also likely to be change. This could result from the user requirements changing, a change in legislation, or expansion of the organisation into new areas of business. Some changes are of significant size so that they become mini development projects in their own right. They way also require changes to existing structures of data as well as building new ones.

- *Lifespan:* Very few database systems live forever! Sometimes the changes in the business and environment of an organisation can become so great or cumulative that it makes more sense to begin a new database rather than try to change an existing one. Also there are changes in the paradigms in the computing world. Relational databases have been pretty much the dominant model for twenty-five years but prior to that there was a succession of models (hierarchical, network etc.). There are models that have come after relational (object-oriented, object relational) which in some types of business such as design have replaced the relational model.

- Does development ever end? Is it ever complete? In the sense that there are always changes to make then one is tempted to answer no to these questions. Even when there is stability there is still maintenance that might require change. However in terms of the experience of the developer then development 'ends' when they move on to another project having delivered a finished system that meets the users' requirements. Having done so they can be content that they have completed the development project.

Slide 33: This slide gives the learning outcomes for this topic; ask students whether they have been met.

Slide 34: This slide gives a list of references which students may find useful.

9.5 Laboratory Sessions

The laboratory time allocation for this topic is 1 hour.

> **Lecturers' Notes:**
>
> Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Any foreign key constraints that were not enforced earlier should now be implemented.

Note that in the past there have been problems with creating check constraints in MySQL where they have been created but then ignored.

Implementing check constraints to enforce business rules.

1. An example of a business rule might be that if a quantity of parts is inserted into the database then it should be greater than 0.

```
alter table job_parts
add check (quantity > 0)
```

2. More complex constraint to enforce a domain:

```
alter table customers
add constraint check (address_line2 in ('London','Birmingham','Manchester'))
```

3. More complex constraints can be added by using triggers. These are specific to individual vendors.

You should add any constraints that you have defined for the Marlowe Interiors database.

Exercise 2: Queries on the Marlowe Interiors database

1. Create a query that shows all the jobs of a particular type
2. Create a query that shows customers who have more than one job
3. Create a query that shows which jobs a particular part is used on
4. Create a query that shows all the parts supplied by a particular supplier
5. Show all the jobs in a particular region

Suggested Answer:

Details will vary depending on data on students’ version of database.

1.

```
select * from job_types, jobs
where jobs.job_type_code = job_types.job_type_code
and job_type_description = 'Kitchen'
```

2.

```
select c.customer_last_name, j.customer_id, count(j.customer_id)
from jobs j, customers c
where j.customer_id = c.customer_id
group by j.customer_id
having count(j.customer_id) > 1
```

Note use of table aliases here to avoid ambiguity of customer_id. Customer_id to be counted must be from job table where each occurrence represents a different job.

3.

```
select j.job_id, j.job_type_code, p.part_name, pt.part_id
from jobs j, parts p, job_parts pt
where pt.part_id = p.part_id
and pt.job_id = j.job_id
and part_name = 'Flange'
```

4.

```
select p.*, s.supplier_name
from parts p, suppliers s
where p.supplier_id = s.supplier_id
and s.supplier_name = 'Scallions'
```

Note use of p.* to select all the columns from the parts table. It is a combination of the p table alias and the * wildcard operator.

5.

```
select j.* from jobs j, customers c
where j.customer_id = c.customer_id
and c.address_line2 = 'Leeds'
```

9.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Read all the relevant material for this topic. This should include the slides and slide notes along with the sections from at least one relevant textbook.

You should prepare a PowerPoint presentation or discussion document that gives an overview of what a transaction is and what the basic operational units of a transaction are.

Be prepared to discuss this in the tutorial session.

Suggested Answer:

The students should understand that a transaction is a series of operations that work on the database. The ACID properties of a transaction are: atomicity, consistency, isolation, durability. These should each be explained.

- *Atomicity:* a transaction must be an indivisible unit. So, all the operations that go to make a transction should be performed or none at all.
- *Consistency:* a transaction must move the database from one consistent state to another, not leaving it with duplication or loss of integrity.
- *Isolation:* transactions act independently of one another. The operations of one transaction should not interfere with those of another.
- *Durability:* the effects of the completed transaction are permanently stored on the database.

The operations of a transaction are: create, retrieve, update, delete. These should be explained and demonstrated with a CRUD diagram.

Exercise 2:

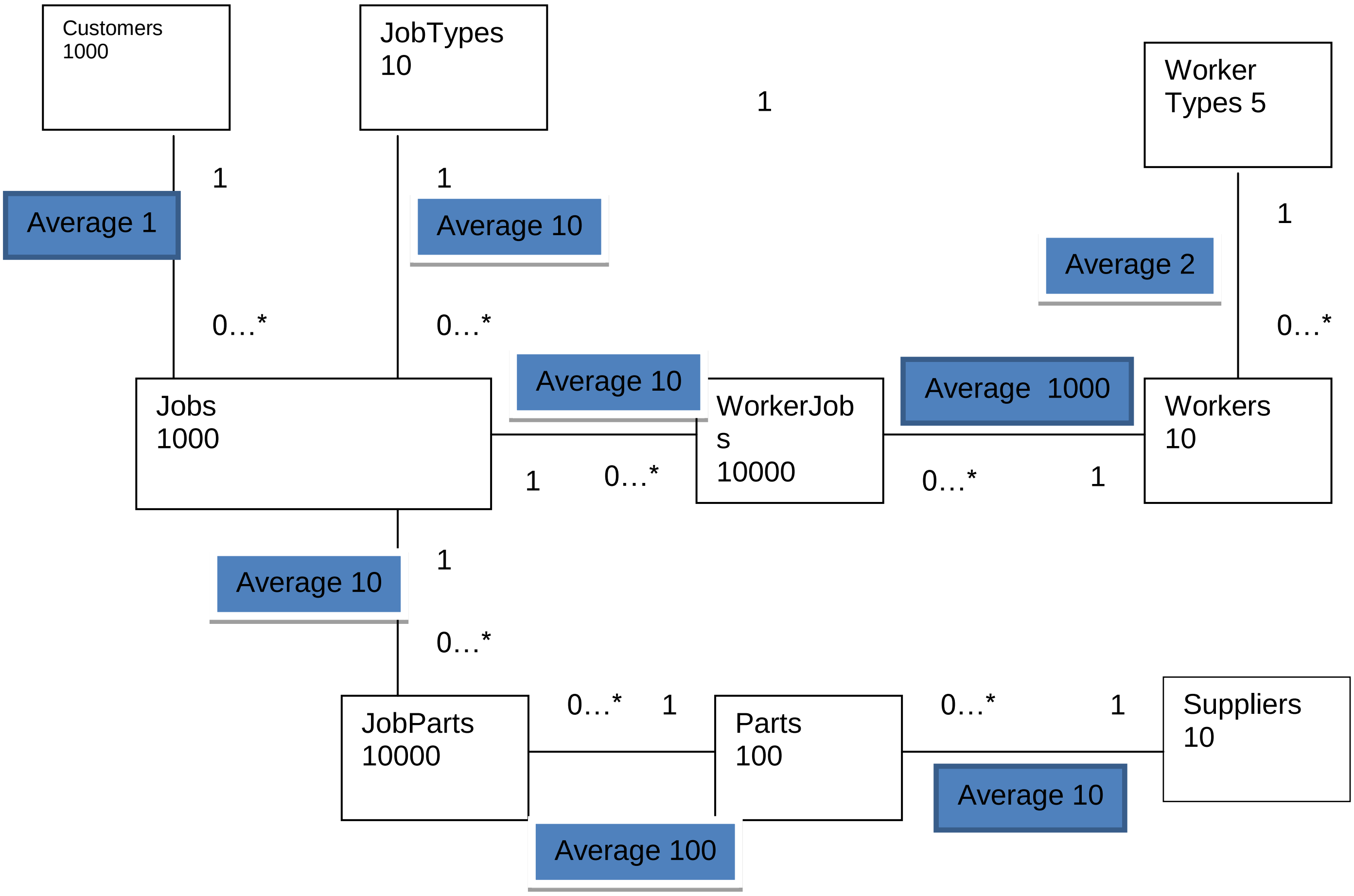
Use a copy of the table diagram and, with valid assumptions, estimate the size of the Marlowe Interiors database.

Be prepared to discuss this in the tutorial session.

Suggested Answer:

There is no particular indication of sizing in the scenario. Students should be allowed to make valid assumptions based on the size of a small to medium business.

Initial assumptions should be of size of particular tables. From this, ratios can be calculated. For example if are there typically 5 different worker types and 10 workers then the ratio is 2. Only averages are shown here.



9.7 Tutorial Notes

The time allowance for tutorials in this topic is 2 hours.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

You will need to provide students with some blank transaction analysis forms either on paper or electronic format for use in Exercise 4. One blank template is also provided in the Student Guide.

Exercise 1:

In small groups, discuss your findings from Private Study Exercise 1 and develop a group presentation that gives an overview of the topics mentioned in the exercise.

Each group should present their ideas and the information should be discussed as a whole class group.

Suggested Answer:

See Private Study Exercises.

Exercise 2:

In small groups, discuss your estimates of database sizing. Check the relationships to ensure consistency between entities. For example where the average ratio is used this should act as a multiplier of rows between relevant entities as illustrated in the slide show example.

Suggested Answer:

See Private Study Exercises.

Exercise 3:

a. What is meant by the term 'performance' with regard to database systems?
b. What is an index and what types of index are available?
c. What are the advantages of using indexes?
d. What are the potential problems with using indexes?

