Higher
Computing Science

Database Design and Development

Pupil Notes

Name:_____

Contents

Analysis

End User and Functional Requirements

During the analysis stage of database development, you should identify the following requirements:

1 End-user requirements:

- the end users are the people who are going to be using the database
- their requirements are the tasks they expect to be able to do using the database

2 Functional requirements:

- processes and activities that the system has to perform
- information that the system has to contain to be able to carry out its functions

These requirements will help:

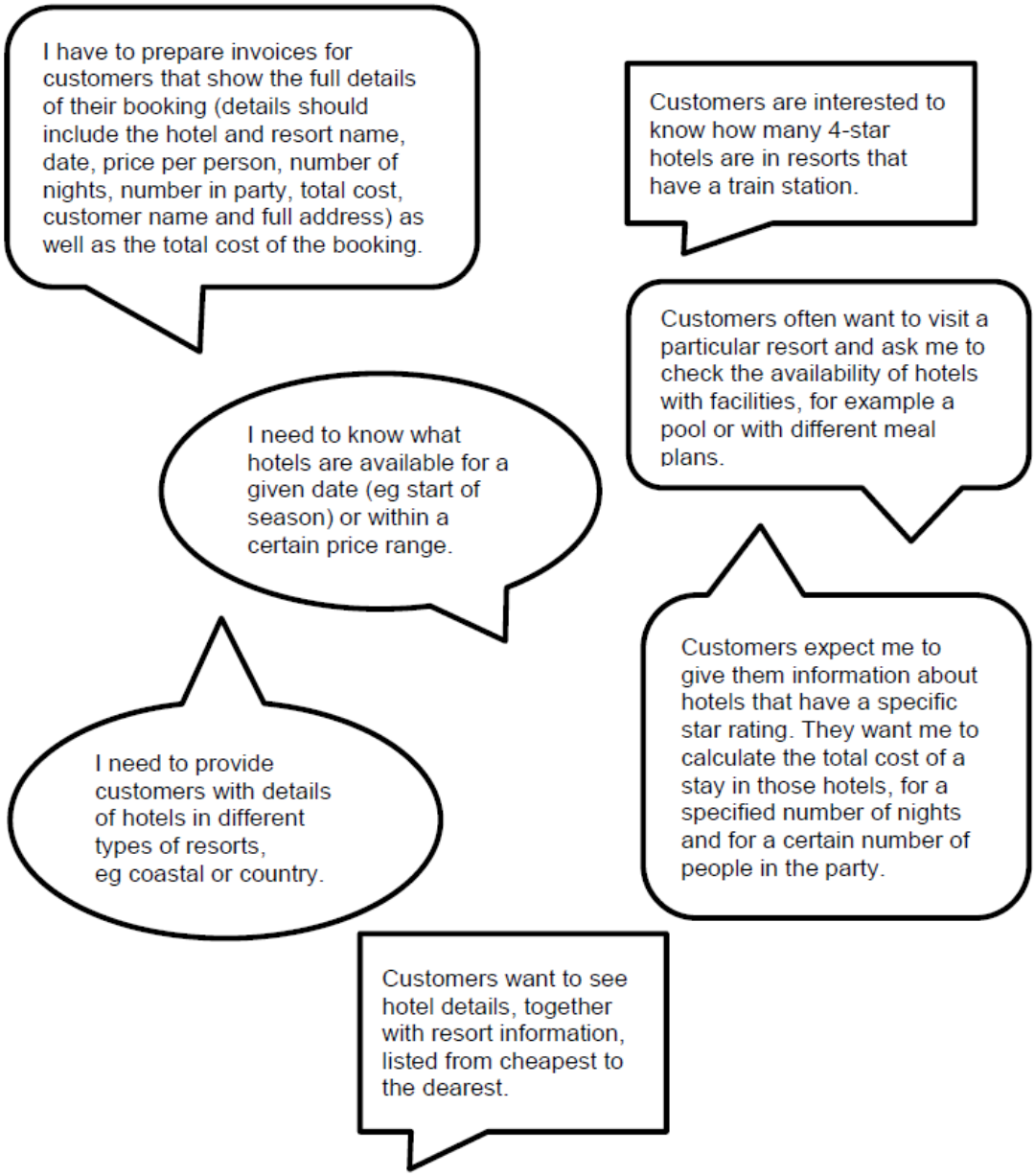
- clarify the design of the database
- identify the features to be implemented on the database
- evaluate whether the system is fit for purpose after development is complete

Worked example

A travel agency wants to create a relational database to store details of bookings for hotels in Scottish holiday resorts. The database will allow travel agents to view details of hotels and make bookings for customers. Four separate entities are required:

- Hotel (used to store details of hotels in each resort)
- Resort (used to store details of Scottish holiday resorts)
- Customer (used to store details of customers who make holiday bookings)
- Booking (used to store details of hotel bookings)

They have appointed a developer team to carry out an analysis of the database requirements. The developers ask some of the travel agency staff about the features they would expect to see in the completed database. The following are a few of the comments made by the staff:



End-user requirements

Travel agency staff should be able to perform a range of searches to display:

- full details of any booking
- availability of hotels in a particular resort, with specified facilities (meal plan or pool)
- details of hotels in a particular type of resort
- details of hotels available for a specified star rating
- resorts that have train stations

Staff should be able to sort search results in order of ascending order of price and should be able to calculate:

- the total cost of any holiday booking
- the number of hotels within a certain price range or available on a certain start date

Functional requirements

The relational database will have four tables: Hotel, Resort, Booking and Customer.
Each table requires a suitable primary key field, with foreign keys linking the four tables.
In addition to a primary key and any necessary foreign keys, the following fields are required:

Hotel:

— hotel name

— start of season date

— check-in time

— price per night

— meal plan

— swimming pool

— star rating

Resort

— resort name

— resort type

— train station

Customer

— first name

— surname

— address

— town

— postcode

Booking

— start date

— number in party

— number of nights

Use the following:

- simple and complex queries to search the database
- a simple sort to order the query results
- a calculation to work out the total cost of a booking
- an aggregate function to work out the number of 4-star hotels located in resorts that have a train station

Reading Review 1

Having read pages 5-7 answer the questions below.

Identify the end users and functional requirements for the below database.

Super Taxi allows users to book taxis from their smartphones. Super Taxi uses a relational database to keep a record of their cars, drivers, bookings and customers. Super Taxi would also like to give customers the option to book certain makes of cars and calculate the cost of their booking before travel.

Car	Driver	Booking	Customer
Registration	Driver ID	Booking ID	Customer ID
Make	First Name	From	Known As
Model	Surname	To	Card Number
Licence Expires	Mobile	Cost	Expiry Date
	Registration*	Driver ID*	Authorisation Code
		Customer ID*	

End-user Requirements:

Function Requirements:

Design

Revision

What is a database?

A **database** is used to store information. Databases can contain thousands of pieces of information, stored in a variety of formats

Databases are used as they can be searched and sorted very efficiently and they can allow a number of people to use the same information simultaneously.

Database Structure

Databases contain tables. Each **table** contains records. One **record** is all the data stored about one person or one object.

The records contain fields; a **field** is one single piece of information.

All the records in a table must have the same fields. Fields can have many different data types.

Linking Tables

Databases can contain more than one table and these tables can be linked using **primary keys** and **foreign keys**.

Types of Computerised Database

There are two types of computerised database

- Flat file Database
- Relational Database

Flat File Database

A flat file is **not** the most efficient way to store data.

