



RELATIONAL DATABASES

Implementation in a DVD retail chain

An illustrated report showing an understanding and knowledge of relational databases, and
how it would apply to our chain

Christopher Hotten

Contents

Introduction	3
Fields	4
Records	5
Tables	5
Entities and Attributes	7
Entity Relationships	9
Cardinality	10
One-to-One	10
One-to-Many	11
Many-to-Many	11
Optionality	12
Mandatory	12
Optional	13
Mixed	13
Primary Key	15
Foreign Key	16
Candidate Key	17
Composite Key	18
Referential Integrity	19
Normalisation	21
Data Dictionary	23
Making a Relational Database work in our store with an ERD	24
Appendices:	
A Database Schema using Normalisation techniques	27
B Entity Relationship Diagram	28
C Data Dictionary	29

Introduction

Our business requires the use of a relational database. Without one, the store will be unable to operate. Purchasing an off-the-shelf solution at this time would cost too much money, so the only viable option at this stage is to produce a database myself. In order to receive approval, I have produced this report to show my understanding and knowledge of relational databases.

The report explains and analyses the different parts that go into making a database. It explains how a relational database is structured using fields, records and tables. Different concepts about entities and attributes, and how these relate to fields, records and tables are covered. The report also explains what relationships there are between entity types and the purpose of them.

The relationships that exist between different entities are examined in terms of their cardinality and optionality. This report explains the purpose of primary keys, foreign keys (and the use of them in relationships between tables) candidate keys, and composite keys. The report explains the purpose of referential integrity, and also the purpose and benefits of normalisation. Throughout, the report will be contain explains of what specific terms mean.

At the back of the report are the three appendices: an entity-relationship diagram (ERD), an illustration of data relating to our store normalised to third normal form, and a data dictionary.

Fields

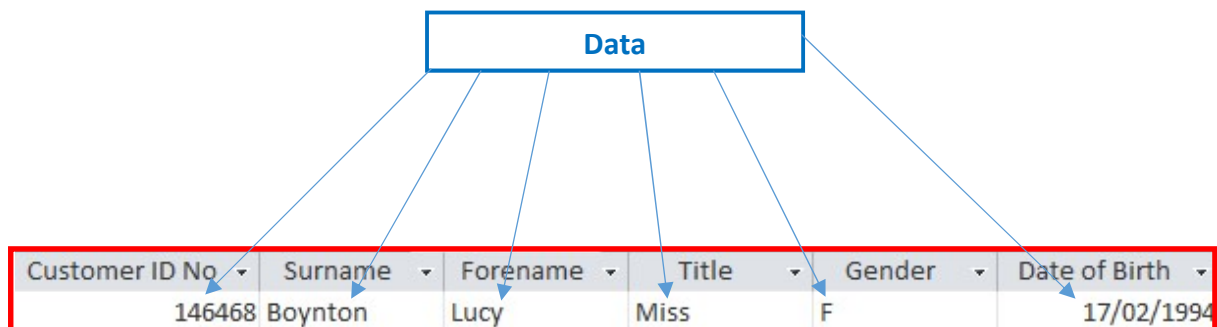
A field is something that will contain only one piece of data about a subject. A database will contain multiple fields, as these are the attributes that make up the characteristics of a subject, and will be the column headers in a table in a database. Different fields will have different names depending on what the database is about. A database that stores information about children in a classroom may have a field for age, a field for height, and a field for gender etc., while a database that holds information about cars in showroom may have a field for model, a second field for licence plate numbers, and a last field for the weight of the car. What the two examples have in common is that each field is only containing one piece of data. A field would not have information about the age and the height of a child, this would have to be split into two separate fields. Figure A below shows some of the fields that could be used for the database for our store, with pieces of data split into separate fields

Customer ID No ▾	Surname ▾	Forename ▾	Title ▾	Gender ▾	Date of Birth ▾
146468	Boynton	Lucy	Miss	F	17/02/1994

Fig A: Data about a customer split into different fields of Customer ID, Surname, Forename, Title, Gender and Date of Birth

Records

A record (known as a 'Tuple') is made up of the fields and the data inside them. Records hold the data about *things*. A record can hold data about a person. A record can hold data about a car. A record can hold data about a company. The central point is that a record should only relate to one person or subject. A database will hold information about multiple subjects, so the database will contain multiple records, with each record containing the fields and the data in them that were input earlier. Figures B and C below show examples of records that could be held in our database



The diagram illustrates the relationship between a record and its fields. A box labeled 'Data' is positioned above a table. Six arrows originate from the 'Data' box and point to each of the six columns in the table: 'Customer ID No', 'Surname', 'Forename', 'Title', 'Gender', and 'Date of Birth'. The table contains the following data for Lucy Boynton:

Customer ID No	Surname	Forename	Title	Gender	Date of Birth
146468	Boynton	Lucy	Miss	F	17/02/1994

Fig B: The record for Lucy Boynton, which contains the separate fields and the data which relates to each field

Customer ID No	Surname	Forename	Title	Gender	Date of Birth
244884	Duncan	Michelle	Ms	F	14/04/1978

Fig C: A separate record for Michelle Duncan. The field names have not changed, but the data within them will relate to Michelle Duncan only in her record.

Tables

Tables (or 'Datasheets') contain the records that contain the fields that contain the data. In the paragraphs and examples above it was shown how the information (or *data*) about a customer would be split into different fields (surname, forename, title, gender etc) with the fields and data then making up the record that relates to the individual customer. To run the business and find data efficiently, the records need to be stored in one place; in this

case, a table. A table holds all the records together *that are related to a subject*. The records for customers would be held in a table. A separate table would hold the records about DVD's we stock. A third table would be used for storing information about staff who work at our store. To store all the information about three different subjects (customers, DVD's and staff) in one table would be impractical and a waste of system resources, so the records relating to a specific subject would be stored in a specific table.



Customer ID No	Surname	Forename	Title	Gender	Date of Birth
146468	Boynton	Lucy	Miss	F	17/02/1994
174871	Hollander	Tom	Mr	M	25/08/1967
244884	Duncan	Michelle	Ms	F	14/04/1978
356454	Crerar	Leila	Dr	F	30/03/1997
443444	Hardy	Ben	Mr	M	02/01/1991
484314	Gwilym	Lee	Dr	M	24/11/1983
646688	Das	Meneka	Mrs	F	03/05/1976

Fig D: The records of all the customers are now in one table labelled 'Customers'



Film Name	Genre	Run Time (n)	Distributor	Store Catalog No
Despicable Me 2	Adventure	98	Universal	4896745
Fast and Furious 6	Action	130	Universal	6188476
Frozen	Musical	102	Walt Disney	6285546
Gravity	Thriller	91	Warner Bros	7888522
Grown Ups 2	Comedy	101	Sony Pictures	7898444

Fig E: The records of all the DVD's are now in one table labelled 'DVD Data'



NI Number	Surname	Forename	Pay Band	Tax Code	House No	Street/Road No
BT638725V	Brennan	Bethany	A	10000L	5	Ramsey Street
DS168546A	Dingle	Tricia	C	10000L	11	Emmerdale Farm
GH245372M	Duckworth	Jack	E	15000D	27	Coronation Street
HE426987S	Robinson	Charlene	D	12000M	10	Lucky Lane

Fig F: The records of all the employees are now in a table called 'Staff Personal Details'. Notice that while 'Customers' and 'Staff Personal Details' both refer to people, the records have different fields

Entities and Attributes

Simply put, an entity can be an object or *thing*. In the previous section, it was explained how records can hold information about a person, about a car, or any object you desire. An object or thing that we want to collect data about in a database is called an *entity*. In our database there will be different entities, or *entity types*. All the customers of our store would be an entity type called 'Customers'. All the staff would be an entity type called 'Staff'. Remember, an entity can be an object, so the DVD's would be their own entity type, the service points would be their own entity types, and so on. All these different entity types will have different information (or *Attributes*) that we need to collect about them.