Suggested Answer:

a. The term 'performance' refers to the efficiency with which a query is performed against the database. It is usually measured in the amount of time it takes for a query to return the result of the query.
b. An index is a file that keeps a list of particular fields in a table that is order in some way so as to make finding a particular row or rows easier. It can be thought of as being, in concept, much like an index in a book. The different types of indexes available are primary index, secondary index and clustering index. A primary index is built around a key field with a

unique value for every entry. A secondary index is used for specifying additional key columns. A clustering index is built around a field which can have many corresponding rows in the table.

c. The advantage of using an index is that it can potentially improve performance where queries would otherwise have to search through large tables to find the rows that the query requires.
d. Disadvantages of indexes can be the overheads that sometimes accompany their use. Adding data or updating data on a table that is indexed will usually take longer as changes will also need to be made to the index file itself. More disk space is also needed to store the index file.

Exercise 4:

Your tutor will supply you with some blank transaction analysis forms either on paper or electronic format.

All transactions previously identified as part of the Marlowe Interiors system should be fully documented now using transaction analysis forms for each.

Suggested Answer:

The students should use the blank transaction form below. This can be supplied to them either electronically or as a photocopy. Transactions should have been outlined earlier and should now be specified in similar fashion to the examples given in the slides. Students should be encouraged to discuss this with their peers and tutor as a way of clarifying the nature of the transactions that they are documenting.

Transaction Analysis Form **Date:** **Transaction Reference and Name:** **Transaction volume** **Average:** **Peak:**
<Insert query to be used here>
Access Entity Type of Access Average Number Peak Number

Topic 10: Distributed Databases

10.1 Learning Objectives

This topic provides an overview of distributed databases. On completion of the topic, students will be able to:

- Recognise the need for distributed data;
- Define the main features of a distributed database;
- Define the different types of distributed databases.

10.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practise the skills during the private study, laboratory and tutorial sessions.

10.3 Timings

Lectures:	2 hours
Laboratory Sessions:	1 hour
Tutorials:	2 hours
Private Study:	7.5 hours

10.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- The need for distributed databases
- Components of distributed databases
- Advantages and disadvantages of distributed databases
- Homogenous and Heterogeneous distribution
- Distributed Database Design

10.4.1 Guidance on the Use of the Slides

Slides 2–3: Ask the students what they understand by the term distribution. ***Answer:*** The concept of being 'spread around' or 'in more than one place' should be the important thing to garner here.

Slide 4: This shows an example of a distributed database. Note that there is no indication here as to how the database is internally structured. What is depicted is various sites presumably within the same organisation. Sites 1,2 and 4 each have a database and workstations. Site 3 has no database but workstations. The various sites are connected over some form of network.

The distributed database approach assumes that there is access to the same database from each of these sites over the network.

Slide 5: A simple definition of a distributed database might be 'a database system that is split over more than one site'. Two key concepts within this field are:

- *Fragmentation:* The splitting of data in some way so that not all parts of the database exist on all sites. There are various ways of achieving this fragmentation. So, in the example in Slide 4 it might be the case that the database is split such that Site 1 has all the data relevant to Site 1, Site 2 to Site 2 etc.

- *Replication:* Reproducing data on more than one site. So in the example in Slide 4, if replication was used, then sites 1,2 and 4 would each have copies of the whole database. This obviously leads to compromising the relational model and opens up the possibility of compromising the integrity of the database.

Activity: Ask the students why this should be the case?

Answer: If whatever processes are used to keep the different copies of the database in line with each other were to fail then a copy of the database on one site might be different from a copy on another. Site 3 (which has no database) has to get its data from one of the other sites: if one version is incorrect then there is the possibility that Site 3 would get incorrect data.

Slide 6: This slide provides a summary of why distributed databases are needed. Organisations of any success tend to get bigger. It often happens that they will develop new sites both within the boundaries of one country and sometimes to other countries. This trend has been enhanced in the last century and even more so in recent decades by the use of technology that makes it easier to operate at a distance (from the telephone to the World Wide Web). With the increasing importance of database to any organisation they the need for distributed databases has grown alongside the trend in organisations being distributed.

Distribution of firms can operate in different ways. There is what might be called a ‘franchise’ model whereby the operations of the organisation are entirely replicated wherever a new centre of that organisation exists. Then there is what might be called the ‘division of labour’ model whereby different sites of an organisation perform different functions, manufacturing on one site, sales and marketing on another. When organisations go into areas of new business, i.e. diversification, then they often operate that new business from a new site; this might involve the acquisition of another business though a takeover. In this case they could inherit the firm’s database, which might be in a completely different format to their own. This raises issues for distributed database as discussed below.

When organisations are distributed and the need to access data across the different sites has been recognised then the issue of performance has to be confronted. The research and development that has gone into investigating distributed databases will aid an organisation in their search for an optimum solution.

Slide 7: This slide shows an example of fragmentation. Here a Customers table is split so that each branch of an organisation only sees customers that are relevant to them.

Slide 8: This slide looks at replication. Here the entire database is replicated at each of the sites. There must be underlying procedures that mean that at some point the databases from the different sites are kept in sync with each other.

Slide 9: With both replication and fragmentation the concept of transparency must be enforced. This means that as far as the users are concerned they are dealing with ONE database at ONE location. They do not need to know the physical location of the database or that they are dealing with a local copy.

Slide 10: Transparency enforces the foundation rule of distributed databases. From this it follows that each site should have local autonomy and that there is no reliance on a central site. This is the case whether the database is replicated or fragmented. In a fragmented case then the sub-set of records that are stored locally must be capable of performing all necessary transactions without relying on a central database. In the case of replication then the copy of the database should behave in all ways like the original.

Slides 11-12: There are different types of fragmentation available. Vertical fragmentation is where only those columns which are needed are selected (the term Project is the term from relational algebra). So in the case shown here the CustomerID, Name and Sex from the customer table are selected.

Slides 13–14: Horizontal fragmentation is covered on this slide. Here only the rows which are needed are selected (with a Restrict operation as defined in relational algebra). So in the example shown here, the Customers that are located in Manchester are shown.

Slides 15–16: Vertical and Horizontal fragmentation: This means selected columns and selected rows are chosen. So in the example given here the Customer ID, Name and Payment type for Customers in Manchester are shown.

Slide 17: Advantages of distributed databases:

- *Emulating organisational structure:* As discussed in the reasons for distributed databases this means where different functionality is spread across different sites then the data can be distributed accordingly.

- *Greater control*: Data is located where it is needed and so it is easier to control. For example if a site has control of their own customers rather than them being on a central database on different site.

- *Improved availability:.* Locally stored data is less likely to have access disrupted by network problems or issues with the central database.

- *Greater reliability:* Locally stored data is more likely to be in the control of those who have knowledge of it and so can keep it in a state that is reliable.

- *Better performance:* Although there is a trade off if data must be retrieved from across a network, where most data is retrieved from a local version (either replicated or fragmented) this could result in an increase in performance as transactions need not traverse the network. If the local database is significantly smaller than a central one then this could also increase performance as it removes the need to sort through lots of rows that are not relevant to the local site.

- *Easier Growth:* Where local sites have control over data relevant to them it is easier for them to plan and structure the growth of the database.

Slide 18: Disadvantages of distributed databases:

- *Complexity:* The computer architecture will be more complex. The structure of the data will be more complex.

- *Cost:* This will increase in terms of both hardware, systems software, application software as well as copies of the DBMS. There might also be costs in utilising networks.

- *Security:* Data travelling over possibly insecure networks. Local copies of the data might exist in situations where the security is not as robust as the central site.

- *Integrity control more difficult:* To keep replicated copies of the data in line with the central copy requires programs to make sure that this happens. Where the database is fragmented then composing tables into wholes where changes have occurred also requires checks to be in place to make sure there are no conflicts of data.

- *Lack of standards:* Standards of data that control the input of new data might not be in line between sites. This could compromise the integrity of the data.

- *Lack of experience:* It is not always easy to recruit database professionals with experience in the construction and maintenance of distributed databases.

- *Database design more complex:* As well as the actual database and architecture being more complex, design becomes more complex. The decision to distribute a database might be taken at various points in the process of development and this will impact on the design.

Slide 19. List of the types of distributed database.

Slide 20: *Homogeneous:* All sites have the same structure in the sense that they all have databases and workstations. They are also running the same DBMS and the same operating system. Issues of integration are kept to a minimum.

Homogeneous distribution is usually the result of a centrally planned approach by an organisation that has planned such a set up from the start.

Slide 21: *Heterogeneous:* This type of distributed database can be the result of organisations either allowing separate sites to develop without an overall central plan based on a single DBMS and network; or where organisations have merged or taken over other organisations where pre-existing databases and operating systems are in place.

Each site has a database and operating system that might be different from the other.

The result will be the need for more complicated software to allow the databases to seem as though they are a single database in accordance with the foundation rule.

Slide 22: A federated database is not strictly a distributed database. Rather it is a series of different databases that can be joined in various databases as the need arises.

The federated structure becomes increasingly possible to achieve as languages such as XML become the standard for encoding data from different sources.

Slide 23. Any DBMS that needs to operate for a distributed database will be more complex than for a stand-alone database.

- *More complex system catalogue*: To keep track of potentially different structures in different sites and to record location of specific structures.

- *Concurrency control:* Needs to be more sophisticated for example to make sure an update that affects more than one site operates in such a way to keep the database consistent and maintain integrity.

- *Query optimiser:* Tracing paths through tables for queries becomes more complex when the tables way be fragmented across different sites.

Finally the DBMS itself must be distributed which means that tuning and managing it becomes a more complex set of tasks.

