TEMASEK JUNIOR COLLEGE

Temasek Junior College

2023 JC2 H2 Computing

Database 1 - Introduction to Databases

Syllabus Objectives

After completing this set of notes, you should be able to:

- Determine the attributes of a database: table, record and field.
- Explain the purpose of and use primary, secondary, composite and foreign keys in tables.
- Explain with examples, the concept of data redundancy and data dependency.
- Describe the characteristics of data in unnormalised form (0NF), first normal form (1NF), second normal form (2NF) and third normal form (3NF).
- Reduce data redundancy to third normal form (3NF).
- Discuss the advantages and disadvantages of normalisation.

1 What is a Database?

Data are facts associated with an object or entity in consideration.

A **database** is a collection of data stored in an organised or logical manner. Storing data in a database allows us to access and manage the data. Examples of databases in real-life include:

- Patient medical records
- Supermarket inventory
- Contact list

2 Relational Database

In the relational database model, data is stored in relational tables and represented in the form of tuples (rows). A relational database is a collection of relational tables.

In relational database, data is stored in separate tables and each table stores data about a single entity.

2.1 Tables

A table is a two-dimensional representation of data stored in rows and columns. Below is an example of a table `Student18S12`, showing data for students from civics class `18S12`.

Field

RegNo	Name	Gender	MobileNo
1	Adam	M	92313291
2	Adrian	M	92585955
3	Agnes	F	83324112
4	Aisha	F	88851896
5	Ajay	M	94191061
6	Alex	M	98675171
7	Alice	F	95029176
8	Amy	F	98640883
9	Andrew	M	95172444
10	Andy	M	95888639

Record

A table is a relational table if it fulfils the following conditions:

- Values are atomic
- Columns are of the same kind
- Rows are unique
- The order of columns is insignificant
- Each column must have a unique name

2.1.1 Records and Fields

From the example of `Student18S12`, we can see that tables are made up of **records** and **fields**.

A **record** is a complete set of data about a single item. A record in the table `Student18S12` refers to the complete set of data of a particular student.

A **field** refers to one piece of data about a single item. The table `Student18S12` has 4 fields – `RegNo`, `Name`, `Gender` and `MobileNo`.

Question
How many records does the table `Student18S12` have?

Answer
10 records

2.1.2 Table Descriptions

A table description can be expressed as:

```
TableName(Attribute1, Attribute2, Attribute3, ...)
```

For example, the table description for table `Student18S12` is:

```
Student18S12(RegNo, Name, Gender, MobileNo)
```

2.1.3 Keys

Since each row of a relational table is unique, they can be identified using a **key**, which is a field (or combination of fields) that uniquely identifies the row.

Candidate keys

A **candidate key** is defined as a minimal set of fields which can uniquely identify each record in a table. It tells a particular record apart from another record.

A candidate key should never be NULL or empty. There can be more than one candidate key for a relational table. A candidate key can also be a combination of more than one field.

Question

Which of the fields in table `Student18S12` are candidate keys?

Answer

`RegNo`
`Name`
`MobileNo`

Primary key

A **primary key** is a candidate key that is most appropriate to become the main key for a relational table. It uniquely identifies each record in a table and should not change over time. That is, a primary key tells a particular record apart from another record.

Question

Which of the candidate key is a suitable primary key for the table `Student18S12`?

Answer

`RegNo`

Secondary keys

Candidate keys that are not selected as primary key are known as **secondary keys**.

Question

Which of the candidate keys are secondary keys for the table `Student18S12`?

Answer

`Name`
`MobileNo`

Composite key

Sometimes, more than one field is needed to uniquely identify a record. A **composite key** is a combination of two or more fields in a table that can be used to uniquely identify each record in a table. Uniqueness is only guaranteed when the fields are combined. When taken individually, the fields do not guarantee uniqueness.

Take a look at the table `Student` below.