For the staff at the store, we need to know his/her individual National Insurance number and tax code so that we can make sure we are paying the correct person the correct amount of money. We may also need body measurements so that we can order the correct size store uniform. The NI number, tax code and body measurements, these would all be attributes that the staff have. For a customer of the store, their body measurements are irrelevant to us. We may need to collect different data though, like their preferences of film genre, so the customer would have different attributes to a member of staff. A DVD obviously has no information like a National Insurance number or gender to collect, but it will still have attributes (just like any entity), such as the title of the film, the length in minutes, the distributor etc. Attributes are also referred to as fields (see first section for an explanation of fields), and when the data is collected, each object will have their own record.

When all the records about the same subject or object are put together, we have the created table – or *entity type*. While an entity type will have a name that allows for easy recognition – e.g. Staff, Customer, DVD etc., the table which holds the records does not necessarily need to have the same name as the entity type. A table holding information about customers could have a name like 'C1', or 'CUST' for instance. Whatever the name, it will be referring to the entity type called 'Customer', and can only refer to that entity type.

CUST									
	Customer ID No	Surname	Forename	Title	Gender	Date of Birth	House/Flat	Street/Road	Address Line 2
+	146468	Boynton	Lucy	Miss	F	17/02/1994		16 St Georges Way	
+	174871	Hollander	Tom	Mr	M	25/08/1967		35 Front Street	Langley Park
+	244884	Duncan	Michelle	Ms	F	14/04/1978		30 Brentwood Av	Jesmond
+	356454	Crerar	Leila	Dr	F	30/03/1997		4 Leybourne Hol	Birtley
+	443444	Hardy	Ben	Mr	M	02/01/1991		6 Market Street	Dudley
+	484314	Gwilym	Lee	Dr	M	24/11/1983		43 Station Road	Forest Hall

Fig G: A table showing the different fields (or attributes) of customers

In Figure G above, data has been collected about customers, with the data being split into different fields, like surname, forename, title, gender etc. – these are the attributes of the customer. There are multiple records (e.g. a record for Lucy Boynton, a record for Tom Hollander, a record for Michelle Duncan etc.) which are together in the table. The records are also referred to as entity instances. When we see the information about a customer, such as Lucy Boynton, born 17/02/1994 and living at 16 St George’s Way, we see the record – or entity instance. This is an instance from the entity type ‘Customer’. Notice at the top of the image that the name of the table is ‘CUST’. We could have given it any name, but it always refers to the entity type ‘Customer’.

Entity Relationships

In the previous section it has been established what entities are and the different entity types that would exist in a database specific to the DVD store. In our store, different entity types will interact with each other in different ways. A customer could rent a DVD. This means that the entity type 'Customer' would have a relationship with the entity type 'DVD' – or an *entity relationship*. Staff on a certain pay band could work on different departments in the store, so there would be relationships between staff and departments. Customers (or the entity type 'Customer') however would not work in any department at the store, so would have different entity relationships to that of the staff (or entity type 'Staff'). Relationships are the links between entities. Relationships are used so that fields are not trying to hold more than one piece of data and tables do not become unwieldy.

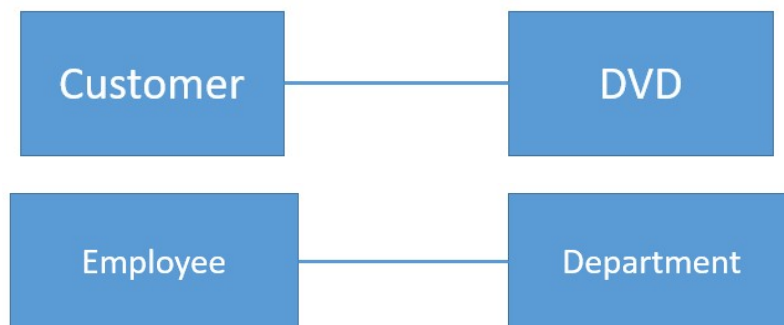


Fig H: Image showing a relationship between entity types 'Customer' and 'DVD', and entity types 'Staff' and 'Department'

Figure H above shows a simplified relationship between entity types – the customer would need to be able to rent a DVD, so a line is drawn connecting the entities types, but would not need to work in a department in the store, so no line exists between the customer and department.

Cardinality

Different entities will have different degrees of a relationship with other entities, which is what cardinality refers to. The relationship between entities is visualised in an 'Entity Relationship Diagram' (ERD). An ERD helps the designer of a database to better understand the data that is needed in a database. In visualising the cardinality, there are three different degrees to explore.

(Appendix B is an illustration of an ERD for our store, a detailed breakdown is provided later in the report under 'Making a Relational Database work in our store with an ERD' after other relevant terms have been explained)

One-to-One Cardinality

The first degree of cardinality is 'One-to-One' (shortened to '1:1'). This is where only one instance of an entity can only relate to another instance from a different entity. A common example of this is something like a marriage. A man can only be married to one woman (and vice versa). So one entity instance of a man can only be married to one entity instance of a woman. In Figure H of the 'Entity Relationships' section, a simple relationship is shown for an employee working in a department. If the one employee can only work in the one department this may be quite limiting. Also if one instance of a customer can only ever rent the one instance of a DVD then the business will obviously not last long, so this type of relationship would not be beneficial for us.

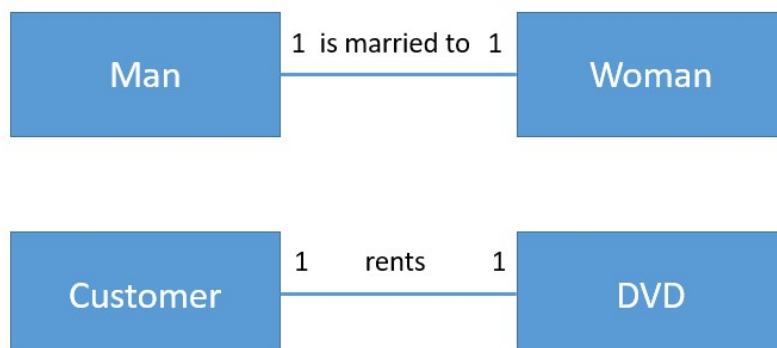


Fig I: Image showing a 'One-to-One' relationship. The '1' next to each rectangle shows the one instance, or "at most one instance"

One-to-Many Cardinality

The next degree of relationship is 'One-to-Many' (shortened to '1:M'). The first part of this degree is the same as the One-to-One degree, however the relationship is now with many instances. This degree of relationship allows for one instance of an entity to have a relationship with multiple instances from a different entity. In our store a manager will be managing many members of staff, so therefore a 'One-to-Many' degree of cardinality would be applied here. If a manager was managing only one member of staff (or one instance of the 'Employee' entity type), then this would be a 'One-to-One' degree of cardinality.