Slide 24: Designing a distributed database necessitates taking into account a number of additional factors as listed here. As mentioned earlier, the decision to distribute might not happen at the outset of the design which might entail re-visiting logical and physical design.

Slide 35: This slide gives the learning outcomes for this topic; ask students whether they have been met.

Slide 36: This slide gives a list of references which students may find useful.

10.5 Laboratory Sessions

The laboratory time allocation for this topic is 1 hour.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

1. The syntax for creating indexes:

```
create index cust_id_index on customers(customer_id)
```

This creates an index called cust_id on the customers_id column on the customers table.

To create a unique index on 2 columns:

```
Create unique index cust_name_idx ON
customers(customer_first_name,customer_last_name);
```

In this case it might not be a good idea to do this because it is a very real possibility that Marlowe Interiors could have customers with the same name.

So drop the index:

```
drop index cust_name_idx on customers
and recreate as a non-unique index
create index cust_name_idx ON
customers(customer_first_name,customer_last_name);
```

2. Now create appropriate indexes for Marlowe Interiors.

Views are created by nesting an SQL statement inside the create view statement as follows:

```
create view london_customers as
(select * from customers
where address_line2 = 'London');
```

This creates a view called 'london_customers' which will contain all customers who have London in address_line2.

Views could implement some of the derived data from earlier.

3. Create a view based on the query you developed earlier which showed the total cost of parts for each job.

4. Create a view based on the query used to calculate labour.

Suggested Answer:

3.

```
create view Parts_total as
(select job_parts.job_id, job_parts.part_id, job_parts.quantity, job_parts.quantity * parts.price
parts_cost
from job_parts, parts
where job_parts.part_id = parts.part_id )
```

4.

You should add the job_id and perhaps think of a way of deducting the weekends.

```
create view labour_costs as
(select worker_jobs.worker_id, worker_jobs.job_id, workers.worker_type_code,
worker_types.hourly_rate, worker_jobs.start_date, worker_jobs.end_date,
datediff(worker_jobs.end_date, worker_jobs.start_date) days_on_job,
datediff(worker_jobs.end_date, worker_jobs.start_date)
* 12 * hourly_rate * 0.71 labour_cost
from worker_jobs, worker_types, workers
where worker_jobs.worker_id = workers.worker_id
and workers.worker_type_code = worker_types.worker_type_code);
```

Note the multiplication by 0.71 is a rather crude way of simulating the weekends as non worked time.

10.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Read all the relevant material for this topic. This should including the slides and slide notes along with the sections from at least one relevant textbook.

You should prepare a PowerPoint presentation or discussion document that gives a **detailed** review of the following topics:

- Definition of distributed databases
- Why distributed databases are needed
- Fragmentation
- Replication
- The Foundation Rule for Distributed Databases

Be prepared to discuss this in the tutorial session.

Suggested Answer:

The students should cover the key points of each of these topics as outlined in the slides.

A simple definition of a distributed database might be 'a database system that is split over more than one site.'

Key areas to cover are:

- *Why distributed databases are needed*: Organisations of any success tend to get bigger. It often happens that they will develop new sites both within the boundaries of one country and sometimes to other countries. This trend has been enhanced in the last century and even more so in recent decades by the use of technology that makes it easier to operate at a distance (from the telephone to the World Wide Web). With the increasing importance of databases to any organisation, the need for distributed databases has grown alongside the trend in organisations being distributed.

- *Fragmentation:* The splitting of data in some way so that not all parts of the database exist on all sites. There are various ways of achieving this fragmentation.

- *Replication:* Reproducing data on more than one site.

- *Foundation rule:* A distributed database should look, from the point of view of the user, the same as a centralised database

Exercise 2

One part of expanding the use of databases in an organisation concerns the concept of scalability. This concerns the ability of a database to operate in the same way at different sizes. Size here refers to the number of rows in a database and the number of users.

Before any move to distributing data then questions of scalability should be dealt with.

Write an email to the head of Marlowe Interiors discussing the following issues of database scalability:

- The components of a large database
- Increasing database storage capacity
- Using memory to make the database faster
- Scaling up to more powerful CPU and servers
- Evolution of the commodity server
- Scaling out across multiple database servers
- High-speed connections between database components
- Database transaction processing
- Optimizing database design and configuration

You may find the following resources useful:

- This taxonomy is adapted from “Database Scalability” by Base One International Corporation (Available Online from www.boic.com/scalability.htm)

- Oracle: Karam, S. (undated) *Introduction to Oracle Scalability Issues.* [Available Online] http://www.dba-oracle.com/t_scalability_features.htm

- SQL Server: Kerwin, D. (undated) *Achieving massive scalability with SQL Server.* [Available Online] http://www.sql-server-performance.com/2003/massive-scalability/

Suggested Answer:

The website references here could be a useful starting point to exploring this topic. Some textbooks cover scalability or aspects of it as indicated by the topic headings. The students are asked to give an overview rather than a comprehensive account. They can use these headings as a guide and alternative approaches are acceptable. The key issues to explore are how databases can cope with increased size and increased traffic in terms of transactions when they are very large and/or distributed. The topics listed above discuss approaches to this. There might said to be a simple division between the two approaches detailed above:

- Scaling up: which is relying on a single high-performance server.

- Scaling out: which is distributing the load to different servers.

10.7 Tutorial Notes

The time allowance for tutorials in this topic is 2 hours.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

Exercise 1:

In small groups, discuss your findings from Private Study Exercise 1 and develop a group presentation that gives an overview of the topics mentioned in the exercise.

Each group should present their ideas and the information should be discussed as a whole class group.

Suggested Answer:

See Private Study Exercises.

Exercise 2:

a. What are vertical fragmentation and horizontal fragmentation? Give an example of each.
b. Outline the advantages of distributed databases for an organisation.
c. What are the potential disadvantages of distributed databases?
d. Outline the main features of both homogeneous and heterogeneous distributed databases.
e. What is a federated database?
f. What features should a distributed DBMS have?
g. How does designing a distributed database system affect the physical design process?

Suggested Answer:

These answers are substantially the ground covered in the slides and slide commentary.

a. Vertical fragmentation is where only those columns which are needed are selected (the term Project is the term from relational algebra). So in the case shown in the slides from a Customers database the CustomerID, Name and Sex from the customer table are selected.

Horizontal fragmentation: Here only the rows which are needed are selected (with a Restrict operation as defined in relational algebra.) So in the example shown in the slides from a Customers database the customers that are located in Manchester are shown.

b. Advantages of Distributed Databases:

- **Emulating organisational structure.** This means where different functionality is spread across different sites then the data can be distributed accordingly.
- **Greater control**. Data is located where it is needed and so it is easier to control. For example if a site has control of their own customers rather than them being on a central database on different site.

- **Improved availability**. Locally stored data is less likely to have access disrupted by network problems or issues with the central database.
- **Greater reliability**. Locally stored data is more likely to be in the control of those who have knowledge of it and so can keep it in a state that is reliable.
- **Better performance**. Although there is a trade off if data must be retrieved from across a network, where most data is retrieved from a local version (either replicated or fragmented) this could result in an increase in performance as transactions need not traverse the network. If the local database is significantly smaller than a central one then this could also increase performance as it removes the need to sort through lots of rows that are not relevant to the local site.
- **Easier Growth**. Where local sites have control over data relevant to them it is easier for them to plan and structure the growth of the database.

c. Disadvantages of Distributed Databases:

- **Complexity.** The computer architecture will be more complex. The structure of the data will be more complex.
- **Cost.** This will increase in terms of hardware, systems software, application software as well as copies of the DBMS. There might also be costs in utilising networks.
- **Security.** Data travelling over possibly insecure networks. Local copies of the data might exist in situations where the security is not as robust as the central site.
- **Integrity control more difficult**. To keep replicated copies of the data in line with the central copy requires programs to make sure that this happens. Where the database is fragmented then composing tables into wholes where changes have occurred also requires checks to be in place to make sure there are no conflicts of data.
- **Lack of standards**. Standards of data that control the input of new data might not be in line between sites. This could compromise the integrity of the data.
- **Lack of experience**. It is not always easy to recruit database professionals with experience in the construction and maintenance of distributed databases.
- **Database design more complex**. As well as the actual database and architecture being more complex design becomes more complex. The decision to distribute a database might be taken at various points in the process of development and this will impact on the design.

d. Homogeneous and Heterogeneous distributed databases:

- **Homogeneous.** All sites have the same structure in the sense that they all have databases and workstations. They are also running the same DBMS and the same operating system. Issues of integration are kept to a minimum. Homogeneous distribution is usually the result of a centrally planned approach by an organisation that has planned such a set up from the start.
- **Heterogeneous.** This type of distributed database can be the result of organisations either allowing separate sites to develop without an overall central plan based on a single DBMS and network; or where organisations have merged or taken over other organisations where pre-existing databases and operating systems are in place. Each site has a database and operating system that might be different from the other. The result will be the need for more complicated software to allow the databases to seem as though they are a single database in accordance with the foundation rule.

e. Federated databases

A federated database is not strictly a distributed database. Rather it is a series of different databases that can be joined in various databases as the need arises.

The federated structure becomes increasingly possible to achieve as languages such as XML become the standard for encoding data from different sources.

f. DBMS for Distributed databases

Any DBMS that needs to operate for a distributed database will be more complex than for a stand-alone database.