RegNo	Name	Gender	CivicsClass
1	Adam	M	18S12
2	Adrian	M	18S12
3	Agnes	F	18S12
4	Aisha	F	18S12
5	Ajay	M	18S12
6	Alex	M	18S12
7	Alice	F	18S12
8	Amy	F	18S12
9	Andrew	M	18S12
10	Andy	M	18S12
1	Adam	M	18A10
2	Bala	M	18A10
3	Bee Lay	F	18A10
4	Ben	M	18A10
5	Boon Kiat	M	18A10
6	Boon Lim	M	18A10
7	Charles	M	18A10
8	Chee Seng	M	18A10
9	Cher Leng	F	18A10
10	Choo Tuan	M	18A10

CivicsClass `18S12`

CivicsClass `18A10`

Now, a single field is not able to uniquely identify a record. A composite key composing 2 fields is needed to uniquely identify a record.

Question

Which 2 fields would you choose to form a composite key for the above table `Student`?

Answer

`RegNo, CivicsClass`

Foreign keys

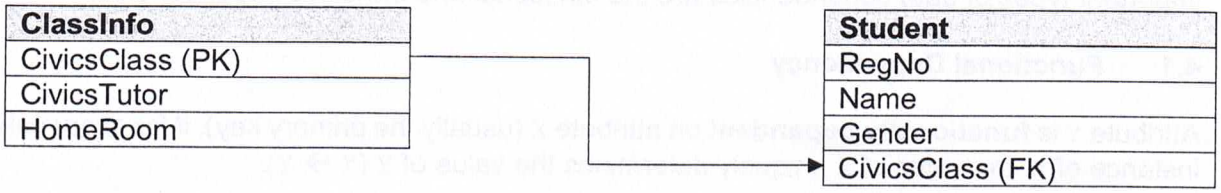
A **foreign key** is an attribute (field) in one table that refers to the primary key in another table.

Another table, `ClassInfo`, as shown below, stores the information about each civics class.

CivicsClass	CivicsTutor	HomeRoom
18S12	Peter Lim	BlkA-L2-3
18A10	Pauline Lee	BlkC-L1-4

`CivicsClass` is chosen to be the primary key (PK) for table `ClassInfo`.

Notice that the primary key in table `ClassInfo` (i.e., the `CivicsClass` field) is related or linked to the `CivicsClass` field in table `Student`. This makes `CivicsClass` field in table `Student` a foreign key (FK).



3 Data Redundancy

Data redundancy refers to the same data being stored more than once.

In a relational database, data redundancy should not occur.

In the table below, data for `CivicsClass` and `CivicsTutor` are repeated for students who are in the same civics class. This is data redundancy.

RegNo	Name	Gender	CivicsClass	CivicsTutor
1	Adam	M	18S12	Peter Lim
2	Adrian	M	18S12	Peter Lim
3	Agnes	F	18S12	Peter Lim
4	Aisha	F	18S12	Peter Lim
5	Ajay	M	18S12	Peter Lim
6	Alex	M	18S12	Peter Lim
7	Alice	F	18S12	Peter Lim
8	Amy	F	18S12	Peter Lim
9	Andrew	M	18S12	Peter Lim
10	Andy	M	18S12	Peter Lim

3.1 Issues with Data redundancy

Data redundancy can cause issues when inserting, updating and deleting data from the database, leading to data inconsistency.

Using the table above as an example, we can expect the following issues:

Inserting data	A new student cannot be inserted unless a civics class and a civics tutor have been assigned.
Updating data	If Peter Lim decides to leave the college, all the records in the table would need to be updated. If any record is missed, it will lead to inconsistent data.
Deleting data	If all the records of the table are deleted, information on civics class and civics tutor will be lost.

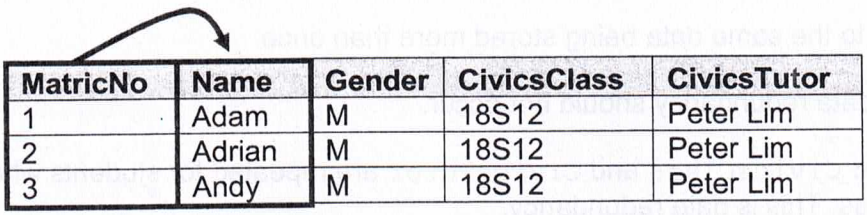
4 Data Dependency

Data redundancy is usually a result of data dependencies between different fields. Two important types of data dependencies are the functional and transitive dependencies.