Member ID	Initial	Surname	Title	Postcode	Tele	Dog Name	Gender	DofB	Breed	Origin	Life Expectancy
1	A	Fish	Mrs	CV35QW	02476111111	Bongo	M	21/08/09	Poodle	China	5
1	A	Fish	Mrs	CV35QW	02476111111	Jiccup	F	08/08/08	Poodle	China	5
1	A	Fish	Mrs	CV53QW	02476111111	Rizla	F	09/09/10	Poodle	China	5
1	A	Fish	Mrs	CV35QW	02476111111	Gov	F	11/01/11	Alsatian	Germany	10
2	C	Here	Mrs	CV27RF	01788222222						
3	D	Lapidated	Mr	CV14RR	02476333333	Manic	M	11/01/11	Poodle	China	5
3	D	Lapidated	Mr	CV14RR	02476333333	Blip	F	02/02/11	Spaniel	France	7
4	V	Ray	Ms	CV12YY	02476444444						
5	Y	Nott	Mr	CV24TT	01788555555	Ruff	M	08/08/10	Poodle	China	5
5	Y	Nott	Mr	CV24TT	01788555555	Addi	M	10/02/10	Poodle	China	5
5	Y	Nott	Mr	CV24TT	01788555555	Catnip	F	10/03/99	Poodle	China	5
5	Y	Nott	Mr	CV24TT	01788555555	Emmi	F	11/03/11	Poodle	China	5
5	Y	Nott	Mr	CV24TT	01788555555	Gov	F	11/01/11	Alsatian	Germany	10
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....

The unnecessary duplication of data values not only wastes memory, it can lead to **modification errors.**

As a result, flat file databases lead to **inconsistencies.**

Example 1

When member 5, Y Nott, moves house, the club secretary will need to change the postcode five times.

*The repeated modification of the same data item can lead to inconsistencies. This type of **modification error** occurs when repeated data is **updated**.*

Example 2

When the details of D Lapidated, are removed from the database, the details of both of his dogs will also be removed.

*If this is the only member with a spaniel, the club will lose the only record containing information about the breed. This type of **modification error** occurs when data is **deleted**.*

Example 3

As the data is organised at present, there is no way to add details of a new breed without first adding details of a member who owns a dog of that breed.

*This type of **modification error** occurs when there is an attempt to **insert** new data.*

Reading Review 2

Having read pages 10 – 11, answer the questions below.

1. Data duplication can lead to modification errors in flat-file database.
Give an example of a modification error that could occur in each of the following situations.

a) Inserting new data.

_____
_____
_____
_____

b) Updating existing data

_____
_____
_____
_____

c) Deleting existing data

_____
_____
_____
_____

2. Give another reason why data duplication is a problem for flat-file databases.

_____
_____
_____

Relational Database

A **relational database** is a database which contains more than one table. The tables are linked together by using primary and foreign keys.

Relational databases avoid the issues caused by flat file databases.

The design of a relational database is mainly concerned with three things:

-Entities

-Attributes

-Relationships

Entities and Attributes

An **entity** is a person, place, thing, event or concept of interest to the business or organisation about which data is to be stored.

For example, in a school, possible entities might be Student, Teacher, Class and Subject.

Any system can be represented as a collection of one or more 'objects', 'things' or **entities**

A specific example of an entity is called an instance or **entity occurrence**.

For example, John Smith, Mary McLeod and Omar Shaheed are all entity occurrences found in the Student entity; English, Computing and Chemistry are all entity occurrences within the Subject entity.

An entity is described by its **attributes**. Each attribute is a characteristic of the entity.

For example, **attributes** of the Student entity would include studentID, firstname, surname and dateOfBirth.

Primary Key

All entities **must** have a **primary key**.

A **primary key** is one or more attributes whose values are used to uniquely identify an individual record in the database

To distinguish primary keys from other attributes it is normal to use underlining.

Customer Number	Surname	Address	Balance Owning
5567	Jones	Cross Road	2.50
2913	Anderson	River Lane	0.00
4890	Murray	West Street	1.50
1622	Jones	Mill Lane	3.00
.....	.....	.....	.....

Foreign Key

A **foreign key** is an attribute in one entity that is the **primary key** of another entity.

Foreign keys are used to **link** entities in a relational database. Some complex entities have more than one foreign key.

Product ID	Product Name	Category	Unit Price	Supplier Code
P001	Corn flakes	Grocery	1.39	OB1
P002	Corn flakes	Grocery	1.95	KG1
P003	Sardines	Grocery	0.48	JW1
P004	Tea bags	Grocery	2.47	OB1
P005	Carrots	Fruit Veg	0.73	OB1
P006	Milk	Dairy	0.86	OB1
.....	.....	.....	.....	.....

Supplier Code	Supplier	Telephone	Email
OB1	Own Brand	01715217348	ownbrand@super,com
KG1	Kelloggs	01234567891	kelloggs@cereals.co.uk
JW1	John West	01812136453	johnwest@oceans.com
.....	.....	.....	.....

This example shows how information about products and suppliers is presented.

- The primary key of the Product entity is **Product ID**.

- The primary key of the Supplier entity is **Supplier Code**
- **Supplier Code** in the Product entity is the **foreign key** which links the Product entity to the Supplier entity

Compound Key

In some entities, no **one** attribute can be used as a primary key.

One solution is to use a combination of two or more attributes to create a unique identifier.

However, those attributes have to be foreign keys – primary keys from another entity.

This is known as a **compound key.**

Reading Review 3

Having read pages 13 – 15, answer the questions below.

1. Explain what is meant by the terms *entity* and *attribute*.

_____

_____

_____

_____

_____

2. Describe a primary key and its purpose.

_____

_____

_____

_____

_____

3. State what is meant by a compound key.

_____
_____
_____
_____
_____

4. Describe a foreign key and its purpose.

_____
_____
_____
_____
_____

Relationships

A relationship is a significant real world association between entities. It is a natural association between one or more entities. For example, Students learn Subjects and Teachers educate Students.

It describes how the two entities are related and can be thought of a connection between them.

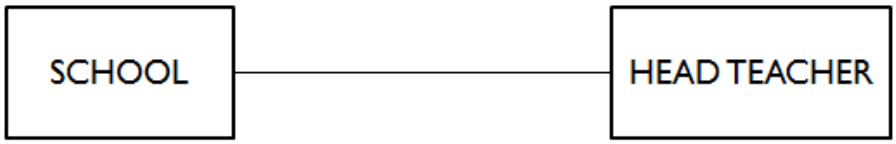
Cardinality

The cardinality of a relationship defines the number of participants in the relationship. It states the number of entity occurrences in one entity that are associated with one occurrence of the related entity. Cardinality can be:

- One-to-one
- One-to-many
- Many-to-many

One-to-one

In this type of relationship, every instance in one entity is linked with **exactly** one instance in another entity.

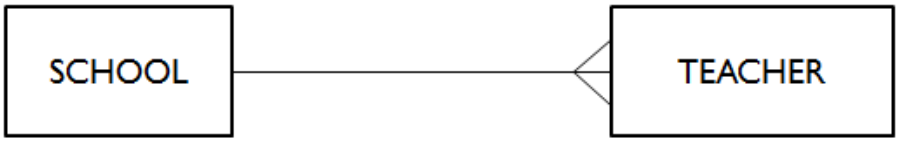


This entity relationship diagram shows a one-to-one relationship *(straight line).*

Each school has exactly one head teacher; each head teacher belongs to exactly one school.

One-to-many

In this type of relationship, every instance in one entity is linked with **several** instances in another entity.



This entity relationship diagram shows a one-to-many relationship *(crow's feet at the many end).*

Each school can have several teachers but each teacher belongs to just one school.

Many-to-many

In this type of relationship, <u>every</u> instance in one entity is linked with **several** instances in another entity.

In turn, each instance in the second entity is linked to many instances in the first entity.

This entity relationship diagram shows a many-to-many relationship *(crows feet at both ends)*

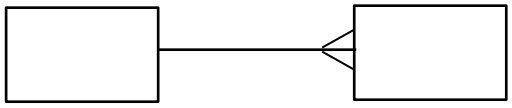
> *Each pupil is taught by many teachers and each of those teachers will teach many pupils.*

Reading Review 4

Having read pages 17 – 18, answer the questions below.