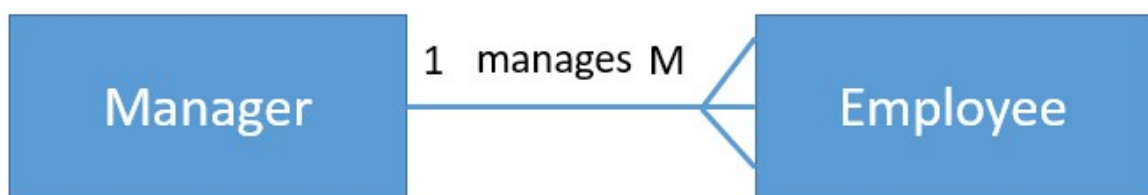


Fig J: Image showing a 'One-to-Many' relationship. The '1' on the left has stayed the same from figure I, but the '1' on the right side is now replaced by an 'M' to represent 'Many'. Notice the three lines (or 'crows foot') now connected to the Employee box to show multiple instances.

Many-to-Many Cardinality

The final degree is a 'Many-to-Many' relationship (shortened to 'M:N'). This relationship occurs where many instances of one entity can have a relationship with many instances of a different entity. In our store will be many copies of different films and we plan to let customers rent more than one film at any one time, so this is a 'Many-to-Many' relationship, as shown in figure K below

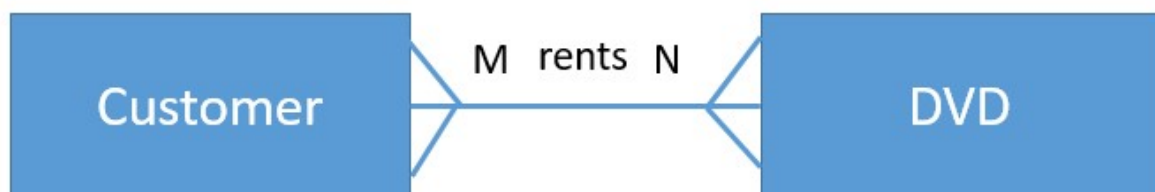


Fig K: Image showing a 'Many-to-Many' relationship, where many instances of a 'Customer' can rent many instances of a 'DVD'. The 'N' also stands for 'Many'. Notice that separate 'crow's feet' are now attached to both boxes in this relationship

Optionality

In the section above, the different degrees of relationship (or cardinality) have been explained, but what is explored below is the optionality of the relationships. Optionality refers to the options that can exist in a relationship – whether a relationship is mandatory, optional or mixed.

Mandatory

In a mandatory relationship, one entity has to use another. A common example of this would be that a child *has* to have a mother, else the child would not exist, and for there to be a mother there has to be a child. So for every instance of entity A, there must be an instance of entity B. In a commercial store, a customer would only become a customer after receiving his/her custom, meaning that a store must have one received at least one order.

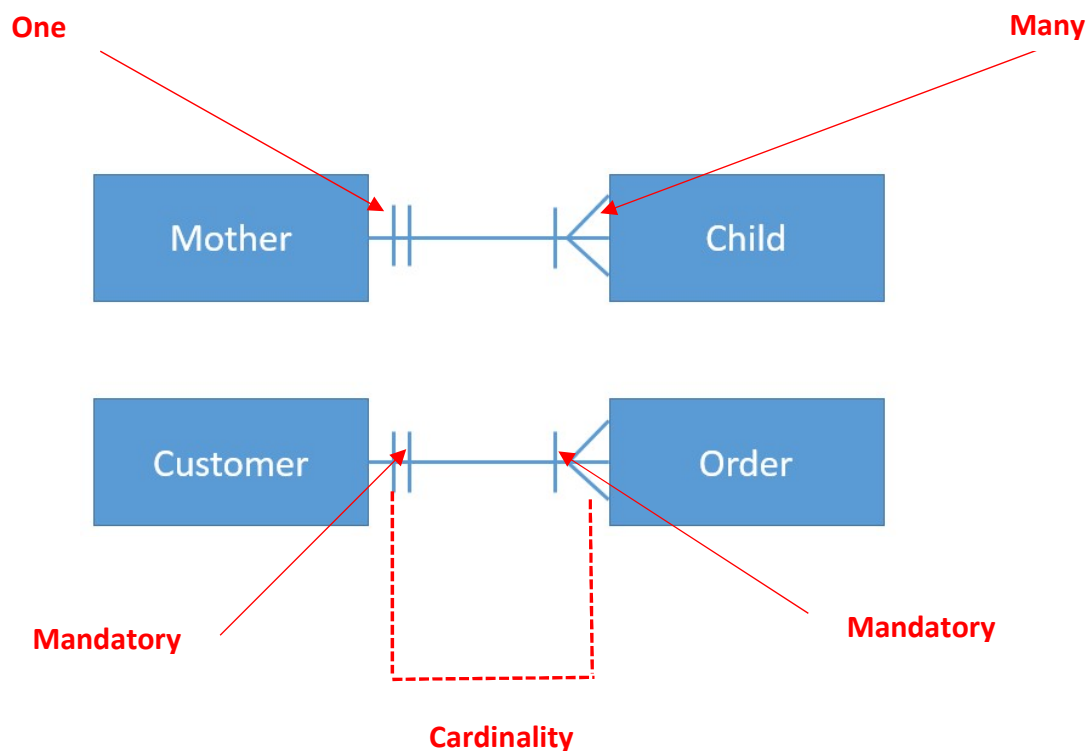


Fig L: Examples of mandatory relationships. A mother and child are both mandatory for each other. Multiple children can have the same mother, but multiple mothers cannot have the same child. A customer cannot exist without having placed an order, and an order can only be made by a customer placing one. Also shown are extra lines added to show the mandatory cardinality of the relationship

Optional

As the name suggests, in an optional relationship there may or may not be a matching record in each entity. In our store, there will be different genres of films. For example, there could be a genre called 'Arthouse'. In this section there could be none, one or many films. One film may or may not be described as being 'Arthouse'. Whether this would make financial sense to have an empty section is not relevant. The important part is that an optional relationship is being described here. It is not mandatory for an 'Arthouse' genre to have any films in it, and it is not mandatory for a film to be in the 'Arthouse' genre.

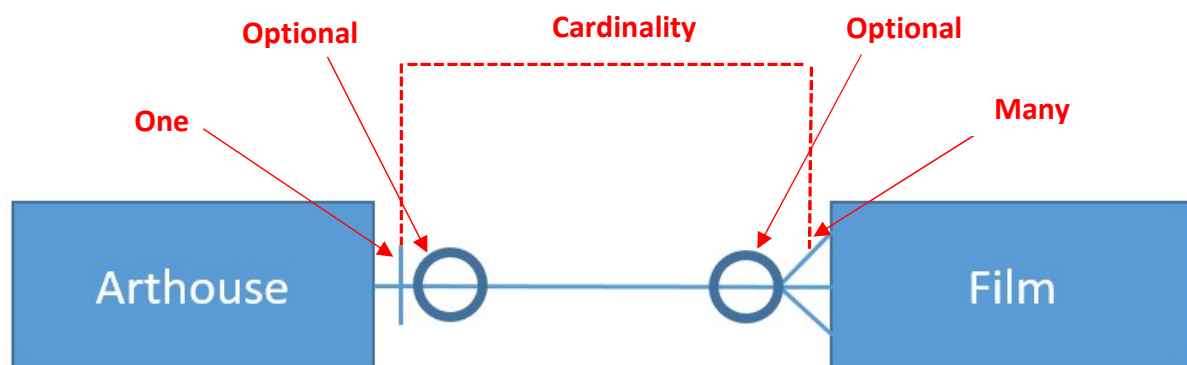


Fig M: An example of an optional relationship. The genre 'Arthouse' could have none, one or many film titles in it. A film may or may not be in the 'Arthouse' genre. The vertical line next to the Arthouse entity represents one and the 'crow's foot' next to the entity type 'Film' represents many, to show a 'One to Many' relationship. Note the added circles, which show the 'zero or many' optionality of the relationship.