4.1 Functional Dependency

Attribute `Y` is **functionally dependent** on attribute `X` (usually the primary key), if for every valid instance of `X`, the value of `X` uniquely determines the value of `Y` (`X` → `Y`).

Suppose we have a table with the following attributes: `MatricNo`, `Name`, `Gender`, `CivicsClass` and `CivicsTutor`.



MatricNo	Name	Gender	CivicsClass	CivicsTutor
1	Adam	M	18S12	Peter Lim
2	Adrian	M	18S12	Peter Lim
3	Andy	M	18S12	Peter Lim

`MatricNo` is a unique number assigned to every student in the college. It uniquely identifies the `Name` because if we know the `MatricNo`, we can know the `Name` associated with it. Therefore, we can say `Name` is functionally dependent on `MatricNo` (`MatricNo` → `Name`).

4.2 Transitive Dependency

A functional dependency is **transitive** if it is indirectly formed by two functional dependencies.

`Z` is transitively dependent on `X` if

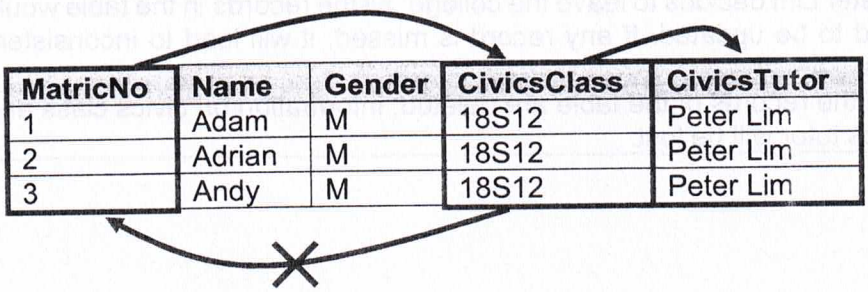
- `Y` is functionally dependent on `X` but `X` is not functionally dependent on `Y`,
- `Z` is functionally dependent on `Y`.

In other words, `X` → `Z` is a transitive dependency if the following hold true:

- `X` → `Y` (e.g. `X` is primary key, determines which `Y` is selected)
- `Y` does not → `X` (`Y` cannot be used to determine what `X` is)
- `Y` → `Z` (`Z` will depend on what `Y` is)

For example:

- `CivicsClass` is functionally dependent on `MatricNo` (`MatricNo` → `CivicsClass`)
- `MatricNo` is however not functionally dependent on `CivicsClass`.
- `CivicsTutor` is functionally dependent on `CivicsClass` (`CivicsClass` → `CivicsTutor`).



MatricNo	Name	Gender	CivicsClass	CivicsTutor
1	Adam	M	18S12	Peter Lim
2	Adrian	M	18S12	Peter Lim
3	Andy	M	18S12	Peter Lim

`CivicsTutor` is thus transitively dependent on `MatricNo` (`MatricNo` → `CivicsTutor`).

5 Normalisation

- The objectives of database normalisation are to make a database more efficient and useful. It centres around reducing redundant data and ensuring data dependencies; in other words, the data in each table is all properly and completely related together.
- When data is duplicated in a database it is known as data redundancy.
- Data redundancy can cause a problem as it can lead to inconsistencies in the data.
- This would reduce the integrity of the data.
- The purpose of normalisation is to reduce the amount of duplicated data and to simplify the structure of a database.

Example
A database stores data about people who buy books.

Each time a person buys a book, their name and address are entered into the database to record the purchase. This means that each time the person buys a book their data will be duplicated in the database.

Over time, the customers' data might be entered into the database many times. This can be very wasteful of storage space and it can also lead to inconsistencies, as the customer data may be entered slightly different each time.

Also, if the person moves to a different address and buys another book, the same name will now be entered, but with a different address. The database now has inconsistent data, as the address fields will not match.