- **More complex system catalogue**. To keep track of potentially different structures in different sites and to record location of specific structures.
- **Concurrency control**. Needs to be more sophisticated for example to make sure an update that affects more than one site operates in such a way to keep the database consistent and maintain integrity.
- **Query optimiser**. Tracing paths through tables for queries becomes more complex when the tables way be fragmented across different sites.
- Finally the DBMS itself must be distributed which means that tuning and managing it becomes a more complex set of tasks.

g. Designing a distributed database necessitates taking into account a number of additional factors. The normal process of design is followed in designing base tables. There needs to be an investigation as to which data is used at which sites and how it is used. Need to decide on the type of fragmentation needed. Consider what topology of network is required to minimise traffic on the network. The decision to distribute might not happen at the outset of the design which might entail re-visiting logical and physical design.

Exercise 3:

Marlowe Interiors want to open 2 new branches, one in Birmingham and one in Manchester. Each branch will have its own set of clients and its own employees. It will, however, share use of the same information about suppliers.

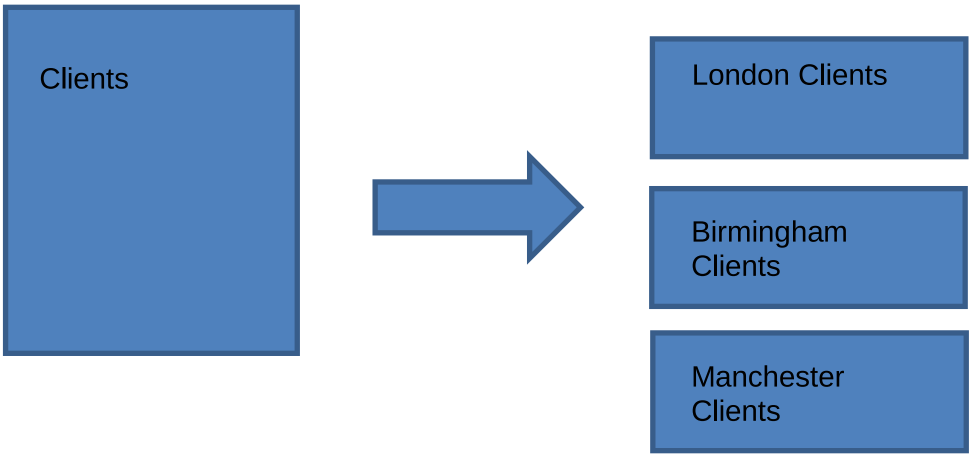
With the aid of a diagram outline how a distributed database could be implemented for Marlowe Interiors.

Suggested Answer:

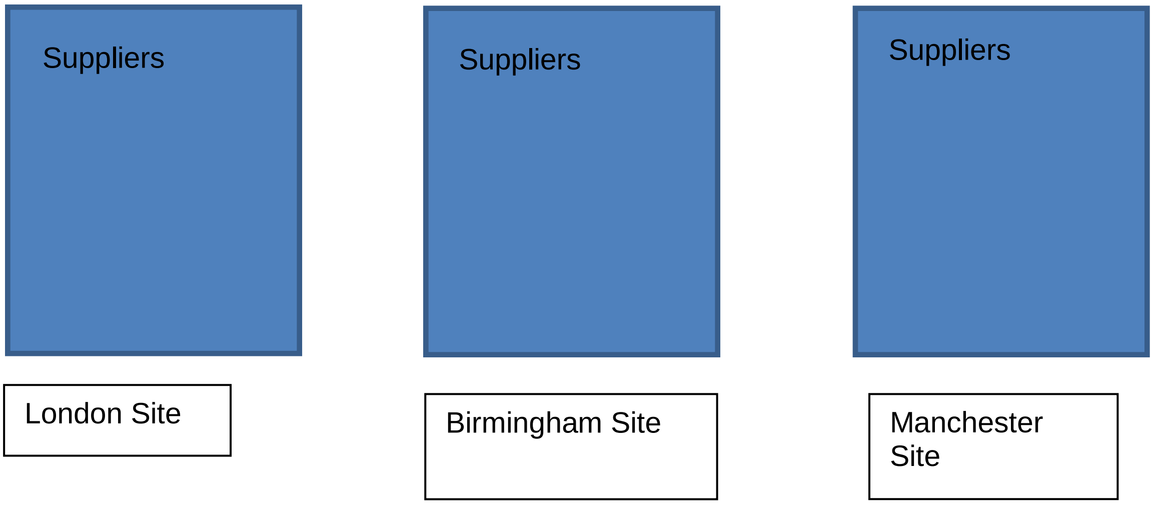
The students should recognise that the database needs to be fragmented and some data replicated.

Clients and Employees will be fragmented horizontally. Suppliers information, because it is shared, can be replicated.

Clients and Employees (only clients shown) is represented schematically (as in the slides) as:



With supplier, because they are shared then the entire table will be replicated:



Topic 11: Data Warehouses

11.1 Learning Objectives

This topic provides an overview of data warehouses. On completion of the topic, students will be able to:

- Understand the potential need for a data warehouse;
- Differentiate between online transaction processing systems and data-warehouse system;
- Identify the main components of a data warehouse.

11.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practise the skills during the private study, laboratory and tutorial sessions.

11.3 Timings

Lectures:	2 hours
Laboratory Sessions:	1 hour
Tutorials:	2 hours
Private Study:	7.5 hours

11.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- The need for business intelligence and the concept of the data warehouse
- The difference between Online Transaction Processing (OLTP) systems and data warehousing
- The architecture and main components of a data warehouse

11.4.1 Guidance on the Use of the Slides

Slides 2–3: Data warehouses are a wholly different utilisation of database technology. It is important to stress that a data warehouse is NOT just a large database but rather a way of amalgamating data from different sources into a single repository.

Slide 4: OLTP systems are what are usually thought of when speaking of a database system. These are the ‘live’ systems that are used to support everyday business activities. So examples of OLTP would be a point-of-sale system in a shop; a student record system in a college; a lending system in library; a phone-call logging system in a call system. These systems are ‘live’ in two senses: the data is relevant now to the organisation; it is also ‘live’ in the sense that when some change is made to the database it happens immediately at that point in time.

OLAP: despite a similarity in name is not a system of the same sort. OLAP is rather a way of viewing data that is stored in a format that makes it suitable to be viewed in this way; that is a multi-dimensional view of data is only possible where the data is stored in a suitable way. This is one of the cases for a data warehouse. It allows data to be viewed in a multi-dimensional way to aid management decision-making.

Slide 5: The two main reasons for a data warehouse are the need for business intelligence and as a way of integrating data from different systems.

Business intelligence is needed in a business environment that is increasingly competitive. Organisations compete with one another for shares of markets and customers. This can even be true of public sector and charitable organisations, for example colleges competing for students or different charities competing for donations. The ‘classic’ firm in economics (private sector proving goods or services) is always in competition with other firms providing the same or similar goods or services.

In order to effectively compete, organisations need to have knowledge of the market, customers, suppliers etc. This will enable them to make effective decisions.

As organisations have implemented IT solutions it has led to a proliferation of different systems. A data warehouse is a way of bringing these systems together in a way that provides an organisation with the sort of business intelligence they need.

Slide 6: Databases designed for day-to-day operations are not necessarily suitable for providing the sort of multi-dimensional data that is useful for business intelligence. There will be examples of this below. But ask the students what they think is meant by the term ‘multi-dimensional’. It means looking at data from lots of different angles: using lots of different variables. For example looking at students in a college from the point of view of what age they are, what part of the country they come from and what the economic background of their parents is. This is 3 different dimensions.

In order to allow an organisation to take this sort of multi-dimensional approach then a number of factors must be taken into account:

- *Content:* the data must be complete in the sense that all that needs to be looked at is in one place.

- *Accessibility:* it must be fairly easy to access the relevant data without the need of navigating across different record structures or systems.

- *Form:* the data needs to be in a form that is easily understood and interrogated.

- *Performance:* Storing information that is useful to management decision-making is not suitable for OLTP systems in many cases as it can affect their performance. OLTP systems are streamlined to operate day-to-day functionality and storing additional information in a suitable form for decision-making can mean duplication and compromising data integrity.

- *Availability:* OLTP systems are used to run the day-to-day business of an organisation and therefore their resources are geared towards that. This can mean that access is not always available for users to interrogate them for management information.

Slide 7: This is an example of IT use in a supermarket chain. The various sites will have point-of-sale systems. These systems will scan the barcode of items brought through the check-out and the prices retrieved based on the barcode. The sales themselves will be recorded in the OLTP for the point-of-sale system. If the customer has a loyalty card then this will be cross-referenced with the items that they have bought. The details of this will be stored in the Customer Records database.

There will be a separate system which will record sales and compare them to items in stock. When stocks on the shelves get low then new items can be ordered from the warehouse.

There might also be separate systems for online shopping.

Slide 8: ***Activity:*** Ask the students to identify the type of data that has been collected on the various systems in this example.

Answer:

Customer data – online will cross-reference sales with customer data from their registration. Point-of-sale systems will cross reference sales to customer data where customers have loyalty cards.

Sales data –online and at point of sale

Stock data – point of sale systems and online for central depots handle online sales.

Slide 9: What benefits would the organisation have from brining the data from these various systems together?

Ask the students this. Ask them to think about what sort of overall picture of the business might be gained from this.

Slide 10: This slide gives an example of some of the trends that can be examine where data is amalgamated from different systems.

For example looking at what and sold against data about the customers. Where customers have loyalty cards various pieces of information will have been gathered about them. For example their age, sex, address and any other information that may have solicited and that they have been willing to give; for example if they own a car, if they have children, if they own their own house. Such data can be cross referenced with their purchases so that their buying habits might be noted and picture of the customer base established that can used to market things like baby products, car insurance etc. Even without volunteering data information can be gleaned: so if someone starts to buy nappies (diapers) then they will be sent direct marketing information about other baby products.