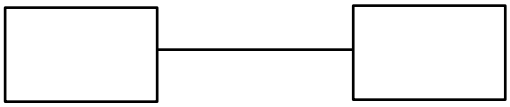
1. State the type of relationship indicated by each of the following diagrams.

a)

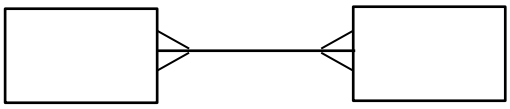


_____

b)



_____

c) 

_____

The Design Stage

includes –

- an **Entity Occurrence Diagram**
- an **Entity Relationship Diagram (ERD)**
- a **Data Dictionary (DD)**

Entity Occurrence Diagram

An entity-occurrence diagram illustrates the relationships between the entity occurrences of one entity, with the entity occurrences within a related entity. The creation of an entity-occurrence diagram helps to identify the cardinality of the relationship that exists between the two entities.

In an entity-occurrence diagram, each entity is shown as a tall oval. Inside each entity, each entity occurrence is represented by the value of its identifier and each relationship is shown by drawing a line between associated entity occurrences.

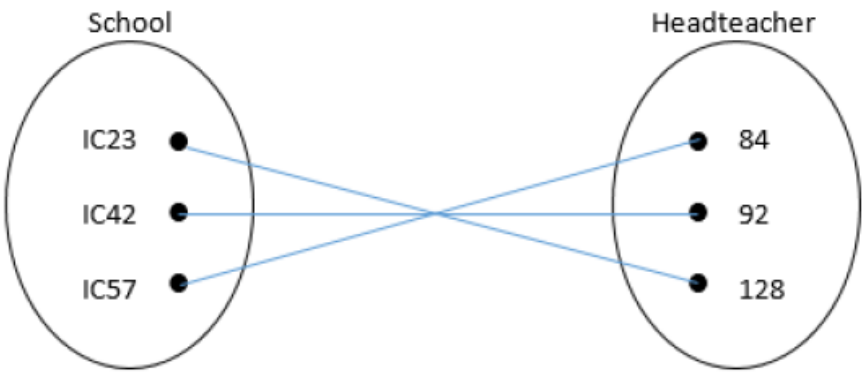
Examples of entity-occurrence diagrams are shown below.

Entity-occurrence diagram example 1: one-to-one relationship

The following table indicates which School is managed by which Head teacher and which Head teacher manages which School.

School	Headteacher
IC42	92
IC57	84
IC23	128

Here is the matching entity-occurrence diagram.



From this entity-occurrence diagram, we can see that each instance in the School entity has an association with one, and only one, entity instance in the Head teacher entity. Similarly, each instance in the Head teacher entity has an association with one, and only one, instance in the School entity.

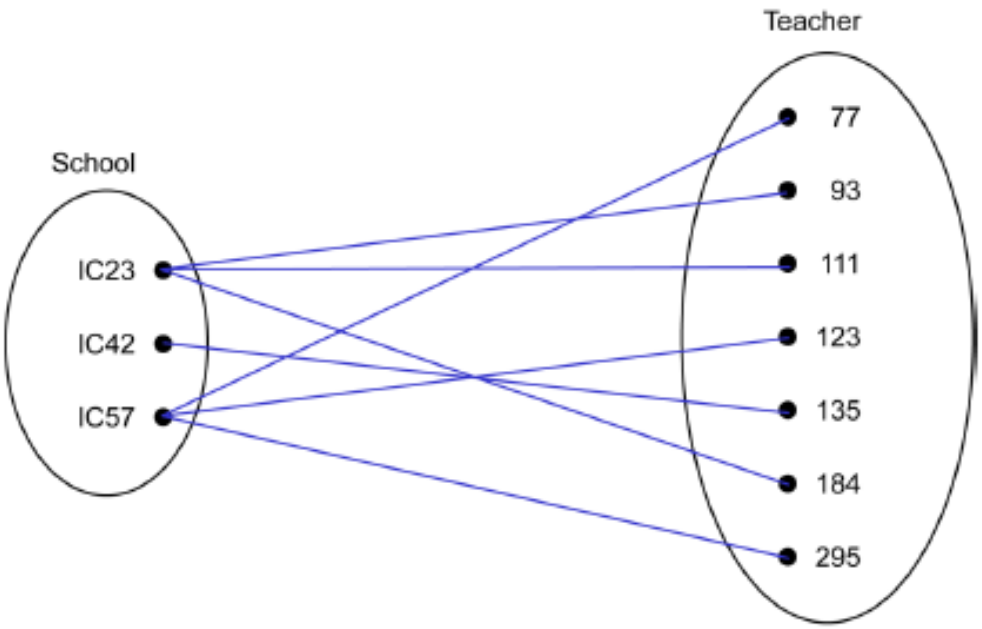
This confirms that there is a one-to-one relationship between the School and Head teacher entities.

Entity-occurrence diagram example 2: one-to-many relationship

The following table indicates which School employs which Teachers and which Teachers are employed by which School.

School	Teacher
IC42	135
IC57	123
IC23	111
IC23	184
IC57	77
IC57	295
IC23	93

Here is the matching entity-occurrence diagram.



From this entity-occurrence diagram, we can see that each instance in the School entity has an association with one or more entity instance in the Teacher entity. We can also see that each instance in the Teacher entity has an association with one, and only one, occurrence in the School entity.

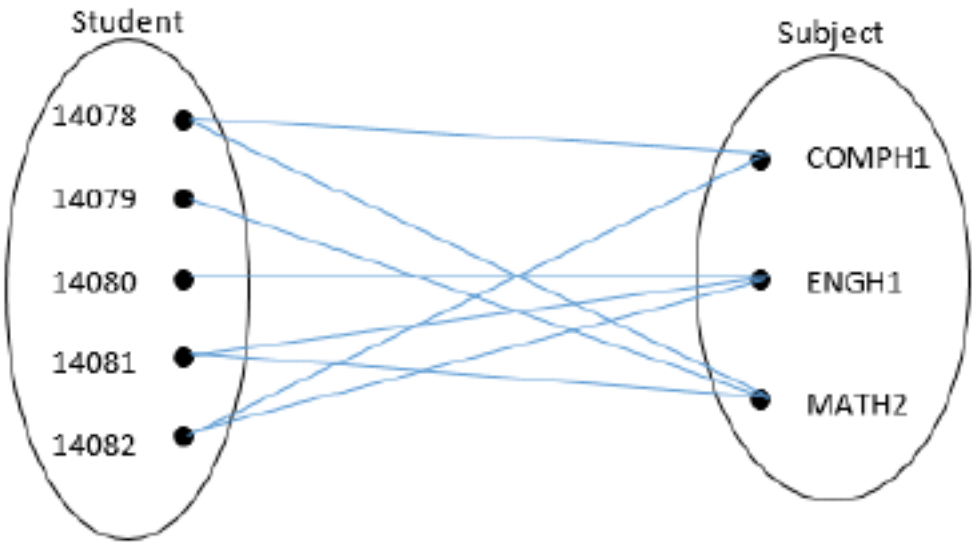
This confirms that there is a one-to-many relationship between the School and Teacher entities.

Entity-occurrence diagram example 3: many-to-many relationship

The following table indicates which Students study which Subjects and which Subjects are studied by which Students.

Student	Subject
14078	COMPH1
14079	MATH2
14080	ENGH1
14078	MATH2
14081	ENGH1
14082	ENGH1
14082	COMPH1
14081	MATH2

Here is the matching entity-occurrence diagram.