- A database that allows duplicated records of data is called an unnormalised database.
- This database can then be normalised to remove the redundant data.
- Normalisation is the process of organising the tables in a database to reduce data redundancy and prevent inconsistent data. There are at least three normal forms associated with normalisation: first normal form (1NF), second normal form (2NF), and third normal form (3NF).

5.1 Stages of Normalisation

5.1.1 Unormalised Form (0NF or UNF)

- A database that is unnormalised has repeated records of data.
- It may also have repeating fields.
- It does not have a primary key.
- An unnormalised database may also have data that is not atomic. This means that each field in the database could contain more than one piece of information.
    - E.g. the field `'Customer Name'` in the database could contain the full name of the customer – both their first name and last name – in this one field.

5.1.2 First Normal Form (1NF)

- A database that is in 1NF does not have any repeating fields within each record.
- All the fields in the database are atomic.
- Each record in the database is unique.
- Each record has a unique identifier that is called a primary key.

5.1.3 Second Normal Form (2NF)

- A database that is in 2NF has all of the characteristics of a database in 1NF i.e. it must be 1NF.
- In addition, any data that is not functionally dependent on the primary key or a part of a composite primary key will be separated from the primary key or that part of a composite primary key.
- This results in the creation of multiple tables that each have its own primary key.
- The fields in each table are only related to the primary key of that table.

5.1.4 Third Normal Form (3NF)

- A database that is in 3NF has all of the characteristics of a database in 2NF i.e. it must be 2NF.
- In addition, any fields that are not directly related to the primary key in a table, but only transitively dependent, are further separated into other tables.
- These tables contain only fields that are directly related, along with their primary key.
- To link these tables together to create relationships, a primary key may appear as a 'foreign key' in a linked table.

5.2 Worked Example 1 - CCA

5.2.1 First Normal Form (1NF)

For a table to be in 1NF, all columns must be atomic. This means there can be no multi-valued columns – i.e., columns that would hold a collection such as an array or another table.

In another words, information in each column of a 1NF form cannot be broken down further.

Consider the following table:

MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom	CCAInfo
1	Adam	M	18S12	Peter Lim	TR1	Tennis Teacher IC = Adrian Tan
2	Adrian	M	18S12	Peter Lim	TR1	Choir Teacher IC = Adeline Wong, SC Teacher IC = David Leong
3	Adam	M	18A10	Pauline Lee	TR2	Rugby Teacher IC = Andrew Quah
4	Bala	M	18A10	Pauline Lee	TR2	Badminton Teacher IC = Lilian Lim
5	Bee Lay	F	18A10	Pauline	TR2	Choir Teacher IC = Adeline Wong, Chess Club Teacher IC = Edison Poh

It is assumed that every civics class only has one civics tutor, and each CCA has only one teacher in-charge (IC).

This table is not in 1NF because the `CCAInfo` column contains multiple values.

In order for the table to be in 1NF, we split `CCAInfo` into two single-value columns `CCAName` and `CCATeacherIC`.

Notice that the students with `MatricNo 2` and `MatricNo 5` have multiple CCAs. We keep this information intact by splitting their records into multiple records, each corresponding to a different CCA. The resulting table is shown below.

MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom	CCAName	CCATeacherIC
1	Adam	M	18S12	Peter Lim	TR1	Tennis	Adrian Tan
2	Adrian	M	18S12	Peter Lim	TR1	Choir	Adeline Wong
2	Adrian	M	18S12	Peter Lim	TR1	SC	David Leong
3	Adam	M	18A10	Pauline Lee	TR2	Rugby	Andrew Quah
4	Bala	M	18A10	Pauline Lee	TR2	Badminton	Lilian Lim
5	Bee Lay	F	18A10	Pauline	TR2	Choir	Adeline Wong
5	Bee Lay	F	18A10	Pauline	TR2	Chess Club	Edison Poh

The values for `CCAName` and `CCATeacherIC` are now atomic for each row. The primary key for the above table will be the composite key formed by `MatricNo` and `CCAName`.

Are you able to reason out why the composite key is as such?

5.2.2 Second Normal Form

For a table to be in 2NF, it must satisfy two conditions:

- The table should be in 1NF.
- Every non-key attribute must be fully dependent on the entire primary key. This means no attribute can depend on part of the primary key only.