Mixed

A mixed relationship is where mandatory and optional relationships occur within the same relationship. In a mixed relationship one entity A requires the use of entity B, but entity B does not require entity A to exist. Earlier, it was explained that a mother and child, and a customer and an order, are in mandatory relationships; one cannot exist without the other. In a mixed relationship though, one entity can exist with or without another entity. Our store will be a medium-sized rental store, which will likely require just one manager and several

floor staff who will be required to report to the manager. If our store was much smaller though, it could be run by one person. That person would manage the store by his/herself, and so would be referred to as a manager by the company and the entity type would be 'Manager', regardless if anyone else worked there or not. If an employee was taken on though, that person would be required to report to a manager. If multiple additional employees were taken they would be required to report to the store manager. We have then the mandatory part (employees reporting to a manager) and an optional part (a manager does not need to manage any staff to be referred to as a manager and so the entity type can exist without employees) to create a mixed relationship.



Fig N: An example of a mixed relationship. A manager can manage none, one or many employees, but one or one or many employees have to report to a manager. Note the two vertical lines next to the 'Manager' entity to show 'one and only one', and the circle next to the 'crow's foot' to show the 'zero or many' optionality of the mixed relationship.

Primary Key

A primary key is an attribute that is unique about an entity, enabling us to identify a single record. In other words, every record in a table will have something specific about it that sets it apart from everything else. A primary key can be letters, numbers, or a mixture of both. For people this could be their National Insurance number. The NI number one person has will be different to the NI number of anyone else. The serial number of an object will be unique to that specific object. The objects may all look the same. They may be the same colour, same dimensions, be the same weight, but that serial number will make it unique in a table. It is important to note that the value that makes something unique is never used again.

If a person dies, then that person's National Insurance number could never be used again. It may be the only unique identifier that person has, making it the only way to identify someone if information about that person was to be found. For reasons like this, primary keys are mandatory; every table in a relational database must have a field which is marked as the primary key

Knowing that a primary key is mandatory for a relational database, in all of our tables there will be a field marked as the primary key. We are identifying the attribute that makes each instance of the entity unique. For all the staff who work at the store this could be their National Insurance Number, as this would be unique to each employee. In the staff records we could have how much that person is paid, forename, surname, how tall they are, but none of these would be guaranteed to be unique. Two members of staff may have the same forename, or be the same height, so those attributes would not be unique. For objects like DVD's or staff passes we could assign individual ID numbers.

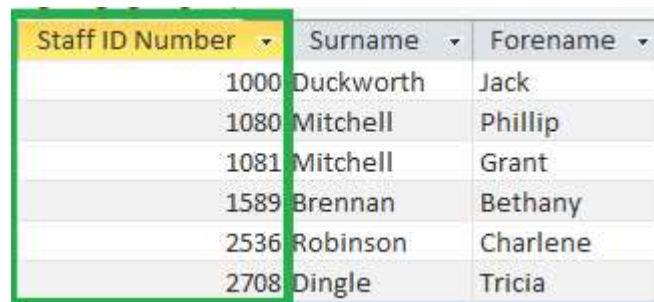
NI Number	Surname	Forename	Pay Band	Tax Code	House Num
BT638725V	Brennan	Bethany	A	10000L	5
DS168546A	Dingle	Tricia	C	10000L	11
GH245372M	Duckworth	Jack	E	15000D	27
HE426987S	Robinson	Charlene	D	12000M	10
JE445587C	Mitchell	Phillip	A	10000L	1
JE445590C	Mitchell	Grant	B	10000L	1

Fig O: A table with the primary key being the National Insurance number of a person. Highlighted in orange are two people with the same attributes for surname, tax code and house number, so the only unique identifiers for these two people are their NI numbers.

Foreign Key

A foreign key is what enables data in tables to be connected, as it enables a link to be made between entities, giving us the relationships between them. Previously it was explained that primary keys are unique attributes about an entity. What about when we want to use those unique attributes in a different entity (or table) that will also have its own primary key? A table can only have one primary key, so a primary key that appears in a different table needs to be identified as a foreign key. This concept is explained in more detail below.

In the table named 'Staff Details' (which represents the entity 'Employee') it was identified that the National Insurance number of each person is unique, and everyone must have one, so this is the attribute that is marked as the primary key. In our store, employees also have to be given ID badges to wear that also enable use of the service points, and so the ID badges will be an entity called 'ID Badge' for instance, and each one will be have a unique badge number assigned to it.



Staff ID Number	Surname	Forename
1000	Duckworth	Jack
1080	Mitchell	Phillip
1081	Mitchell	Grant
1589	Brennan	Bethany
2536	Robinson	Charlene
2708	Dingle	Tricia

Fig P: A table containing unique staff ID badge numbers

There will be various attributes for the entity 'Employee', like address, phone number, date of birth etc. If a member of staff leaves/is dismissed from the company and did not return his/her pass, we need to know which pass to disable on the system so no unauthorised logins are made to the service point. The ID badges will only contain data like the unique badge number, forename and surname of the person, but if two people have the same forename and surname then we would not know which ID badge to disable.

If we connect the entity 'ID Badge' to the entity 'Employee' through the use of the unique badge number, then we have made a link between the tables and so a relationship is formed between the 'ID Badge' and 'Employee' entities.

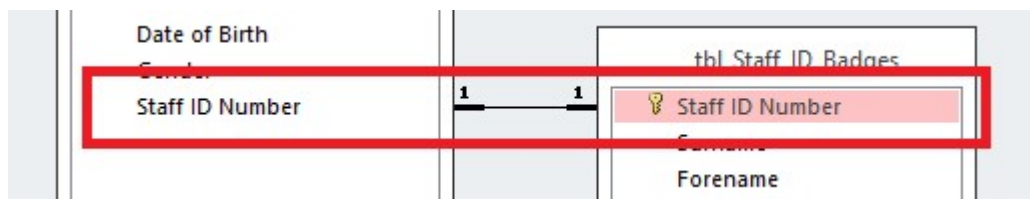


Fig Q: A visual representation of the primary key in the 'Staff ID Badges' table being inserted as the foreign key in a table containing staff details

NI Number ▾	Surname ▾	Forename ▾	Pay Band ▾	Staff ID Nun ▾
BT638725V	Brennan	Bethany	A	1589
DS168546A	Dingle	Tricia	C	2708
GH245372M	Duckworth	Jack	E	1000
HE426987S	Robinson	Charlene	D	2536
JE445587C	Mitchell	Phillip	A	1080
JE445590C	Mitchell	Grant	B	1081

Fig R: A table containing staff details that also now has the staff ID number foreign key in it, that was the primary key in the 'Staff ID Badges' table

The primary key of the 'IB Badges' table is now connected to the 'Staff Details' table, so the badge number identifier is now a foreign key in the 'Staff Details' table. This means that if a member of staff leaves/is dismissed, we are able to look up the record and disable the appropriate ID badge.