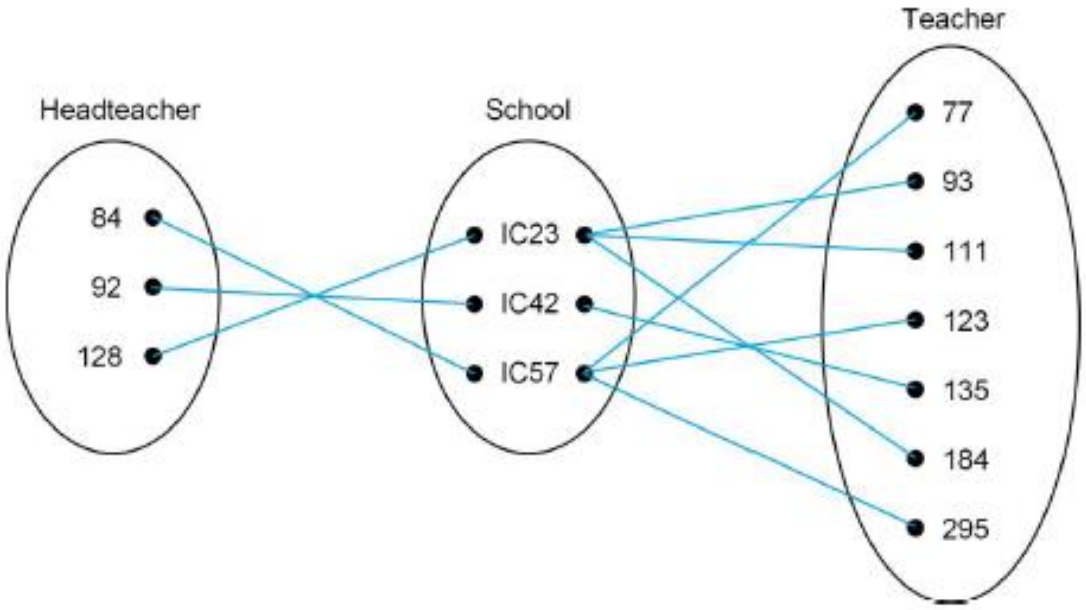


From this entity-occurrence diagram, we can see that each occurrence in the Student entity has an association with one or more entity occurrences in the Subject entity. We can also see that each occurrence in the Subject entity has an association with one or more occurrences in the Student entity.

This confirms that there is a many-to-many relationship between the Student and Subject entities.

Entity-occurrence diagram example 4: multiple entities

An entity-occurrence diagram may be used to illustrate the relationship between multiple entities as shown below:



Reading Review 5

Having read pages 20 – 24, answer the questions below.

1. Draw entity occurrence diagrams for the below entities.

   a)

Shop	Owner
SH123	Jones
SH245	Smith
SH111	Peters
SH196	Wilson

b)

Shop	Employees
SH123	ID176
SH245	ID776
SH111	ID123
SH196	ID234
SH123	ID222
SH196	ID143

Entity Relationship Diagrams (ERD)

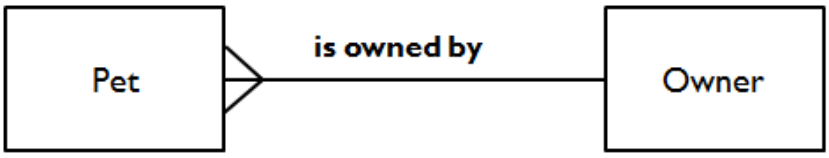
An entity-relationship diagram is a graphical representation of the entities in a system. It is used to summarise the relationship that exists between two or more entities. An entity-relationship diagram indicates:

- the name of each entity in the system
- the name of the relationship between two entities
- the cardinality of the relationship between two entities
- if required, the name of each attribute can be shown

Entities are shown as rectangles:

Relationships are shown as labelled lines:

The many end of a relationship is indicated using crow's feet.



This relationship states that each pet has one owner and each owner has many pets.

Creating ER Diagrams

When creating an entity relationship diagram, you should:

1. list each pair of entities and the relationship between them (start with the word ‘each’)
2. position the entities on the page
3. draw the relationships between the entities
4. resolve any many-to-many relationships

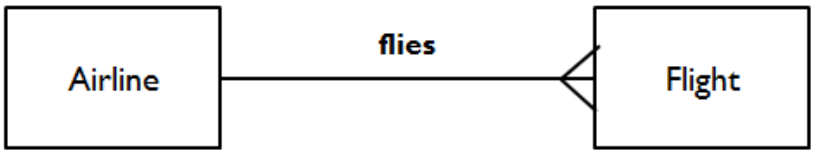
Example 1

An airline can fly many flights, but each flight is flown by only one airline.

What would the entity relationship diagram look like for this example?

The entities and relationships mentioned are:

- Each airline field many flights
- Each flight is flown by one airline



Example 2

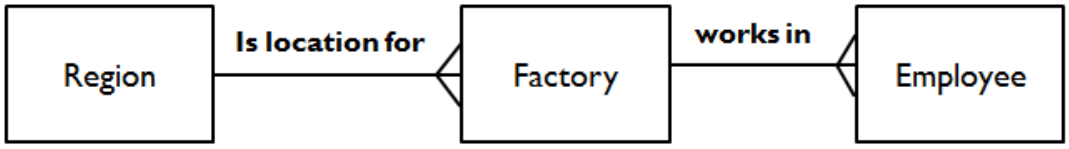
The MakeIt Corporation operates many factories and each of those factories is located in a particular region.

Each region is the location of many of the factories. Each factory employs many employees, but each of these employees is employed by only one of the factories.

What would the entity relationship diagram look like for this example?

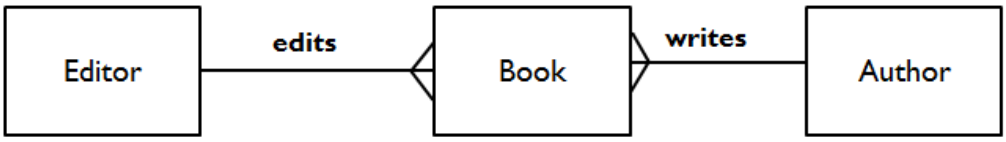
The entities and relationships mentioned are:

- Each factory is location in one region
- Each region is the location for many factories
- Each factory employs many employees
- Each employee works in one factory



Reading ER diagrams

Example 1

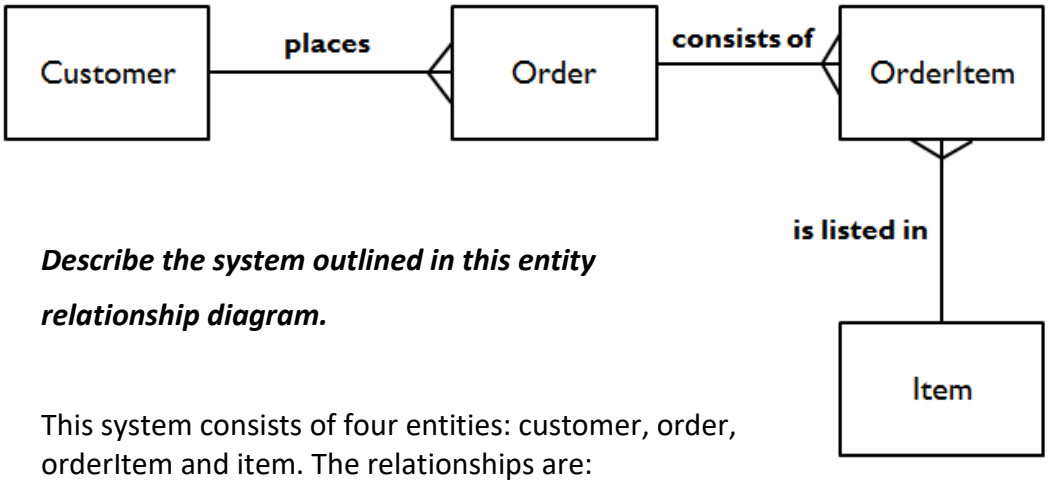


Describe the system outlined in the entity relationship diagram above

The system consists of three entities: editor, book and author. The relationships are:

- Each editor edits many books
- Each book has one editor
- Each book has one author
- Each author writes many books

Example 2



Describe the system outlined in this entity relationship diagram.

This system consists of four entities: customer, order, orderItem and item. The relationships are:

- Each customer can place many orders
- Each order is placed by one customer
- Each order consists of many order items
- Each order item is associated with one particular order
- Each order item refers to one specific item
- Each item can appear in many order items

Reading Review 6

Having read pages 27 – 30, answer the questions below.

1. The primary keys of each of the following entities are underlined. Draw an entity relationship diagram for each pair of entities.

a) Student and Tutor.