`Name`, `Gender`, `CivicsClass`, `CivicsTutor` and `HomeRoom` is dependent on only part of the primary key, `MatricNo`. A `Student` table can be formed using these columns.

`CCATeacherIC` is dependent only on another part of the primary key, `CCAName`. A `CCAInfo` table can be formed using these columns.

MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom
1	Adam	M	18S12	Peter Lim	TR1
2	Adrian	M	18S12	Peter Lim	TR1
2	Adrian	M	18S12	Peter Lim	TR1
3	Adam	M	18A10	Pauline Lee	TR2
4	Bala	M	18A10	Pauline Lee	TR2
5	Bee Lay	F	18A10	Pauline	TR2
5	Bee Lay	F	18A10	Pauline	TR2

CCAName	CCATeacherIC
Tennis	Adrian Tan
Choir	Adeline Wong
SC	David Leong
Rugby	Andrew Quah
Badminton	Lilian Lim
Choir	Adeline Wong
Chess Club	Edison Poh

A third table `StudentCCA` containing the columns `MatricNo` and `CCAName` can be formed, so that data in the `Student` and `CCAInfo` tables can be related.

We can thus decompose the 1NF table into a 2NF form with three tables as shown below.

Student

MatricNo	Name	Gender	CivicsClass	CivicsTutor	Home Room
1	Adam	M	18S12	Peter Lim	TR1
2	Adrian	M	18S12	Peter Lim	TR1
3	Adam	M	18A10	Pauline Lee	TR2
4	Bala	M	18A10	Pauline Lee	TR2
5	Bee Lay	F	18A10	Pauline Lee	TR2

StudentCCA

MatricNo	CCAName
1	Tennis
2	Choir
2	SC
3	Rugby
4	Badminton
5	Choir
5	Chess Club

CCAInfo

CCAName	CCATeacherIC
Tennis	Adrian Tan
Choir	Adeline Wong
SC	David Leong
Rugby	Andrew Quah
Badminton	Lilian Lim
Chess Club	Edison Poh

The primary key for table `Student` will be `MatricNo`.

`CCAName` will be the primary key for table `CCAInfo`.

`MatricNo` and `CCAName` will form a composite key for table `StudentCCA`.

5.2.3 Third Normal Form

For a table to be in 3NF, it must satisfy two conditions:

- The table should be in 2NF.
- The table should not have transitive dependencies.

Looking at the table `Student`, `CivicsTutor` and `HomeRoom` are dependent on `CivicsClass` and `CivicsClass` is dependent on `MatricNo`. Therefore, `CivicsTutor` and `HomeRoom` are transitively dependent on `MatricNo`.

Student

MatricNo	Name	Gender	CivicsClass	CivicsTutor	HomeRoom
1	Adam	M	18S12	Peter Lim	TR1
2	Adrian	M	18S12	Peter Lim	TR1
3	Adam	M	18A10	Pauline Lee	TR2
4	Bala	M	18A10	Pauline Lee	TR2
5	Bee Lay	F	18A10	Pauline Lee	TR2

To remove the transitive dependency, we can further decompose the `Student` table as follows:

Student

MatricNo	Name	Gender	CivicsClass
1	Adam	M	18S12
2	Adrian	M	18S12
3	Adam	M	18A10
4	Bala	M	18A10
5	Bee Lay	F	18A10

Civics

CivicsClass	CivicsTutor	HomeRoom
18S12	Peter Lim	TR1
18S12	Peter Lim	TR1
18A10	Pauline Lee	TR2
18A10	Pauline Lee	TR2
18A10	Pauline Lee	TR2

`MatricNo` remains the primary key for table `Student`. `CivicsClass` will be the primary key of the newly formed table `Civics`.