Cross referencing sales data against store data could also be useful. Maybe stores sell different sorts of products depending on where they are located. This could have a bearing on what goods to stock in that store, what goods to promote etc.

Slide 11: Students should understand the distinction between data and information. Ask them if they can articulate this. Answer: information is data that has been transformed in some way so as to make it meaningful. The purpose of this is to make management decision making more intelligent. So, from the examples above, the decision about whether to market a particular product will depend on some of the trends that can be discovered if data are brought together in a way that the data warehouse does.

Slide 12: Features of a data-warehouse.

Slide 13: Data in a data-warehouse is structured in such a way that it is orientated on the categories that are meaningful to the users rather than categories that are appropriate to carrying out data operations.

While it may be true that the development of entities can produce categories that are close to the users categories this is not always the case. Where a database has been de-normalised and designed with particular applications in mind then the entities might not necessarily be a close match to real world subject categories.

Database system applications in OLTP systems are usually designed around the core operations of the system. For example in a student record system this would be enrolling student; assigning them to modules etc. The data warehouse should be organised instead around major subject areas such as student, course etc.. This orientation on subject areas affects the design and implementation of the data in the data warehouse. The major subject area of a given piece of data should form part of the unique identifier of a particular piece of data stored in a data warehouse (known as the key structure.)

The application world is concerned both with database design and application design. The data warehouse world focuses on data modelling and database design exclusively. Application design (in its classical form) is not part of the data warehouse environment since the data warehouse will NOT be used to support transactions. So, in our supermarket example, a data warehouse with data about products would NOT need to be organised in such as way as to support point of sale activity.

Data warehouse data excludes data that will not be used for helping to support management decision making, while operational application-oriented data contains data to satisfy immediate functional/processing requirements that may or may not be of use to decision-making. Although it is worth noting that it is not always easy to see in advance what data will be useful in making decisions.

Another important way in which the application-oriented operational data differs from data warehouse data is in the relationships of data. Operational data is concerned with an on-going relationship between two or more tables based on a business rule that is in effect. For example a customer having a loan. Data warehouse data spans a spectrum of time and the relationships found in the data warehouse are many; so it would concern the customer's financial history over a period of time.

Slide 14: Easily the most important aspect of the data warehouse environment is that data found within the data warehouse is integrated.

The integration shows up in many different ways - in consistent naming conventions, in consistent measurement of variables, in consistent encoding structures, in consistent physical attributes of data, and so forth.

In the example shown, the operational systems keep data in many different forms. When the process of integration occurs that will produce the data-warehouse then a single form of data representation is produced. The nature and format of the final data representation is decided upon as part of the process of data-warehouse design.

Slide 15: Data in a data warehouse is time-variant in the sense that a record is kept of the time when the data was significant. This differs from the operational OLTP system where data is relevant 'now' because it is a 'live' system as defined above.

It is important to keep track of the time with data in a data warehouse so that trends over a period of time can be traced. For example in a supermarket sales figures will be recorded along with when they occurred.

Data in a data-warehouse is therefore relevant at a period of time it is not live data in the way it is in an OLTP.

Slide 16: Data in a data-warehouse is not updated, deleted or inserted. Once the data is loaded from the OLTP systems into the data-warehouse it is keep in its new state and not allowed to be altered. This is because it is, in a sense, a historical record of data as it has been used and changed in the source systems (the OLTP/operational systems from which the data has been extracted).

Slide 17: This is a generic diagram of the data-warehouse functional model. It shows the process by which data is extracted and prepared, stored and then accessed via appropriate tools. The tools listed here are explained in the next few slides.

Slide 18: Acquisition: The data extraction and preparation involves:

- Identifying the data sources. What systems are needed? What data in those systems is relevant to the data-warehouse.

- A validation process that ensures that the data to be extracted is in an accurate state.

- Cleaning the data. This could possibly involve:

- Cleansing the data to make sure that it is accurate and that it does not contain values that are corupt or meaningless or obsolete

- Formatting and standardising the data in occurdance with the metadata rules that have been establishe to ensure integration

- Reconciling data from different sources (eitther within or across systems) that contradict each other.

- Eliminating any duplication

- Enforcing referential integrity between data

Slide 19: Storage: the schemas for this will be discussed below.

Slide 20: Access Tools that will be used:

- *Query tools*: using a standard query language like SQL or query languages specific to the commercial tool.

- *OLAP:* On-line analytical processing. Which allows multi-dimensional views of data.

- *Statistics*: Statistical analysis tools.

- *Discovery tools:* Used to uncover trends in database. What is known as ‘data mining’ whereby the tools discover trends that the users might not necessarily be aware of.

Slide 21: These steps are based on an approach taken by the Oracle corporation to building a data-warehouse. These are fairly self-explanatory and outline a process that incorporates some similar steps to that of database development in general.

Activity: One concept that is mentioned here is ‘metadata’. Ask the students if they remember what metadata is.

Answer: Metadata is data about data. It is important here because data warehouses involve the integration of data from lots of source systems which might be in different formats. Metadata is therefore needed to make sure that the eventual format of the data is standardised.

Slide 22: This diagram is again based on the Oracle example. Multiple source systems are shown (Purchasing system, Order processing system, Inventory system). They all go through the transformation/integration process moves the data into the data warehouse as defined by the metadata which makes sure the data is integrated in a standard format.

The various applications can then access the data-warehouse for querying purposes.

Slide 23: Schemas for data-warehouses often involve specialist approaches to enable fast retrieval. The general methodology of designing schemas as discussed in general database design is still relevant but since data-warehouses are built for retrieval only this has meant a number of specific solutions have been proposed.

Slide 24: Star Schema: A central ‘fact’ table is surrounded by a number of reference tables that store data relevant to the particular dimensions that will be used to query that core data. In the example shown here the central fact table contains data about sales trends. This is supported by some of the ways in which store trends might be queried e.g. by customer, store, online sales: there could be more.

Each reference table might be a de-normalised version of a number of other tables. For example the customers table here might have all the customers’ details from the OLTP customer, orders, products table.

Slide 25: Snowflake schema is variation on star schema. Here each dimension might have, in turn, its own dimensions. In this case the reference tables are not de-normalised. Thus customer is the customer table in its own right with a related dimension to do with item types.

Slide 26: Starflake schema is a hybrid of Star and Snowflake. Some reference tables will be de-normalised and some will be normalised. In this example the store table will not be de-normalised.

Slide 27: One of the main tools for querying data-warehouses in OLAP. OLAP tools have a number of features:

- *Consolidation:* allowing aggregation of data. For example Customers being aggregated into Customer Type, or Products into Product Types.
- *Drilling down:* this is the opposite of consolidation. It involves the breaking down of data into finer levels of detail. So data about Product Types could in turn be broken down into the individual products and even into sales of those products at a particular site.
- *Pivoting:* being able to analyse the same data from different viewpoints. So looking at sales trends from a time-period point of view; a product point of view; a region point of view etc.
- *Multi-dimensional data:* the way in which OLAP achieves the above is by data structures that have been visualised as having lots of dimensions.

Slide 28: Example of a multi-dimensional cube. Here there are three dimensions, year, month and region. It is possible to draw this but within a computer it possible to store many dimensions.

Slides 29-30: Codd’s rules for OLAP tools. Features that an OLAP tool should support:

- *Multi-dimensional conceptual view:* As described above.
- *Transparency:* the underlying architecture should not be visible to the user.

- *Accessibility:* the OLAP tool should be able to access data in different formats (relational, non-relational etc.)

- *Consistent reporting performance:* the user should not experience a loss of performance as the data-warehouse grows in size or the number of dimensions is increased.

- *Client-server architecture:* the OLAP tool should be able to work in a client-server environment. (One might update this rule so that the tool should also operate in a browser based web architecture environment).

- *Generic dimensionality:* every dimension in the data-warehouse should be capable of being queried to the same level of detail.

- *Dynamic sparse matrix handling:* Sparsity refers to the fact that in a data-warehouse comprised of data from many sources there will be large amounts of empty cells as data are amalgamated. The OLAP tool should be able to handle this sparsity without loss of performance.

- *Multi-user support:* the tool should be able to handle concurrent users.

- *Unrestricted cross-dimensional operations:* as well as being able to view data from different dimensions the tool should be able to perform operations such as calculations with data from different dimensions.

- *Intuitive data manipulation:* the pivoting, drill down and consolidation operations should be accessed by the user using GUI operations such as point and click. There should, in other words, be support for strong usability.

- *Flexible reporting:* rows, columns and cells should be able to be arranged in a way that meets the users' needs.

- *Unlimited dimensions and aggregations levels:* there should be no restrictions on these.

Slide 31: This slide gives the learning outcomes for this topic; ask students whether they have been met.

Slide 32: This slide gives a list of references which students may find useful.

11.5 Laboratory Sessions

The laboratory time allocation for this topic is 1 hour.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1: Queries and transactions for Marlowe Interiors

Use some of the views created earlier to complete the following.

1. Using the view you created in the previous exercise create a query that retrieves the total costs of parts, the total cost of labour and the total cost of the entire job.

2. Modify the query to include those jobs where the costs have not yet been calculated. You will need to use an inner join.

3. Now build on what you have learnt to create a full bill for a job. This should include the customer's name, the parts and labour used and the final totals.