Student

Student No:	18728	Firstname:	Joseph
DOB:	06/02/96	Surname:	Millar

Course ID	Course Name	Credits	Tutor ID
D5561	Computing	1	1989
D766	Hardware	1	7817
C991	DB Design	2	7817

Tutor

Tutor ID	Tutor Name	Tutor Extension
1989	Mr Boyd	761
7817	Mrs Harris	991
6516	Mr George	123

b) Employee and Department

Employee

Employee ID	Department ID	Employee Name
AD191	Oo2	Harry Ward
ER123	001	George Clooney
GH818	003	Dawn Smith
AD154	002	Stephanie Swift
Ty176	001	Lorrenta Went

Department

Department ID	Department
001	Sales
002	HR
003	Manufacturing

c) Driver and Taxi

Driver

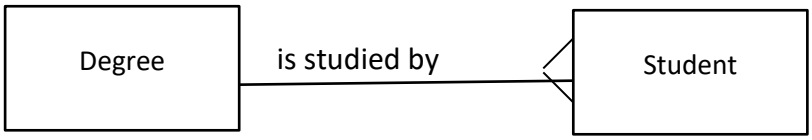
Driver ID	Driver Name	Taxi
03	Jo Fleming	TY528UQ
15	Ian Smith	AS512ER
18	Pat Page	VJ537KL

Taxi

Registration Number	Carries	Disabled Access
Ty528UQ	5	Yes
AS512ER	4	No
VJ537KL	8	Yes

2. Complete the description of the following one-to-many relationships

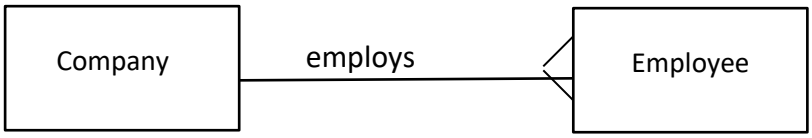
a)



1) Each degree is studied by _____

2) Each _____

b)

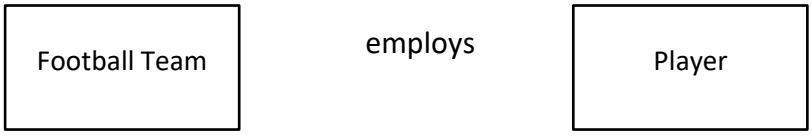


1) _____

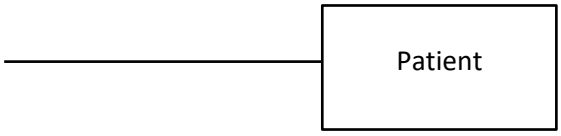
2) _____

3. Complete the entity relationship diagrams below to show the one-to-many relationships

a) Each football team can have many players. Each player can play for only one team.



b) Each ward looks after many patients. Each patient can be resident in only one ward.



Resolving Many-To-Many Relationships

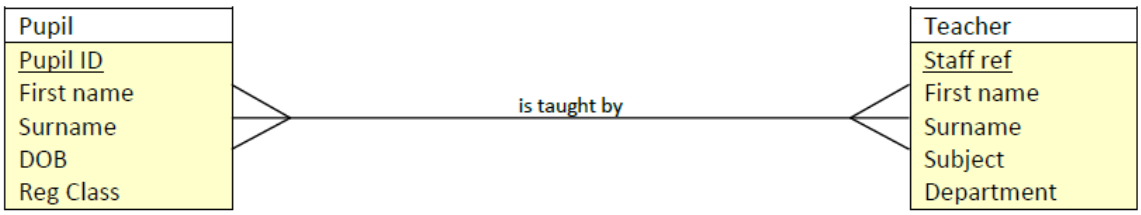
A **many-to-many** relationship **cannot** be implemented in a database system.

Instead, a third entity must be introduced.

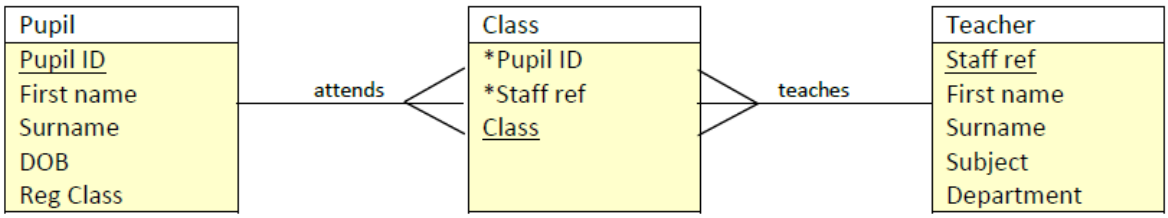
This new entity replaces the many-to-many relationship with **two** separate **one-to-many** relationships.

The new entity includes a **foreign key** from **each** of the other entities.

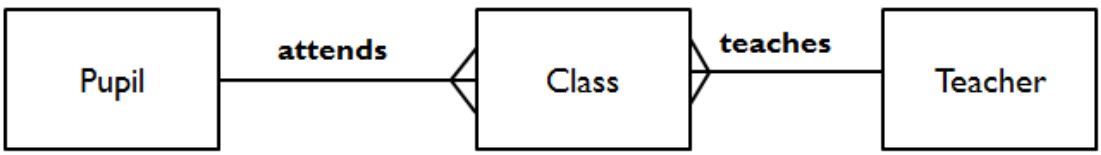
The relationship between Pupil and Teacher is a many-to-many relationship**.**



This many-to-many relationship can be resolved by introducing a new entity:



We can illustrate the relationships between the entities Pupil and Teacher in an Entity Relationship Diagram:



Reading Review 7

Having read page 33, answer the questions below.

1. Express the relationships modelled in the following entity relationship diagrams in English.

a)

_____

_____

_____

_____

_____

_____

b)

_____

_____

_____

_____

_____

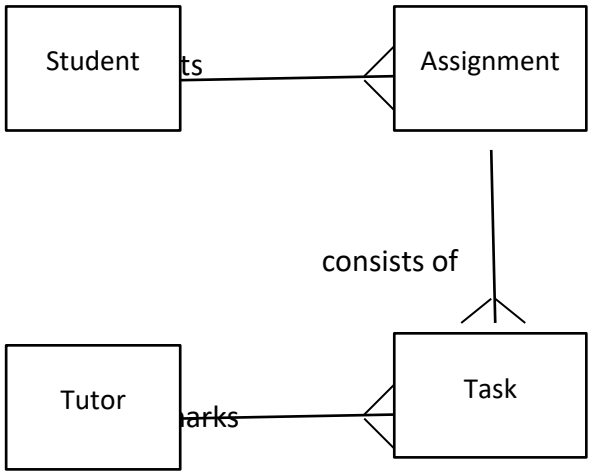
_____

2. A music CD can contain many tracks. A track can appear on many different CDs (for example, on a compilation CD, a greatest hits CD etc).

a) Complete the ER diagram to illustrate the many-to-many relationship.

b) Resolve the many-to-many relationship into two one-to-many relationships. Draw the final ER diagram.

3. Consider the entity relationship diagram shown below. Express each relationship indicated in the diagram in English – remember that a relationship works in two directions.



_____

_____

_____

_____

_____

_____

_____

_____

4. Consider the entities MOVIE and STAR with a many-to-many relationship used to link them.

Draw an entity relationship diagram to represent this many-to-many relationship.

5. Consider the entities PROJECT and DEVELOPER

Draw an entity relationship diagram to represent this many-to-many relationship.

Data Dictionary

A **data dictionary** is used to record details about the database. It provides a description of the constraints or rules that apply to each of the attributes of each entity in the system.

A data dictionary is simply a large table that stores **metadata** – in other words, it stores data about data.

The structures needed to store data in the database are **planned** using a data dictionary and an **entity relationship diagram**.

Entity	Attribute	PK/ FK	Data Type/ Size	Unique	Required	Validation	Format
Customer	CustomerID	PK	Number	Y	Y	>=1 and <=9999	0000
	Firstname		Text (15)	N	Y		
	Surname		Text (20)	N	Y		
	Street		Text (30)	N	Y		
	Town		Text (20)	N	Y		
	Postcode		Text (8)	N	Y		
	Ski Level		Text (12)	N	Y	Choose one of Beginner, Intermediate or Expert)	
Resort	Resort	PK	Text (20)	Y	Y		
	Resort Manager		Text (35)	N	Y		
	Resort Address		Text (50)	Y	Y		
	Spa		Boolean	N	Y		
	Crèche		Boolean	N	Y		

- Each row provides details about **one attribute** in the system
- Each column **specifies a rule** or restriction that applies to the attributes.