Candidate Key

A candidate key is any field in a table that can be used to uniquely identify a record without referring to any other data. There may be multiple fields in a table that uniquely identify a record. Whichever field is chosen to be the unique identifier of a record, that candidate key then becomes the primary key. In tables there may only be one field that can make each record unique. In the example for a primary key, figure O showed that the National Insurance number of a member of staff would be the unique identifier for each person. None of the other fields would qualify as a candidate key as there is no guarantee that the data in them would be unique (i.e. forenames, surnames, dates of birth; none of these are unique).

If we had originally expanded our table to include the NHS number or birth certificate number of a person, then these would be candidate keys as the number is unique and never repeated.

Composite Key

A composite key is created when two or more columns in a table are combined to identify each record in a table. The columns by themselves would not be able to uniquely identify a record, but when they are combined this allows identification. This means that a composite key has the same function as a candidate key (enabling unique identification of records), but because it relies on two or more columns being combined, it is a special type of candidate key. To help explain this, the table containing the staff details will be referred to again. The primary key was established as being the National Insurance Number of a member of staff as this is a unique, non-repeating number.

If we had included the NHS number or birth certificate number of every member of staff, these would be candidate keys due to their uniqueness, but what if none of these were available? As none of the other fields would contain data that is guaranteed to be unique, we would have to combine fields in order to guarantee unique identifiers. If we combined the forename, surname, the first line of address and the date of birth of a member of staff together as one composite key, it is likely that this would make each record unique, but is not guaranteed at all (i.e. it is unlikely but there could be two people living at the same address who have the same name, same date of birth and work at our store)

We would have to include more and more fields in order to guarantee uniqueness, which is why something like a National Insurance number is so important to have in our relational database.

Referential Integrity

Referential integrity refers to making sure that the data in one table does not contradict the data in another table. It is a type of data integrity, and to make sure we are preserving the integrity of our data (i.e. making sure the data between tables do not contradict each other) we need to create rules that need to be observed. In the previous section there is an explanation of what a foreign key is and an example of it in use in our store, with regards to staff details and the ID badges they are assigned. The staff ID numbers are all four digits long, with no letters involved. This is the rule for the primary key in the 'ID Badge' table and also for the foreign key in the 'Staff Details' table. If a decision was made to change the rules for the data in the primary key field of the 'ID Badge' table, such as instead of being four digits long, the staff number was made up of three letters and two numbers, we need to make sure that we make the change where the foreign key appears in the 'Staff Details' table, else we are not keeping the integrity of the data between the tables.

NI Number ▾	Surname ▾	Forename ▾	Pay Band ▾	Tax Code ▾	Staff ID Number ▾
BT638725V	Brennan	Bethany	A	10000L	RTG78
DS168546A	Dingle	Tricia	C	10000L	DSE25
GH245372M	Duckworth	Jack	E	15000D	DCS46
HE426987S	Robinson	Charlene	D	12000M	TGY88
JE445587C	Mitchell	Phillip	A	10000L	PES34
JE445590C	Mitchell	Grant	B	10000L	CVB75

Fig S: A table containing staff details that also now has the staff ID number foreign key in it, but the data rules have been changed from the Staff ID number being four digits long only, to being a combination of three letters followed by two numbers

Staff ID Number ▾	Surname ▾	Forename ▾
1000	Duckworth	Jack
1080	Mitchell	Phillip
1081	Mitchell	Grant
1589	Brennan	Bethany
2536	Robinson	Charlene
2708	Dingle	Tricia

Fig T: The 'Staff ID Badges' table which has not had the data rule changed, so the ID numbers are still four digits long only

If we did not observe this rule change between the tables, the data would be inconsistent and the integrity is compromised. In the example that is used in the foreign key section, this would mean that we would no longer be able to identify which badges are assigned to which members of staff, with records becoming lost (or 'orphaned') when queries are run in the database, as the queries would only bring up results that matched the criteria that has been set in a search. This means that if we change the rules for what an ID badge number is in the 'ID Badges' table, then we must apply the same rule to the foreign key in the 'Staff Details' table to keep the integrity of the data.

Normalisation

Normalisation is the process of organizing a database to reduce redundancy and improve data integrity. What this means is that we organise a database in a way that reduces the amount of data in a database while still having all the required attributes (or fields) for entities (or tables). It also means improving the chances of keeping the data consistent across different tables. Normalisation is also important as it breaks a structure down into *atomic elements*. This means that we take data and break it down into further parts until it cannot be broken down further. A typical example of this is a person's name. A name like John Smith could appear in a database under a column simply called 'Name', but it would be more useful to split this into columns of 'Forename' and 'Surname'. Since neither can be broken down any further, the forename and surname would be referred to as atomic elements.

There are multiple benefits to using normalisation. It minimises the amount of duplicate data (otherwise known as *redundancy*) in a database. This is important as it helps with referential integrity, as multiple tables do not need to be updated separately with the same new values for data (i.e. the amount someone is paid only needs to be updated in one table, instead of several) and also keeps down the actual file size of database, which is important when our database is operational as the network will struggle when trying to deal with a large database that has unnecessary repeats of data. While it may initially appear a negative factor that the process of normalisation can add more tables by the end of the process, this becomes a positive, as it becomes easier to search and sort through tables for information, as we do not need to try and squeeze an uncomfortable amount of columns in a page.

The normalisation process involves putting information into a flat file database (or ONF), then, by arranging the data in tables and columns (referred to as a *schema*) we can break down the data into 1st, 2nd and 3rd Normal Forms (or 1NF, 2NF and 3NF respectively). It is possible to go past 3NF to 4NF and 5NF, but that will not be required for our database. Appendix A of this report is a schema using the data that will be in our database normalised to 3NF.

In the schema for our database we can see that the data starting in the ONF column is just in a flat structure. Nothing has been organised into tables, and there may be the chance that in this structure there would be repeating data. An example of this would be in the 'PayPerAnnum' column. If lots of staff are on the same wage, this would have to be input

multiple times and would take up unnecessary amounts of room, and could also possibly lead to wrong amounts being entered. In the 1st Normal form the data is now organised into tables and columns.

The rules of 1NF is that each table cell should contain a single value and each record needs to be unique. Some of the columns would only relate to the customer, some to the staff, and some to the DVD's in our store. The way the tables have been organised, in each column there would only be one piece of data, and each record would be unique. So in the table that is headed by the 'StaffID Number' column and at the bottom is the 'PayPerAnnum' column, all the single pieces of data that would be in the columns in between those two would make every record unique.

As we move in into the 2NF, a central question that helps organise the data for us is whether the columns in the table are dependent on the primary key. Looking at a table in the 1NF column will help explain this. In the table that starts with 'DVDStoreCatalogueNumber' there is 'DateRented' and 'DateReturned'. These two columns are not dependant on the individual catalogue number that we assign for each DVD, and they would not help to identify a DVD in the store. Also in that table though is a column called 'RentalIDNumber', and when a film is rented and returned would be dependent on the number that we assign the rental transaction, as just on their own the dates would mean nothing, so the in 2nd Normal Form that tables contain columns which are reliant on the primary key (the primary key column is in bold).