Suggested Answer:

1.

```
select  parts_total.job_id, parts_cost, labour_cost, parts_cost * labour_cost job_total
from parts_total, labour_costs
where parts_total.job_id = labour_costs.job_id
```

2.

```
select parts_total.job_id, parts_cost, labour_cost, parts_cost * labour_cost job_total
from parts_total
left join labour_costs on (labour_costs.job_id = parts_total.job_id)
```

3.

Note: this is only a suggested solution and, as with all the queries, will be depend on the students' databases. Notice the use of aliases here. A query like this could be embedded in a report in a third party tool.

```
select c.customer_id, customer_first_name, c.address_line2, j.job_id, j.start_date, j.job_type_code,
jt.job_type_description, lc.labour_cost, pt.parts_cost, lc.labour_cost + pt.parts_cost total_cost
from customers c, jobs j, job_types jt, labour_costs lc, parts_total pt
where j.customer_id = c.customer_id
and j.job_type_code = jt.job_type_code
and j.job_id = lc.job_id
and j.job_id = pt.job_id
```

11.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

Read all the relevant material for this topic. This should including the slides and slide notes along with the sections from at least one relevant textbook.

You should prepare a **detailed** review of the following topics:

- Why organisations might need a data warehouse
- The difference between OLTP and OLAP
- Why data designed for OLTP systems might not be suitable for OLAP systems
- How bringing data together can aid an organisation in its decision making

Be prepared to discuss this in the tutorial session.

Suggested Answer:

The students should demonstrate an understanding of the essentially different nature of a data warehouse from an operational OLTP system. The topic headers given here are discussed in detail in the slides and commentary.

Some additional remarks:

- **Why organisations might need a data warehouse**

  Students should bring out that the data warehouse has come about due to both an organisational / business trend and a technological trend. Businesses are much more spread out and diverse with lots of different data being possible. This is reflected in the technological proliferation of systems and the data that makes available. Gathering management information in this context requires bringing this data together.

- **The difference between OLTP and OLAP**

  Despite similarities in the acronyms there is really very little similarity here. OLTP are the everyday systems that operate the functions of a business. OLAP is one approach to modelling data across lots of dimensions.

- **Why data designed for OLTP systems might not be suitable for OLAP systems**

  The fundamental characteristics of a data warehouse, that it is time variant, non-volatile, integrated and subject oriented. Data in an OLTP doesn't necessarily fit this criteria.

- **How bringing data together can aid an organisation in its decision making**

  Discovery of trends becomes possible when data is brought together Comparisons become possible between systems.

Exercise 2: Management Support Systems

You have been asked to prepare an email or memo detailing the different sorts of management information systems and approaches to data analysis that will be able to utilise a proposed data warehouse.

Give an overview of:

- Decision Support Systems
- Management Information Systems
- Data Mining

Be prepared to discuss this in the tutorial session.

Suggested Answer:

Each of these particular approaches has its own characteristics which the students should bring out.

Decision Support Systems – are systems that gather data and provide tools to aid management decision-making. For example they will have statistical analysis, graphic representation of data and reporting tools. They aim is to support management decision-making rather than replace it.

Management Information Systems (MIS) – although sometimes used to refer to all types of Management Support System the MIS has particular characteristics that differentiate it from, say, a decision support system. It is much more likely to produce regularly scheduled reports than allow free format interrogation of data. It is also likely to support operation as well as managerial level staff with data useful to their tasks.

Data mining - is an approach that looks for patterns in larges sets of data such as a data warehouse. The vital characteristic is that the patterns are discovered rather than just consciously sought. Because of the vast amalgamations of data in a warehouse it is possible that trends, patterns and dependencies exist that the user will not know about. The tools available to data-mining will discover these and represent them to the user.

11.7 Tutorial Notes

The time allowance for tutorials in this topic is 2 hours.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

Exercise 1:

In small groups, discuss your findings from Private Study Exercise 1 and develop a group presentation that gives an overview of the topics mentioned in the exercise.

Each group should present their ideas and the information should be discussed as a whole class group.

Suggested Answer:

See Private Study Exercises.

Exercise 2:

In small groups, discuss your findings from Private Study Exercise 2 and develop a group presentation that gives an overview of the topics mentioned in the exercise.

Each group should present their ideas and the information should be discussed as a whole class group.

Suggested Answer:

See Private Study Exercises.

Exercise 3:

a. Describe the four key features of a data warehouse: subject orientation; integration; time-variance; and non-volatility.
b. Within the data warehouse functional model what is the acquisition process and what activities take place within it?
c. Why is the definition of metadata important when building a data-warehouse?
d. Outline the steps needed when building a data warehouse.
e. Define the three main schemas used for data warehouses.
f. What are the four key features of OLAP?
g. Give an example of multi-dimensional data.

Suggested Answer:

These answers are substantially the ground covered in the slides and slide commentary.

a. Four key features of a data warehouse.

- **Subject orientation**. Data in a data-warehouse is structured in such a way that it is orientated on the categories that are meaningful to the users rather than categories that are appropriate to carrying out data operations.
- **Integration.** This is about bringing the data together and structuring it so that it is consistent. The integration shows up in many different ways - in consistent naming conventions, in consistent measurement of variables, in consistent encoding structures, in consistent physical attributes of data, and so forth.
- **Time-variance**. Data in a data warehouse is time-variant in the sense that a record is kept of the time when the data was significant. This differs from the operational OLTP system where data is relevant 'now' because it is a 'live' system as defined above. It is important to keep track of the time with data in a data warehouse so that trends over a period of time can be traced.
- **Non-volatility**. Data in a data-warehouse is not updated, deleted or inserted. Once the data is loaded from the OLTP systems into the data-warehouse it is keep in its new state and not allowed to be altered. This is because it is, in a sense, a historical record of data as it has been used and changed in the source systems (the OLTP/operational systems from which the data has been extracted.)

b. The Acquisition Process

The data extraction and preparation involves:

- **Identifying the data sources**. What systems are needed? What data in those systems is relevant to the data-warehouse.
- **A validation process** that ensures that the data to be extracted is in an accurate state.
- **Cleaning the data**

Cleaning the data. This could possibly involve:

- Cleansing the data to make sure that it is accurate and that it does not contain values that are corupt or meaningless or obsolete
- Formatting and standardising the data in occurdance with the metadata rules that have been establishe to ensure integration
- Reconciling data from different sources (eitther within or across systems) that contradict each other.
- Eliminating any duplication
- Enforcing referential integrity between data

c. Metadata is important because it helps to ensure consistency of data. Metadata is data that defines the structure of data within the database. Part of the process of building the data warehouse is to make sure that data from different sources is structured in the same way. The correct definition of metadata helps ensure this.

d. Steps in building a data warehouse. These steps cover the core activities in building a data warehouse. There will be alternative approaches to these that should be credited.

- Determine the needs of the end user.
- Identify the necessary data sources
- Analyse the data sources in depth
- Use the information to work out how the data will need to be transformed
- Create the metadata which describes the transformation and integration that will occur
- Create the physical data warehouse and populate from the various sources
- Create the end user applications

e. Main schemas of data warehouse

- **Star Schema:** A central 'fact' table is surrounded by a number of reference tables that store data relevant to the particular dimensions that will be used to query that core data. In the example shown here the central fact table contains data about sales trends. This is supported by some of the ways in which store trends might be queried e.g. by customer, store, online sales: there could be more. Each reference table might be a de-normalised version of a number of other tables. For example the customers table here might have all the customers' details from the OLTP customer, orders, products table.
- **Snowflake schema:** Variation on star schema. Here each dimension might have, in turn, its own dimensions. In this case the reference tables are not de-normalised. Thus customer is the customer table in its own right with a related dimension to do with item types.
- **Starflake schema:** A hybrid of Star and Snowflake. Some reference tables will be de-normalised and some will be normalised. In this example the store table will not be de-normalised.

f. Key features of OLAP

- **Consolidation**: allowing aggregation of data. For example Customers being aggregated into Customer Type; or Products into Product Types.
- **Drilling down:** This is the opposite of consolidation. It involves the breaking down of data into finer levels of detail. So data about Product Types could in turn be broken down into the individual products and even into sales of those products at a particular site.
- **Pivoting:** being able to analyse the same data from different viewpoints. So looking at sales trends from a time-period point of view; a product point of view; a region point of view etc.
- **Multi-dimensional data: The** way in which OLAP achieves the above is by data structures that have been visualised as having lots of dimensions.

g. Multi-dimensional data.

Any relevant example is satisfactory here so long as there are more than 2 axes. Students can use a diagram similar to that shown in the slides.

Exercise 4:

Marlowe Interiors has expanded rapidly and taken over several other building and interior design organisations. Each of these organisations has its own database built using a different vendor to that of Marlowe Interiors.

What are the challenges that would face the builder of a data warehouse for Marlowe Interiors? What benefits would Marlowe Interiors gain from building a data warehouse?

Suggested Answer:

The challenges will be the general ones discussed with regard to building a data warehouse. These will be the problems of integrating data from different systems. The nature of the acquisition process and potential issues of organising storage. The source systems are explicitly mentioned as being different vendors.

The advantages are being able to get a picture of their customer base across the various sites; being able to check on types of jobs and labour/materials used on these across different sites and region. This can be a basis for decision making about which direction to take the business in the future.

Topic 12: Module Overview

12.1 Learning Objectives

This topic provides an overview of the module as a whole. On completion of the topic, students will be able to:

- Summarise the key module topics;
- Give an outline of the knowledge needed about each topic for assessment purposes.

12.2 Pedagogic Approach

Information will be transmitted to the students during the lectures. They will then practise the skills during the private study, laboratory and tutorial sessions.