Each row of the data dictionary is completed by stating the appropriate value or rule for each column

Where no rules apply, the column value is left empty.

- **PK/FK** (Is the field a primary key / foreign key – a field can be both)
- **Data Types** (Text, Number, Boolean, Date/time)
- **Size** (Maximum number of characters /digits)
- **Required** (Mandatory or Optional. Primary key must be required)
- **Validation** (Presence/Length/Range checks, restricted choice)

Reading Review 8

Having read page 37, answer the question below.

1. A publishing company stores details about books and customer orders. The entities and attributes stored in the system are shown below where primary keys have been underlined and foreign keys are indicated using an asterisk *.

Customer Entity	Order Entity	Book Entity	Author Entity
Customer Number	Order Reference	ISBN	Author ID
Customer Name	Order Date	Book Title	Author Name
Customer Address	ISBN *	Price	Author Address
Customer Town	Quantity	Author ID *	Author Town
Customer Postcode	Customer Number *		Author Postcode

Sample customer orders and book details are shown below.

Customer Orders

Customer Number:	00782
Customer Name:	Inverclyde Books
Customer Address:	52 High Street
Customer Town:	Gourock
Customer Postcode:	PA19 1UX

Order Date	ISBN	Quantity
12/03/2014	0901714564X	9
16/03/2013	7289192000	15
27/03/2013	0901714564X	20
04/04/2013	3781972928	12
11/04/2013	1217292921	9

Book

ISBN:	0901714564X
Book Title:	Weather Time
Price:	£15.99
Author Code:	87281
Firstname:	Julie
Surname:	Adams
Address:	15 West Street
Town:	Greenock
Postcode:	PA19 7XE

Complete the data dictionary for these entities on the next page

Entity	Attribute	PK/FK	Data Type	Required	Required	Validation

Query Design

The design of the SQL query should indicate:

- any field(s)/attributes or computated values required
- the table(s)/entities needed to provide all of the details required
- any search criteria to be applied
- what grouping is needed (if appropriate)
- the field(s) used to sort the data and the type(s) of sort required

Planning the design of a query before creating the SQL code is good practice.

This gives the developer time to think carefully about the fields that are required, which in turns, helps them to identify the table or tables that will be needed.

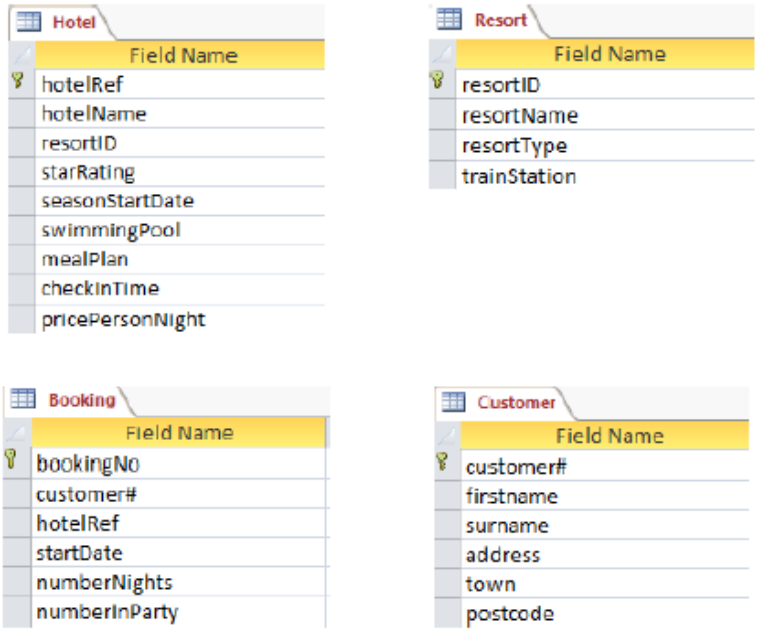
It also allows developers to consider the purpose of the query (search and/or sort), together with any required search criteria and/or sort order.

Planning ahead helps to reduce the errors that developers may otherwise encounter when working with the SQL code.

Example

A travel agency uses a relational database to enable their employees to view details of hotels in Scottish holiday resorts and make bookings for customers. The details are stored in four separate tables called Hotel, Resort, Booking and Customer.

The structure of these tables is shown below:



Example 1: design a query to display the name, swimming pool details, resort and resort type of any hotel in a coastal resort that starts with the letter 'A'.

Field(s) and calculation(s)	hotelName, swimmingPool, resortName, resortType
Table(s)	Hotel, Resort
Search criteria	resortType = "coastal" and resortName like "A*"
Grouping	
Sort order	

Example 2: design a query to display a customer's full name, booking number, start date, hotel name and resort name for all customers who have an 'h' as the second letter of their surname. List these details in alphabetical order of surname; listing customers with the same surname in order of the earliest holiday first.

Field(s) and calculation(s)	firstname, surname, bookingNo, startDate, hotelName, resortName
Table(s)	Customer, Booking, Hotel, Resort
Search criteria	surname LIKE "?h*"
Grouping	
Sort order	surname ASC, startDate ASC

Example 3: design a query that uses a readable heading to display the cheapest and dearest price per night.

Field(s) and calculation(s)	Dearest price per night = MAX(pricePersonNight), Cheapest price per night = MIN(pricePersonNight)
Table(s)	Hotel
Search criteria	
Grouping	
Sort order	

Example 4: design a query to display the average number of nights booked.

Field(s) and calculation(s)	AVG(numberNights)
Table(s)	Booking
Search criteria	
Grouping	
Sort order	

Example 5: design a query to display a list of the different types of resort, together with the number of resorts in each of those categories.

Field(s) and calculation(s)	resortType, COUNT(*)
Table(s)	Resort
Search criteria	
Grouping	resortType
Sort order	

Example 6: design a query to display the number of bookings for hotels in coastal resorts. Show the resort type and use a readable heading for the results returned by the aggregate function.

Field(s) and calculation(s)	resortType, Number of Hotels = COUNT(*)
Table(s)	Resort, Hotel, Booking
Search criteria	resortType = "coastal"
Grouping	resortType
Sort order	

Example 7: design a query to display a list of each type of meal plan, together with the number of bookings made for each of those meal plans. List the details from the least popular meal plan to the most popular.

Field(s) and calculation(s)	mealPlan, COUNT(*)
Table(s)	Hotel, Booking
Search criteria	
Grouping	mealPlan
Sort order	COUNT(*) ASC

Example 8: design a query that uses a readable heading to display the total number of people booked into a hotel in July.

Field(s) and calculation(s)	People booked in July = SUM(numberInParty)
Table(s)	Booking
Search criteria	startDate LIKE "*/07/*"
Grouping	
Sort order	

Example 9: design a SELECT query to display the hotel name and the improved rating, if all hotels in Ayr gain an extra star (use a readable heading to display the improved ratings).

Field(s) and calculation(s)	hotelName, Improved rating = starRating + 1
Table(s)	Hotel, Resort
Search criteria	resortName = "Ayr"
Grouping	
Sort order	

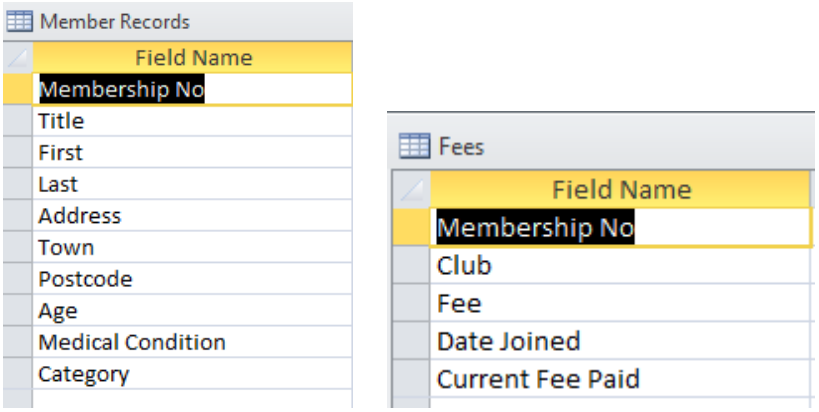
Example 10: design a query to display the surname, booking number, number of nights, number in party, price per night and the total cost of each booking (with a readable column heading). Display the dearest booking first.