In the 3NF for our data, a new rule is introduced – *tables in 3NF can only contain columns that are non-transitively dependant on the primary key*. There are two parts to this term that need to be explained. The *transitively* part can be described as reviewing if one column is related to others through another column. The *dependence* part relates to if the value if one column is dependent on the data in another column. The schema in Appendix A helps to explain this concept.

Primary Key	Column A	Column B
Customer ID	CustForename	CustSurname

In this example there is no transitive dependence, as a person's surname has no effect on what their forename is

Primary Key	Column A	Column B
DVDCatalogueNumber	Certificate	CertificateDescription

In this example there is transitive dependence. The certificate description will depend on what the certificate is. If a certificate that was previously PG-13 is changed to something like PG-11, this would change the description of who is allowed to see the film and so we would have to check information for new values when customers are renting out films. The certificate description is not dependent on the catalogue number for the DVD but is dependent on the certificate. To make sure the rule of 3NF is observed, the certificate and description of the certificate are in their own table, with the primary key being 'Certificate'. That primary key then appears in the table headed by 'DVDStoreCatalogueNumber' as a foreign key, so this means that the table does not have any transitive dependencies, but can also still contain information about what certificate rating the film has, without having to repeat data about the description of the certificate.

Data Dictionary

A Data Dictionary is used for describing all the elements in a table within a database. It serves as a collection of descriptions about the fields in a table and the records in them. The dictionary can then be used by anyone who is working on the back-end of the system as they will be able to see the rules that have been established for data (e.g. a date of birth has to be in DD/MM/YYYY format). Appendix C of this report is a data dictionary that shows the fields that will be in our database. The data dictionary for our relational database contains the names of all the tables, the Primary Key field, field names, data types, length of field, the validation rule, a description of what the field is used for, and finally typical data that would appear in that field.

The rules that have been established are called 'Validation Rules', and these rules can be set for every field that requires it. For instance, in the National insurance Number field, the

format will always have to be two letters followed by six numbers and one number at the end, as this is how NI numbers are always formatted, so a rule would have to be set that a NI number can only be entered in this format. If an attempt is made to enter an NI number in a different format, the field will simply refuse to accept the data. Appendix C is the data dictionary that is being used for our database. An analysis of a field in one of our tables will help to explain these terms.

Field Name: CustPostcode

The name that has been given to the field

Data Type: Short Text and Field Length: 8

The postcode for a customer will never need to be a large sequence of numbers and letters, so we know that the type of data required will just be a relatively short sequence of letters and numbers, and the length of the field has been specified as eight, as this is all a postcode will need (i.e. two letters, two numbers, a space, one number then two letters, making in total eight spaces required)

Validation Rule

In this we can make a rule that no special characters can be input (e.g. an asterisk or ampersand) and that the last three characters should only be one number followed by two letters and that the total number of characters will be no more than eight)

Description

Simply a description of what the field is used for (e.g. the postcode of the customer)

Example

NE13 4FG

Making a Relational Database work in our store with an ERD

All the sections above have explained and given examples for various terms, as what they refer to will be used throughout our database. Three appendices appear at the back of this report, which are specially related to the relational database our store will use.

Appendix B is the Entity Relationship Diagram (ERD) for our store. In the diagram appears the entities, the cardinality and the optionality of the relationships. Below I propose the logic behind the decisions of the relationships between the entities so that anyone else who works on this database will be able to understand. The central question I have asked myself for the relationships is this: Does entity 'A' *have* to have an entity 'B'? If so, a relationship is mandatory, if not, it is optional, and the same question is asked of entity 'B' to entity 'A'

(Starting from top left to right, then working down)

Certificate Entity and DVD Entity

A certificate *does not* have to have a DVD. Every instance of a Certificate does not require a DVD, so an instance of a Certificate can be used for none or many DVD's

A DVD *does* have to have a rating certificate. It is illegal to sell films that do not have a rating certificate. For every instance of a DVD, there must be one (and only one) Certificate attached

Genre Entity and DVD Entity

A genre *does not* have to have a DVD. Every instance of a Genre does not require a DVD, so an instance of a Genre can be used for none or many DVD's

A DVD *does not* have to have a genre. There are rare films that defy all genres which we can still sell. If a DVD does have a genre though, it will only have one genre (this is mainly for easy placement of cases in our store), so it is an optional to have Genre, but at most it will have one Genre.

DVD Entity and Rental Entity

A DVD *does not* have to have a Rental to exist. Before our store even opens, the database will contain records of hundreds of DVD's. For every instance of a DVD, there does not need to be an instance of a Rental, though many rentals can use many DVD's, so it is an optional at this end

A Rental requires at least one DVD. We rent out DVD's, so for every instance of a Rental, there *has* to be an instance of at least one DVD attached. With no DVD's, there can be no rentals, so this is mandatory

Rental Entity and Customer Entity

For every rental there *has* to be a customer. For a Rental entity to exist, a Customer entity must exist (as a rental has to be made by a customer). A rental record will include a customer ID number, so one or many films can be rented out, and it has to be tied to one instance of a Customer in a mandatory part of the relationship

A Customer does not need a Rental to exist. In our store, someone becomes a customer once they have created an account with us (this is how we assign a customer ID number to that person), to create an account with us they do not need to rent a DVD. That person may never rent a film or could rent many films at one time. This is an optional part of the relationship.

Pay Band Entity and Staff Entity

A Pay Band can exist without Staff. There are multiple pay bands used by the company, but every instance of a Pay Band *does not* have to be used by Staff, so this is optional. For example, Pay Band C is £17000 per annum, but just because this exists does not mean a member of staff has to be on it, but then many staff could be on it, so this is an optional part of the relationship.

Staff do *have* to have a Pay Band though. For every instance of Staff, there has to be a Pay Band. So for every member of staff, they have to be on a pay band, but can only ever be on one pay band at any one time.

Staff Entity and ID Badge Entity

Staff *have* to have an ID badge, as without one they will not be allowed to enter the store or get service point access. I have decided that every instance of Staff has to have an

instance of ID Badge, so for every staff member input on the system, there has to be an ID badge number assigned, making this mandatory