12.3 Timings

Lectures:	2 hours
Laboratory Sessions:	1 hour
Tutorials:	2 hours
Private Study:	7.5 hours

12.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Module overview

12.4.1 Guidance on the Use of the Slides

Slides 2–4: This lecture highlights some of the main topic areas that have been covered during the module as a whole. It is not meant to serve as a shortened version of the whole module in the sense that it is not only what is highlighted here that is relevant for assessment purposes. Rather it should be seen as a key into re-exploring each of the topics in detail.

Slide 5: The main features of a database should be familiar.

Slide 6: The different types of databases systems. Ask the students if they can define each in turn. The original definitions are as follows:

- *Transaction Processing System (TPS)* – A typical ‘everyday’ database system. When an application, such as a form, updates something, then it will be saved to the database straight away. For example, entering a new customer’s details onto a sales system; when ‘Save’ is pressed, the new customer is saved to the database.

- *Management Information System (MIS)* – A system that looks at data in various ways to get knowledge. Reports will be produced from either a TPS or data warehouses.

- *Data Warehouses* - Amalgamations of databases. Data is not live on these systems, because it can’t be updated; it will be drawn from TPS. These systems will be interrogated by applications to gain management information, such as discovering trends, e.g. the sorts of products that were bought in a particular geographic area.

- *Distributed Databases* - Large organisations might be spread geographically and so their databases have to cope with this. Data might be split or replicated.

It is worth noting that the main focus of the module has been on the development of transaction processing systems.

Slide 7: Students should have a clear understanding of metadata and why it is important.

Metadata is data that contains the structure of other data. The structure of the tables in a relational database is kept within the database itself in the form of metadata. This defines the name of the table, the name of the column, the length of the column and the data-type. The students could be asked at this point about their understanding of data-types, e.g. what data-types are available? It should be pointed out that different implementations of relational databases from different vendors might have slightly different names for the same data-type, even though there are standards. Metadata is important, because it is the way a database knows about its own structure and is able to realise one of the advantages of databases: program-data independence. This is because with metadata, the structure of the database, such as the tables, is stored in the database itself. A programmer only needs to know how to access this in order to be able to retrieve data from the database. The collection of metadata in a database is known as the data dictionary.

Slide 8: The students should have a clear understanding of the relational model as a whole. The key concepts outline here should, by now, be second nature.

- *Relations and Tables* - Tables can be thought of as the most basic structure in the relational model. Within the model itself, as opposed to its implementations, the equivalent of table is known as a relation. Tables are made up of attributes. Relations are implemented in the database as two-dimensional tables made up of columns and rows.

- *Attribute* - The qualities or property of an entity (already mentioned). It corresponds to a column in a table.

- *Domain* - The set of valid values of an attribute. For example, the valid values in the domain ‘sex’ would be ‘Male’ and ‘Female’.

- *Tuples and rows* - An instance of an entity is known as a tuple. It corresponds to a row in a table. Each tuple or row represents the set of values in the attributes of a particular table.

- *Primary Key* - The attribute or attributes that uniquely identify a row in a table.

- *Foreign Key* – This is an attribute which references an attribute (usually the primary key) in another table. Foreign keys are the way in which relationships are represented in the implemented database.

Slides 9–10: The phases of design that have been covered are shown. Students should be able to clearly define each of these.

- *Conceptual Database Design* - At this stage, data is investigated and design is undertaken without regard either to the physical implementation OR the data model. So the designers at this stage do not even assume they will be using the relational model. The sorts of areas of investigation are: what data does the enterprise hold? What format is it in? How is it used? How might the use of data affect how it is held? What terminology do users have for their data?

- *Logical Database Design* - Constructs the model without regard for the particular DBMS that will be used. However, the data model (e.g. the relational model) is known. Activities include normalisation, entity relationship modelling, defining attributes.

- *Physical Database Design* - The move from entities to tables is one of the key activities here involving what is known as designing the base relations. But this phase also involves other activities, such as indexing, de-normalisation, view creation and query tuning.

Slides 11–13: Properties of a relations and normalisation. Students should have a clear understanding of all aspects of normalisation up to 3rd normal form. They should be able to normalise a document like the one shown in Slide 13.

Slides 14–16: The steps of logical design. The students should been able to identify these and give a definition of each of them.

Slide 17: The concept of iteration in design. It has been stressed throughout that design involves re-visiting steps made after consultation with users. Although the way we have described the design process as a set of distinct steps this has been for practical educational purposes. Within a business environment the involvement of users and the constant refinement of design during the development process are key aspects.

Slides 18–21: The purpose of SQL should be clear. Students should have a strong grasp of both data definition and data manipulation. Through the SQL workshops they will have encountered complex queries and joins.

Slides 22-23: The steps of physical design. The students should be able to identify these and give a definition of each of them.

Slides 24–25: Derived data. Student should be able to define derived data and state the ways in which it can be derived as discussed in the lecture.

Discuss with students the differences between storing in a table column, in a view or generating at run-time. Through the SQL workshops they will also have written queries that calculate complex derived fields.

Slide 26: Define the different sorts of constraints.

- *Entity integrity*: enforcing the rule that no part of a primary key can be null.
- *Referential integrity:* any foreign key must match a candidate key in the parent table.
- *Propagation constraints:* enforcing the rules as to what happens if data that is referenced elsewhere is altered. There are various options:
  - No action means the record in the table with the foreign key is left as it is.
  - Cascade which means that any change (including a delete) is replicated in the table with the foreign key in it.
  - Set Default, which means that the change in the parent table causes the record in the child table to be set to some sort of default.
  - Set Null. Similar to Set Default except that the table with the foreign key has that foreign key set to Null.
- *Domain constraints:* enforce domains (the set of valid values for a column). There are different ways in which this can be done.
- *Table constraints:* enforce more complex business rules.

Slide 27–28: Transactions. Students should be able to understand what a transaction is and define the operations that go to make it up.

Slide 29–31: Distributed database. The full set of notes will need to be examined. Fragmentation and Replication are key topics. The students must be able to articulate the advantages and disadvantages of distributed databases.

Slide 32–36: Data Warehouses. The need for a data warehouse in the context of the proliferation of systems should be clearly understood. The schemas and structure of multi-dimensional data are key topics.

Slide 37: This slide gives the learning outcomes for this topic, ask students whether they have been met.

12.5 Laboratory Sessions

The laboratory time allocation for this topic is 1 hour.

Lecturers' Notes:

Students have copies of the laboratory exercises in the Student Guide. Answers are not provided in their guide.

The following exercises introduce some further structures in SQL. They use the Students and Courses tables created in Topic 1.

Exercise 1: SQL Logical Operators

There are 3 logical operators in SQL.

- AND this joins two or more conditions together and specifies that both must be true. You will have been using this to compose joins throughout the exercises.
- OR this joins two or more conditions and specifies that at least one of them must be true
- NOT this specifies that the condition must not be true

Example

To get the students whose surname is surname is 'Smith' or 'Singh'

```
select first_name, last_name
from students
where last_name = 'Smith' or last_name = 'Singh'
```

Exercise 2:

1. Select students whose age is 19 , 20 or 21.

2. Select the first name, last name and course name for students whose age is 21, 20 or 19 and whose course is History or Geography.

3. The NOT operator negates a condition. It can be put in front of a condition so that the opposite is true.

4. Modify the query from 2. using the NOT operator to select the first name, last name and course name for students whose age is 21, 20 or 19 and whose course is NOT History or Geography.

Suggested Answer:

1.

```
select * from students
where age = 21 or age = 20 or age = 19
```

2.

```
select * from students, courses
where students.course_id = courses.course_id
and (age = 21 or age = 20 or age = 19)
and (course_name = 'History' or course_name = 'Geography')
```

Note: Without the brackets this gives an odd, and incorrect result.

3.

```
select * from students, courses
where students.course_id = courses.course_id
and (age = 21 or age = 20 or age = 19)
and not (course_name = 'History' or course_name = 'Geography')
```

Exercise 3: UNION, UNION ALL, INTERSECT

1. Create a table to store the details of students who are studying by distance learning.

2. Insert the following values into the table:

Student ID	First Name	Last Name	Course ID	Age
8	Gary	Smith	3	33
9	Charlie	Brown	3	21
10	Ray	Patel	2	21
11	Amelia	Drongo	2	21
3	Dawn	Cove	1	21

The UNION, UNION ALL and INTERSECT operators are set operations. They only work where a query specifies the same types of rows from the tables being used. For example to select the student id, first name and last name from the students on both the normal Students table, and those on the Distance_students table then use the following:

```
select student_id, first_name, last_name
from students
UNION
select student_id, first_name, last_name
from distance_students
```

Note that the student ‘Dawn Cove’ appears only once. That is the function of the UNION operator.

3. Now run the same query as above but instead of UNION use the UNION ALL operator.

4. Select all the student from Students and from Distance_Students who are under 30 years of age.

The INTERSECT operator selects just those rows that appear in both tables.

The LIMIT operator get extreme (highest or lowest) values.

For example, to get the two oldest students from the student table:

```
select first_name, age
from students
order by age desc
limit 2;
```

5. Run this query.

6. How would you get the youngest 3 students? Run the necessary query.

Suggested Answer:

1.

```
create table distance_students
(student_id int not null primary key,
first_name varchar(30),
last_name varchar(30),
course_id int,
age int)
```

2.