Field(s) and calculation(s)	surname, bookingNo, numberNights, numberInParty, pricePersonNight, Total Cost = (numberNights * numberInParty * pricePersonNight)
Table(s)	Customer, Booking, Hotel
Search criteria	
Grouping	
Sort order	numberNights * numberInParty * pricePersonNight DESC

Reading Review 9

Having read pages 40 -44, answer the questions below.

The structures of database tables are shown below:



1. Design a query to list the Membership No, Club and First Name of all members with a first name starting with the letter "L".

Field(s) & Calculations	
Table(s)	
Search Criteria	
Groupings	
Sort Order	

2. Design a query that uses a readable heading to display the cheapest and dearest fee.

Field(s) & Calculations	
Table(s)	
Search Criteria	
Groupings	
Sort Order	

3. Design a query to display a list of the different types of Medical Conditions, together with the number of ages which each medical condition.

Field(s) & Calculations	
Table(s)	
Search Criteria	
Groupings	
Sort Order	

4. Design a query to display the number of members' part of the Ayr Club. Show the date joined and use a readable heading for the results returned by the aggregate function.

Field(s) & Calculations	
Table(s)	
Search Criteria	
Groupings	
Sort Order	

5. Design a query to display the average age of members with a heart condition. List these details in alphabetical order of surname.

Field(s) & Calculations	
Table(s)	
Search Criteria	
Groupings	
Sort Order	

Implementation

N5 Query Revision

Select Queries

```
SELECT Assessment.AssessmentType, Assessment.Title, Assessment.Marks
FROM Assessment
WHERE Assessment.AssessmentType = “Multiple Choice” AND Assessment.Marks <=100
ORDER BY Assessment.AssessmentType DESC, Assessment.Title ASC;
```

Select Queries (Equi-Join)

```
SELECT Student.Forename, Student.Surname, Result.Mark
FROM Student,Result
WHERE Student.StudentNo = Result.StudentNo AND  Result.Mark > 60
```

Update Queries

```
UPDATE Tutor Group
SET Room = 'D15'
WHERE TGCode = ‘TG5’;
```

Insert Queries

```
INSERT INTO Result (StudentNo, AssessmentCode, Mark, AssessmentDate)
VALUES ('101237','X216','99', '2016-10-23');
```

Delete Queries

```
DELETE FROM Result WHERE Result.StudentNo = "101237"
```

Higher Queries

You must be able to implement UPDATE, SELECT, DELETE and INSERT queries making use of:

- Wildcards
- Aggregate Functions (MIN, MAX, AVG, SUM, COUNT)
- Computed Values, Alias
- GROUP BY
- ORDER BY
- WHERE

Wildcards

A wildcard character is used to replace one or more characters in a string.

Wildcards are useful in situations when incomplete information is available and it would be impossible to write a WHERE clause using one of the existing logical operators =, <, >, ≤ or ≥.

Wildcard characters are used with the SQL LIKE operator. The LIKE operator is used in a WHERE clause to perform search operations.

Two different wildcards can be used:

% (the percent symbol) is used to represent zero, one or multiple characters

_ (the underscore symbol) is used to represent a single character

Note: MS Access uses an asterisk (*) rather than % and a question mark (?) rather than __

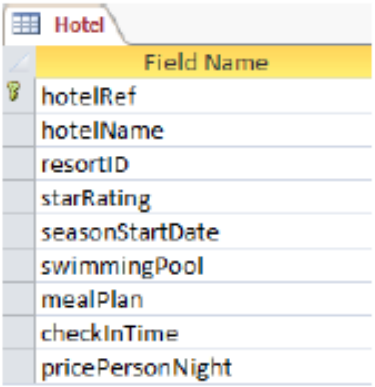
The following are some examples of LIKE used with the wildcards:

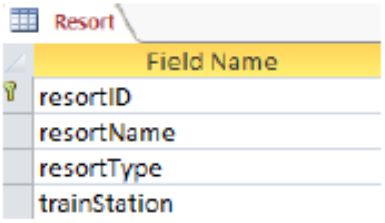
Example	Purpose
`WHERE surname LIKE 'Thom%'`	Used to find any values in the surname field that start with "Thom"
`WHERE surname LIKE '%son'`	Used to find any values in the surname field that end with "son"
`WHERE surname LIKE '%is%'`	Used to find any values that have "is" anywhere in the surname field
`WHERE surname LIKE '_h%'`	Used to find any values in the surname field that have "h" as the second character
`WHERE surname LIKE 'm % %'`	Used to find any values in the surname field that start with "m" and have at least 3 characters
`WHERE surname LIKE 'a%z'`	Used to find any values in the surname field that that start with "a" and end with "z"

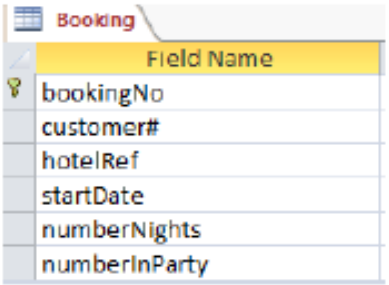
Example:

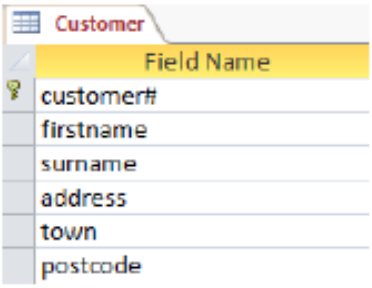
A travel agency uses a relational database to enable their employees to view details of hotels in Scottish holiday resorts and make bookings for customers. The details are stored in four separate tables called Hotel, Resort, Booking and Customer.

The structure of these tables is shown below:









Example 1: used to search the database to display the name, swimming pool details, resort and resort type of any hotel in a coastal resort that starts with the letter 'A'.

```
SELECT hotelName, swimmingPool, resortName, resortType

FROM Hotel, Resort

WHERE Hotel.resortID = Resort.resortID AND resortName LIKE 'A*' AND resortType =
'coastal';
```

Example 2: used to display the customer's full name, booking number, start date, hotel name and resort name of all customers who has an 'h' as the second letter of their surname. These details should be listed in alphabetical order of surname; customers with the same surname should be listed so that the customer with the earliest holiday should be listed first.

```
SELECT firstname, surname, bookingNo, hotelName, resortName, startDate

FROM Customer, Booking, Hotel, Resort

WHERE Customer.[customer#]=Booking.[customer#] AND
Booking.hotelRef=Hotel.hotelRef AND Hotel.resortID=Resort.resortID AND surname
LIKE '?h*'


ORDER BY surname ASC, startDate ASC;
```

Reading Review 10

Having read page 48-51, answer the questions below.

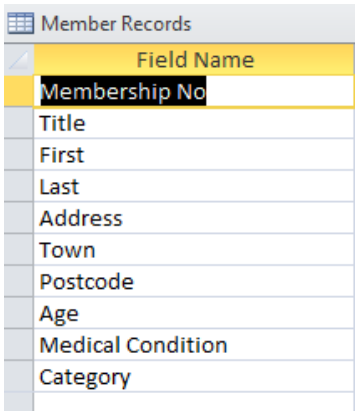
1. Explain the purpose of a wildcard.

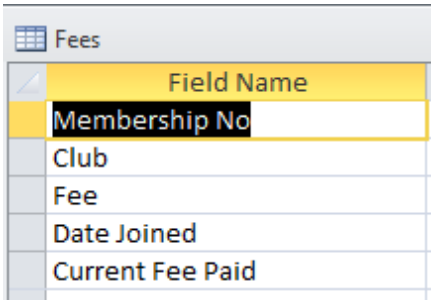
_____

_____

_____

_____





2. Write a query to display the membership number, first name, last name and club of any members' part of the Prestwick club and with a postcode beginning with "K".