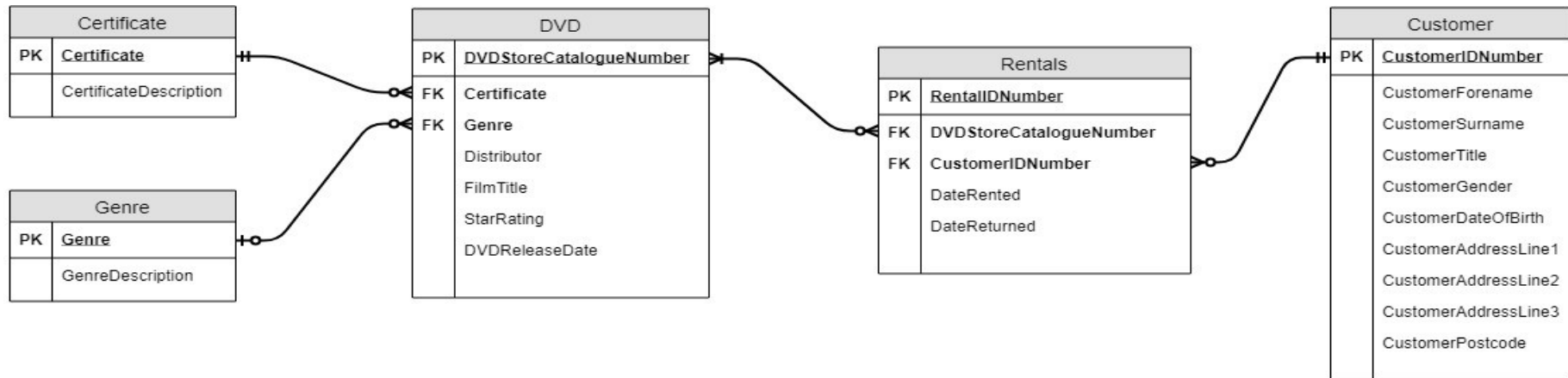
On the other side this means that for every instance of ID Badge, there *has* to be an instance of Staff, as an ID badge cannot be created with no one to assign it to. Only one instance of ID Badge can be assigned to one instance of Staff, so this is also a mandatory part of the relationship.

Appendix A

* = Foreign Key

ONF	1NF	2NF	3NF
Certificate CertificateDescription CustomerIDNumber CustomerForename CustomerSurname CustomerForename CustomerSurname CustomerTitle CustomerGender CustomerDateOfBirth CustomerAddressLine1 CustomerAddressLine2 CustomerAddressLine3 CustomerPostcode FilmTitle Genre GenreDescription Distributor DVDStoreCatalogueNumber StarRating DVDReleaseDate PayBand PayPerAnnum RentalIDNumber DateRented DateReturned StaffNINumber StaffForename StaffSurname TaxCode StaffAddressLine1 StaffAddressLine2 StaffAddressLine3 StaffPostcode JobTitle StaffDateOfBirth BadgeIDNumber BadgeDisplayName BadgeDisplayFavouriteFilm	CustomerIDNumber CustomerForename CustomerSurname CustomerTitle CustomerGender CustomerDateOfBirth CustomerAddressLine1 CustomerAddressLine2 CustomerAddressLine3 CustomerPostcode StaffNINumber StaffForename StaffSurname TaxCode StaffAddressLine1 StaffAddressLine2 StaffAddressLine3 StaffPostcode JobTitle StaffDateOfBirth BadgeIDNumber BadgeDisplayName PayBand PayPerAnnum DVDStoreCatalogueNumber Genre CertificateDescription Distributor FilmTitle StarRating DVDReleaseDate RentalIDNumber DateRented DateReturned Certificate Certificate Description BadgeDisplayFavouriteFilm	Certificate CertificateDescription CustomerIDNumber CustomerForename CustomerSurname CustomerTitle CustomerGender CustomerDateOfBirth CustomerAddressLine1 CustomerAddressLine2 CustomerAddressLine3 CustomerPostcode DVDStoreCatalogueNumber Distributor FilmTitle StarRating DVDReleaseDate Genre Genre description PayBand PayPerAnnum RentalIDNumber DateRented DateReturned StaffNINumber StaffForename StaffSurname TaxCode StaffAddressLine1 StaffAddressLine2 StaffAddressLine3 StaffPostcode JobTitle StaffDateOfBirth BadgeIDNumber BadgeDisplayName BadgeDisplayFavouriteFilm	Certificate CertificateDescription CustomerIDNumber CustomerForename CustomerSurname CustomerTitle CustomerGender CustomerDateOfBirth CustomerAddressLine1 CustomerAddressLine2 CustomerAddressLine3 CustomerPostcode DVDStoreCatalogueNumber *Certificate *Genre Distributor FilmTitle StarRating DVDReleaseDate Genre Genre description PayBand PayPerAnnum RentalIDNumber *DVDStoreCatalogueNumber *CustomerIDNumber DateRented DateReturned StaffNINumber *BadgeIDNumber *PayBand StaffForename StaffSurname TaxCode StaffAddressLine1 StaffAddressLine2 StaffAddressLine3 StaffPostcode JobTitle StaffDateOfBirth BadgeIDNumber BadgeDisplayName BadgeDisplayFavouriteFilm

Appendix B



Key:

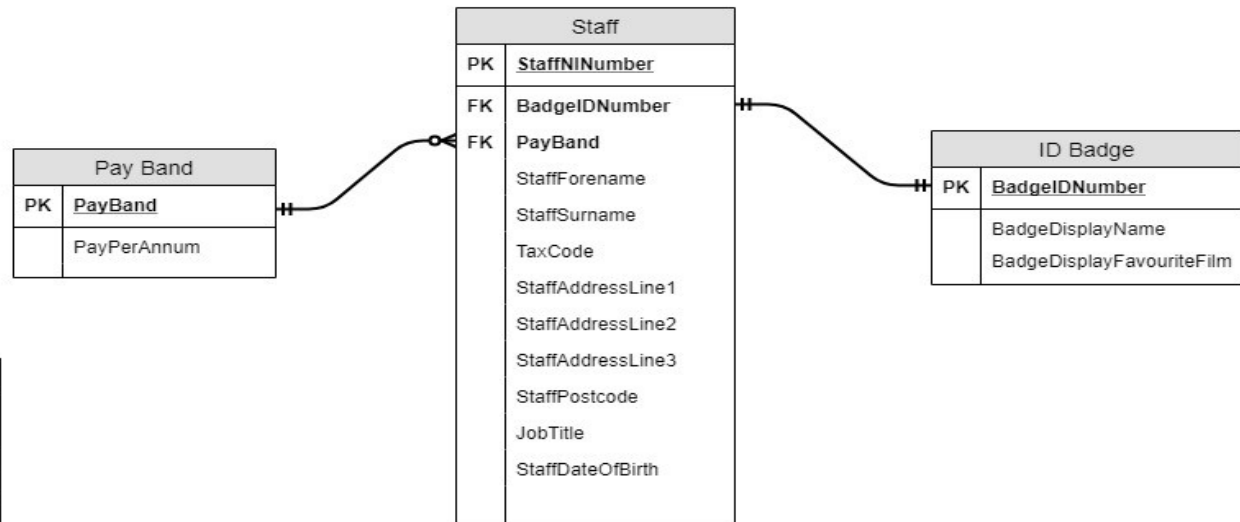
Zero or many
(optional)

One and only
one
(mandatory)

Zero or one
(optional)

One or more
(mandatory)

Entity Name	
PK	Primary Key
FK	Foreign Key
	Attribute
	Attribute



Appendix C

Database File	DVDStoreDatabase.accdb
----------------------	------------------------

Table Name	tblCertificate	Primary Key Field	Certificate
-------------------	----------------	--------------------------	-------------

Field Name	Data Type	Field Length	Input Mask/Validation Rule	Description	Typical Data
Certificate	Short Text	5	N/A	Ratings certificate of the film. Determines who can watch the film. Unique sequence of letter(s) and possibly numbers (hyphen attaches numbers to letters if numbers involved) for each rating certificate of a film	PG-13
CertificateDescription	Short Text	200	N/A	Text description of factors that make up a certificate	Parents Strongly Cautioned - Some material may be inappropriate for children under 13. Parents are urged to be cautious. Some material may be inappropriate for...