Format is:

```
    insert into distance_students values (1,'Gary','Smith',3,33)
```

3.

```
select student_id, first_name, last_name
from students
UNION ALL
select student_id, first_name, last_name
from distance_students
```

4.

```
select student_id, first_name, last_name
from students
WHERE AGE < 30
UNION
select student_id, first_name, last_name
from distance_students
WHERE AGE < 30
```

5.

```
select student_id, first_name, last_name
from students
intersect
select student_id, first_name, last_name
from distance_students
```

Note: there is no INTERSECT operator in MySQL.

6.

```
select first_name, age
from students
order by age asc
limit 3;
```

12.6 Private Study

The time allocation for private study in this topic is expected to be 7.5 hours.

Lecturers' Notes:

Students have copies of the private study exercises in the Student Guide. Answers are not provided in their guide.

Exercise 1:

These questions review important material covered in the module. Provide written answers for each question and be prepared to discuss them in the tutorial session:

1. What is meta-data and why is it important in a database system?
2. What is 'requirements gathering' and what role does it play in database development?
3. What are the key properties of a relation
4. Give a definition and example of the concept of functional dependence.
5. Give a definition of the concept of a domain and explain how it is related to the concept of a data-type.
6. What is the difference between a super key and a candidate key?
7. Explain the concept of a recursive relationship.
8. What is an aggregate function in SQL and why must it be used with the Group By clause?
9. What is the purpose of an Outer Join in SQL?
10. What is the Cartesian product of two tables in SQL?
11. What is the difference in purpose between logical and physical design?
12. What is a CASE tool and how might it be used in database development?
13. Define the concept of derived data.
14. How are SQL aggregate functions used in generating derived data?
15. What is the entity integrity rule and why is there a danger of it being compromised in some vendor's implementations of the relational model?
16. What is a sequence or auto increment used for?
17. What does the term 'performance' refer to with regard to database transactions?
18. What is an index on a database and what is its purpose?

19. What are the advantages of distributed databases?

20. What does the concept of ‘scalability’ refer to?

21. What is meant by the term ‘non-volatility’ with regard to data-warehouse systems?

22. Why is meta-data an important concept in the development of data-warehouses?

Suggested Answer:

1. Metadata is data that contains the structure of other data. The structure of the tables in a relational database is kept within the database itself in the form of metadata. This defines the name of the table, the name of the column, the length of the column and the data-type. The students could be asked at this point about their understanding of data-types, e.g. what data-types are available? It should be pointed out that different implementations of relational databases from different vendors might have slightly different names for the same data-type, even though there are standards. Metadata is important, because it is the way a database knows about its own structure and is able to realise one of the advantages of databases: program-data independence. This is because with metadata, the structure of the database, such as the tables, is stored in the database itself. A programmer only needs to know how to access this in order to be able to retrieve data from the database. The collection of metadata in a database is known as the data dictionary

2. ‘Requirements Gathering’ is part of systems analysis. It involves investigation of what users want and what might be needed in any new system. This could involve various different techniques such as interviews and examining existing systems and data. There are different methodologies available, but recently iterative approaches have been popular. These entail a large amount of user involvement, for example by using prototypes that are shown to the user to enable them to better identify what they want.

3. Relation has a name that is unique within the relational schema.

   - Each cell in the relation contains a single (or atomic) value.
   - Each attribute has a name unique within the relation.
   - The values of any one attribute should be drawn from the same domain.
   - There are no duplicate tuples (rows) within a relation.
   - There is no significance to the order of attributes.
   - There is no significance to the order of tuples.

4. Within a relation if it is said that ‘A determines B’ then this means that if you know the value of ‘A’ then you will know the value of ‘B’. So A functionally determines B. Or B is functionally determined by A. Note that the reverse is not true. The diagrammatic representation of this is:

   A □ B
   A concrete example is attributes from a table of students.
   StudentID □ StudentName

   If we know the StudentID then we know the StudentName. But the reverse is not true because we might have students who have the same name.

5. A domain is a set of valid values from which an attribute draws. This is related to the concept of a data type which is the format of the column in terms of whether it is a number or character string. A domain is a more strict definition of the values of a column. For example if we wanted to record the potential values of a six-sided die. The data would be integer, the domain would be the set of integers from 1 through to 6.

6. Super Key - An attribute or set of attributes that uniquely identifies a tuple.

   Candidate Key - A candidate key should be a super key. However ALL the attributes of this super key must be necessary to uniquely identify it: i.e. there should be no superfluous attributes that are part of the key. It should not be the case that any subset of the attributes that go to make up this key should qualify as a super key; ALL the attributes are necessary

7. A recursive relationship is where an entity has a relationship to itself.

8. Aggregate functions are functions which perform calculations on a number of rows, for example the COUNT function counts the number of rows that meet a certain criteria. Aggregate functions must be used with the GROUP BY clause since they are, by their nature, going to give a result that is valid for a whole group of rows rather than for individual rows.

9. The outer join is used to rows from a table where there are unmatched values in the data set. For example, using the Students and Courses tables used in the laboratory sessions: if the Course and Students tables are joined then Students who do not yet have a course will NOT be retrieved with a normal join. An Outer Join will remedy this.

10. The Cartesian product is every row from one table along with every row from another table regardless of any link between them.

11. Logical design is performed with the data model in mind but not the target DBMS. Physical design is performed with target DBMS taken into consideration.

12. A CASE tool is a tool for Computer Aided Software Engineering. It has tools for defining data structures such as entities, attributes, tables and columns etc. It also has code generators to generate SQL Create scripts. Processes and transactions can also be defined, so allowing application development. It is useful to record and automate some key aspects of database development.

13. Derived data is defined as a column whose value is derived from the value of one or more other columns in the database. These might be columns within the same table or columns from one or more other tables.

14. These functions are used to calculate totals based on values in the database. For example a total column could be derived by using the SUM function to add up some sub-totals.

15. The rule states that if an attribute or group of attributes is specified as a Primary Key then it cannot contain a NULL value. However the relational model is a theoretical construct that is implemented in different ways by different vendors. Defining something as a PRIMARY KEY on most vendor's databases will enforce the rule that no part of it can be NULL. However, as previously noted, it is possible to create tables that do not have primary keys. If, for example in Oracle, we created a table and enforced the Primary Key with a Unique index this would not enforce the Entity integrity rule as it allows NULLS

16. It is used to generate a unique sequential number. This is usually for a primary key field.

17. Performance refers to the amount of time transactions take.

18. Indexes are a way of increasing the performance of a query. They can be thought of operating in the same way as an index in a book. If someone wants to find a particular topic in a book then one way of doing it would be to start on page one and turn each page until the desired topic is found. This is not a very efficient way to go about locating the topic unless the book has very few pages. Using a book index is usually more efficient since the index just contains the topic titles and page numbers and will be much shorter in page numbers than the whole book.

    Indexes in databases operate in much the same way. They are separate files that contain location pointers to the actual rows in the database. Instead of a topic title like in a book index they usually contain one or more attributes that belong to a row. They could, for example, contain just customer numbers from a customer table, rather than all the data for the customers as the actual database contains.

19. Emulating organisational structure. As discussed in the reasons for distributed databases this means where different functionality is spread across different sites then the data can be distributed accordingly.

    Greater control. Data is located where it is needed and so it is easier to control. For example if a site has control of their own customers rather than them being on a central database on different site.

    Improved availability. Locally stored data is less likely to have access disrupted by network problems or issues with the central database.

    Greater reliability. Locally stored data is more likely to be in the control of those who have knowledge of it and so can keep it in a state that is reliable.

    Better performance. Although there is a trade off if data must be retrieved from across a network, where most data is retrieved from a local version (either replicated or fragmented) this could result in an increase in performance as transactions need not traverse the network. If the local database is significantly smaller than a central one then this could also increase performance as it removes the need to sort through lots of rows that are not relevant to the local site.

    Easier Growth. Where local sites have control over data relevant to them it is easier for them to plan and structure

20. The concept of scalability refers to how well a given database performs at various sizes of implementation. Size here refers to numbers of data structures, rows of data and numbers of users.

21. This refers to the fact that in a data-warehouse data, once loaded, is not changed in any way. Data in a data-warehouse is not updated, deleted or inserted. Once the data is loaded from the OLTP systems into the data-warehouse it is keep in its new state and not allowed to be altered. This is because it is, in a sense, a historical record of data as it has been used and changed in the source systems (the OLTP/operational systems from which the data has been extracted).

22. It is important here because data warehouses involve the integration of data from lots of source systems which might be in different formats. Metadata is therefore needed to make sure that the eventual format of the data is standardised

Exercise 2:

Look over the module materials, noting any areas where you require further explanation. Prepare a detailed set of notes on these areas and take them to the tutorial session.

12.7 Tutorial Notes

The time allowance for tutorials in this topic is 2 hours.

Lecturers' Notes:

Students have copies of the tutorial activities in the Student Guide. Answers are not provided in their guide.

Some of the tutorial time can be spent reviewing the answers the students' prepared to the questions in Private Study Exercise 1, before attempting the exercise below. As this is the final laboratory session, you may also wish students to attempt the sample examination paper, which can be found on the NCC Education Campus (http://campus.nccedu.com).

Exercise 1:

In small groups discuss the areas of the module content that you do not fully understand. Attempt to answer each other's questions. If you are not able to answer these questions between you then make a note of the questions and raise these with your tutor.

Suggested Answer:

It is envisaged that this session is structured around questions about the module that the students have. Students should be split into groups. Students should have questions that they feel require further explanation. Within their group they should be encouraged to raise theses and to attempt to answer the questions raised by other members of the group. You should monitor the group's progress and encourage discussion.

There should then be a plenary session with a discussion of each of the issues the students have raised and how each group has dealt with them. You should offer guidance and explanation of the materials within this session.