SELECT

FROM

WHERE

3. Write a query to full name, town, fee, date joined and age of all members who has an 'e' as the second letter of their first name. These details should be listed in alphabetical order of surname; members with the same surname should be listed so that the member with the earliest joining date should be listed first.

SELECT

FROM

WHERE

Aggregate functions

An Aggregate function is a function that groups together the values of multiple rows to return a single statistical value.

The most common aggregate functions used are listed below:

Function	Description
AVG( )	returns the average value of a numeric column or expression
COUNT( )	returns the number of rows that match the criteria in the WHERE clause
MAX( )	returns the largest value of the selected column or expression
MIN( )	returns the smallest value of the selected column or expression
SUM( )	returns the total sum of a numeric column or expression

In the same way that pre-defined programming functions receive parameter values, SQL aggregate functions require an expression. This expression is usually a column name but it can be a column name together with an operator.

SUM( ) and AVG( ) can only be applied to numeric data types

MIN( ) and MAX( ) work with characters, numeric, and date/time datatypes;

COUNT( ) works with all data types.

All aggregate functions except, COUNT( ), ignore nulls.

COUNT( ) always returns a positive integer or zero. The other aggregate functions return null if the set contains no rows or contains rows with only nulls.

An aggregate expression cannot be used in a WHERE clause.

Use of more than one aggregate expression

It is possible to use more than one aggregate expression in a SELECT statement as shown here:

```
SELECT MIN(price), MAX(price)

FROM Product;
```

Incorrect SELECT Statement

Mixing non-aggregate and aggregate expressions in a SELECT statement is not permitted. A SELECT statement must contain either all non-aggregate expressions or all aggregate expressions. The query below is illegal, as it mixes non-aggregate productName with the aggregate function MAX.

```
SELECT productName, MAX(price)

FROM Product;
```

Readable Headings

Example: Use of readable headings to display the cheapest and dearest price per night.

```
SELECT MIN(pricePersonNight) AS [Cheapest Price per Night],

MAX(pricePersonNight) AS [Dearest Price perNight]

FROM Hotel;
```

Rounded Average

Display the average number of nights booked.

```
SELECT ROUND(AVG(numberNights),2)

FROM Booking;
```

The SQL ROUND( ) function is used to round the average to 2 decimal places.

Count

Used to display a list of the different types of resort together with the number of resorts in each of those categories

```
SELECT resortType, COUNT(*)

FROM Resort

GROUP BY resortType;
```

SUM

Uses a readable heading to display the total number of people booked into a hotel in July

```
SELECT SUM(numberInParty) AS [People on holiday in July]

FROM Booking

WHERE startDate LIKE '*/07/*’;
```

Computed values with aliases

Arithmetic expressions can be used to compute values as part of a SELECT query. The arithmetic expressions can contain column names, numeric numbers, and arithmetic operators.

Whenever a value is generated by a query, it is allocated its own column in the query answer table. A computed value is temporary — it only exists within the query. Because of this, computed values are not stored in the database, which eliminates the need to store data that can be computed at run-time.

An **alias** can be used to give any column in an answer table a temporary name. Doing this makes the headings in the answer table more readable. Since it is generated at run-time, an alias only exists for the duration of the query. An alias is listed in the SELECT list by using the AS statement.

For example, the query below will display the name, price, quantity and cost of each product in a specified order:

```
SELECT productName AS ['Product Name'], price, quantity, price*quantity

FROM Product, Order

WHERE Product.productID = [Order].productID AND order# - 123456;
```

Reading Review 11

Having read page 53-56, answer the questions below.

1. Explain the term aggregate function and give two examples of aggregate functions.

_____

_____

_____

_____

Member Records

Field Name
Membership No
Title
First
Last
Address
Town
Postcode
Age
Medical Condition
Category

Fees

Field Name
Membership No
Club
Fee
Date Joined
Current Fee Paid

2. Write a query to display the highest club fee and lowest club fee

```
SELECT

FROM
```

3. Write a query display the average ages of members.

```
SELECT

FROM
```

4. Uses a readable heading to display the total number of people living in Ayr who have a first name beginning with “D”

```
SELECT

FROM

WHERE
```

5. Use a readable heading to display the cheapest fee price for all memebers who have “Mr” as their title.

```
SELECT

FROM

WHERE
```

6. Uses a readable heading to count the total number of members who joined their club in September.

```
SELECT

FROM

WHERE
```

7. Write a query to display a list of the different types of clubs together with the number of membership numbers in each of those categories

```
SELECT

FROM

GROUP BY
```

Testing & Evaluation

Testing

It is important to test your database queries to ensure the **expected output** matches the **actual output**. Below is an example of how we should test a database:

Consider the **Customer** table shown below.

custo	foreName	surname	street	town	package	directDebit	paymentDu
101	Lauren	Calder	2 Paisley Street	Wemyss Bay	Premier	☑	12/04/2016
102	Abigail	Cameron	16 Paisley Street	Wemyss Bay	Standard	☐	12/04/2016
103	Ryan	Collins	17 Dunoon Drive	Gourock	Premier	☑	19/05/2016
104	Nicole	Rutherford	5A Panama Place	Port Glasgow	Large	☐	13/04/2016
105	Justine	O'Docherty	7 High Street	Kilmacolm	Premier	☑	18/04/2016
106	Shelby	Sweeney	3 Paisley Road	Port Glasgow	Premier	☑	26/05/2016
107	Donald	McAndrew	1 Big Hill Avenue	Inverkip	Standard	☑	01/08/2016
108	Rowan	Hastings	6 Clydeview Crescent	Gourock	Large	☑	05/05/2016
109	Grant	Donaldson	9 Dunoon Drive	Gourock	Premier	☐	07/04/2016
110	Christine	Flowers	63 Hamilton Drive	Greenock	Large	☑	19/04/2016
111	Ross	Lambie	12 Paisley Road	Kilmacolm	Standard	☑	27/03/2016
112	Paul	McGill	3C Cow Lane	Kilmacolm	Premier	☐	10/04/2016
113	Jack	Shields	4 Brookside Close	Port Glasgow	Large	☐	10/05/2016
114	Pauline	Milne	32 High Street	Kilmacolm	Premier	☑	09/05/2016
115	Margaret	Rice	5 Drumchapel Square	Greenock	Standard	☐	14/04/2016

Query testing

A query is required to display the full name and town of all customers who live in Gourock. The details should be listed in alphabetical order of customer surname.

Expected output

This table shows the output predicted from the query.

Expected output	Forename	Surname	Town
Details of customer listed **first**	Ryan	Collins	Gourock
Details of customer listed **last**	Rowan	Hastings	Gourock

Actual output

This is the output from the query used to perform the task.

surname	town
Collins	Gourock
Donaldson	Gourock
Hastings	Gourock

Comparing the answer table with the predicted output, it is possible to evaluate the query.

The query output is accurate because it displays details of the three customers who live in Gourock however it does not display the correct fields so it is not fit for purpose.

Evaluation

When evaluating a completed database system, there are two aspects to consider:

- Fitness for purpose
- Accuracy of output

Fitness for purpose

This is concerned with reviewing whether or not the completed solution carries out all of the functional requirements identified in the initial analysis.

If the solution does not do everything it was supposed to do (and that the user requires) then it is not fit for purpose.

Accuracy of Output

If the SQL queries do not produce the correct results then the database is unreliable. This prevents the database from being an effective and trustworthy system.

It is vital that thorough testing of SQL queries is carried out to ensure that they produce accurate output every time.

Reading Review 12

Having read page 60-61 answer the questions below.

1. Why is it important to test your database?

_____

_____

_____

_____

2. Explain the difference between expected output and actual output.

_____

_____

_____

_____

3. Explain the term fit for purpose in relation to databases.

_____

_____

_____

_____

4. Explain the term accuracy of output in relation to databases.

_____

_____

_____