Table Name	tblCustomer	Primary Key Field	Cust ID Number
-------------------	-------------	--------------------------	----------------

Field Name	Data Type	Field Length	Input Mask/Validation Rule	Description	Typical Data
CustIDNumber	Number	N/A	>000000	Unique six-digit number for each customer	146468
CustSurname	Short Text	30	N/A	Surname of the customer	Smith
CustForename	Short Text	30	N/A	Forename of the customer	John
CustTitle	Short Text	10	N/A	Title of the customer	Mr
CustGender	Short Text	1	Combo Box, Value List "M", "F"	Gender of the customer. Either M or F	M
CustDateOfBirth	Date/Time	N/A	DD/MM/YYYY <#14/03/2001#	Customer's date of birth	08/12/1999
CustHouse/Flat No	Number	N/A	N/A	Customer's house or flat number	12
CustStreet/Road Name	Short Text	30	N/A	First line of customer's address	Withybrook Road
CustAddressLine2	Short Text	30	N/A	Second line of customer's address	Langley Park
CustTown/City	Short Text	30	N/A	Town or city customer lives in	Whitley Bay
CustCounty	Short Text	30	N/A	County customer lives in	County Durham
CustPostcode	Short Text	8	>LL00\ 0LL;0;_	Customer's address postcode	NE20 6TH

Table Name	tblDVD	Primary Key Field	Store Catalogue Number
-------------------	--------	--------------------------	------------------------

Field Name	Data Type	Field Length	Input Mask/Validation Rule	Description	Typical Data
StoreCatalogue Number	Number	N/A	Format 00000000	Unique number for each DVD in the store. Eight digits long. Leading zero at start if number does not contain eight digits	04896745
FilmTitle	Short Text	50	N/A	Name of the film	Despicable Me 2
Genre	Genre	15	Combo Box, Table/Query SELECT [tblGenre]	Genre of the film. Select from a list.	Horror
RunTime(mins)	Number	N/A	<=999	Number of minutes the film lasts for, cannot be longer than 999	120
Distributor	Short Text	30	N/A	The film's distributor	Universal
Certificate	Short Text	5	Combo Box, Table/Query SELECT [tblCertificate]	Ratings certificate of the film. Determines who can watch the film. Select from a list.	PG-13
StarRating(out of 5)	Short Text	5	Combo Box, Value List "*****", "*****", "*****", "*****", "*****"	Metacritic score of the film (five stars being the best)	***
SequelToA PreviousFilm?	Yes/No	N/A	N/A	Whether the film is a sequel to a previous film in a franchise	Yes
DVDReleaseDate	Date/Time	N/A	DD/MM/YYYY <#14/03/2018#	Date the DVD was released to the public	08/02/2013

Table Name	tblGenre	Primary Key Field	Genre
-------------------	----------	--------------------------	-------

Field Name	Data Type	Field Length	Input Mask/Validation Rule	Description	Typical Data
Genre	Short Text	15	N/A	Name of a genre that a film can be in	Horror
GenreDescription	Short Text	255	N/A	Text description of the characteristics of the genre	Movies in the horror genre involve blood, gore, the supernatural and things that go bump in the night. It includes ghost stories, alien invasions....

Table Name	tblPayBands	Primary Key Field	Pay Band
-------------------	-------------	--------------------------	----------

Field Name	Data Type	Field Length	Input Mask/Validation Rule	Description	Typical Data
PayBand	Short text	1	Like "[F-Z]"	Individual letters act as a symbol for the different amounts paid to staff per annum. One character long. Rule set so any new pay bands can only start from letter F upwards	A
PayPerAnnum	Currency	N/A	N/A	The amount in pounds paid per annum according to the pay band character. No decimal places	£26000

Table Name	tblRental	Primary Key Field	Rent ID Number
-------------------	-----------	--------------------------	----------------

Field Name	Data Type	Field Length	Input Mask/Validation Rule	Description	Typical Data
RentIDNumber	Short Text	7	Like "R*", >L0000000	Unique ID number for every rental that is put through the system. Letter R followed by six numbers	R123456
StoreCatalogue Number	Number	N/A	Format 00000000	Unique number for each DVD in the store. Eight digits long. Leading zero at start if number does not contain eight digits	04896745
CustIDNumber	Number	N/A	>000000	Unique six-digit number for each customer	146468
DateRented	Date/Time	N/A	DD/MM/YYYY	Date the DVD was rented from the store	08/01/2019
DateDue	Date/Time	N/A	DD/MM/YYYY	Date the DVD is due back at the store in order to avoid a fine	22/01/2019
DateReturned	Date/Time	N/A	DD/MM/YYYY	Date the DVD was returned to the store	24/01/2019
LateReturn?	Yes/No	N/A	=If("DateReturned">"DateDue",Yes)	If the film was returned at a date that was past the due date or not	Yes
FineAmount	Currency	N/A	£0.50*days="DateReturned">"DateDue"	The fine amount in pounds owed by the customer due to late return of the DVD	£2.00

Table Name	tblStaff	Primary Key Field	NI Number
-------------------	----------	--------------------------	-----------

Field Name	Data Type	Field Length	Input Mask/Validation Rule	Description	Typical Data
NINumber	Short Text	9	>LL000000L;0;_	National Insurance number of the employee. Two letters followed by six numbers then a letter	FE112233G
StaffSurname	Short Text	30	N/A	Surname of the employee	Jones
StaffForename	Short Text	30	N/A	Forename of the employee	John
PayBand	Short Text	1	Combo Box, Table/Query SELECT [tblPayBands]	Individual letters act as a symbol for the different amounts paid to staff per annum. One character long	A
TaxCode	Short Text	7	N/A	Tax code of the employee. Four or five numbers followed by a letter	1000L
StaffHouse/Flat No	Number	N/A	N/A	Employee's house number	12
StaffStreet/Road Name	Short Text	30	N/A	First line of employee's address	Woodlands Road
StaffAddressLine2	Short Text	30	N/A	Second line of employee's address	Hitchen Park
StaffTown/City	Short Text	30	N/A	Town or city employee lives in	Whitley Bay
StaffCounty	Short Text	30	N/A	County employee lives in	County Durham
StaffPostcode	Short Text	8	>LL00\ 0LL;0;_	Employee's address postcode	NE21 7TH

JobTitle	Short Text	30	Combo Box, Value List	Employee's job title at the store	Manager
EnrolledInWork Pension?	Yes/No	N/A	N/A	Whether the employee has enrolled in the company pension scheme or not	No
StaffDateOfBirth	Date/Time	N/A	DD/MM/YYYY <#14/03/2001#	Employee's date of birth	25/05/1989
StaffGender	Short Text	1	Combo Box, Value List "M", "F"	Gender of the employee. Either M or F	F
StaffTitle	Short Text	10	N/A	Title of the employee	Miss
StaffIDNumber	Number	N/A	>0000	Individual four digit number for each member of staff	1080

Table Name	tblStaffIDBadge	Primary Key Field	Staff ID Number
-------------------	-----------------	--------------------------	-----------------

Field Name	Data Type	Field Length	Input Mask/Validation Rule	Description	Typical Data
StaffIDNumber	Number	N/A	>0000	Individual four digit number for each member of staff	1080
BadgeDisplay Name	Short Text	30	N/A	Name displayed on a badge. First name only	Phil
BadgeDisplayFavouriteFilm	Short Text	50	N/A	An employee's favourite film	Scream