The final design after normalisation is as shown:

Student

MatricNo	Name	Gender	CivicsClass
1	Adam	M	18S12
2	Adrian	M	18S12
3	Adam	M	18A10
4	Bala	M	18A10
5	Bee Lay	F	18A10

Civics

CivicsClass	CivicsTutor	HomeRoom
18S12	Peter Lim	TR1
18A10	Pauline Lee	TR2

StudentCCA

MatricNo	CCAName
1	Tennis
2	Choir
2	SC
3	Rugby
4	Badminton
5	Choir
5	Chess Club

CCAInfo

CCAName	CCATeacherIC
Tennis	Adrian Tan
Choir	Adeline Wong
SC	David Leong
Rugby	Andrew Quah
Badminton	Lilian Lim
Chess Club	Edison Poh

The design can be represented using the following set of table descriptions:

`Student ( MatricNo , Name , Gender , CivicsClass )`

`Civics ( CivicsClass , CivicsTutor , HomeRoom )`

`StudentCCA ( MatricNo , CCAName )`

`CCAInfo ( CCAName , CCATeacherIC )`

The primary key for each table is indicated by underlining one or more attributes. Foreign keys are indicated by using a dashed underline.

5.3 Worked Example 2 - Concert

This section works through the stages in taking a database from unnormalised form to third normal form.

An entertainment company books music bands and artists for concert dates in the UK. The company handles the bookings for multiple bands at multiple venues and it communicates with multiple agents to do this.

5.3.1 Unormalised Form (0NF or UNF)

An example of a database for this company in unnormalised (0NF) form is shown below:

ArtistID	ArtistName	VenueID	ConcertVenue	ConcertDate	AgentName	AgentEmail
RO01	Rock On	O21	O2 Academy - Birmingham	01/06/2016	Alice Jones	ajones@welovemusic.com
SF01	Stage Fright	HA1	Hammersmith - London	01/06/2016	Bill Richards	billyrich@amazingartists.com
JS01	John Smith	O22	O2 Academy - Leeds	10/06/2016	Henry Cooper	henry.cooper@me.com
RO01	Rock On	BC1	Barbican Centre - London	06/06/2016	Alice Jones	ajones@welovemusic.com
RO01	Rock On	O23	O2 Academy - Liverpool	10/06/2016	Alice Jones	ajones@welovemusic.com
SF01	Stage Fright	RA1	Royal Albert Hall - London	20/06/2016	Bill Richards	billyrich@amazingartists.com
JY01	Jimmy	RA1	Royal Albert Hall - London	28/06/2016	Henry Cooper	henry.cooper@me.com
TD01	The Delights	O21	O2 Academy - Birmingham	13/06/2016	Alice Jones	ajones@welovemusic.com
RO01	Rock On	HA1	Hammersmith - London	13/06/2016	Alice Jones	ajones@welovemusic.com
JY01	Jimmy	BC1	Barbican Centre - London	30/06/2016	Henry Cooper	henry.cooper@me.com

We can see from this database that there are multiple records for each artist. This means that there is more than one record in the database for an artist.

The field `'Concert Venue'` is not atomic – that is, it has more than one item of data in the field. This means that the data in that field could be broken down further. For example, the concert venue and the location are both in the same field `'Concert Venue'`. This could be broken down into two fields.

Some may consider that `'Agent Name'` is not atomic as well, as it can be said that the agent's first name and last name are both in the same field `'Agent Name'`. This could be broken down into two fields. For this example, we will separate the first name and last name into two fields for this example, though it is not necessary.

Another feature of this database is that each record in the database does not have a unique identifier.

5.3.2 First Normal Form (1NF)

We can convert this unnormalised database into 1NF by making sure that each field is atomic (only contains one item of data), and by giving each record a unique identifier.

We do not need to remove any repeating fields in each record as none occur in this database. There are repeated records but these are removed in a later stage of normalisation.

In 1NF our database looks like this:

ArtistID	ArtistName	VenueID	ConcertVenue	Location	ConcertDate	AgentFirstName	AgentLastName	AgentEmail
RO01	Rock On	O21	O2 Academy	Birmingham	01/06/2016	Alice	Jones	ajones@welovemusic.com
SF01	Stage Fright	HA1	Hammersmith	London	01/06/2016	Bill	Richards	billyrich@amazingartists.com
JS01	John Smith	O22	O2 Academy	Leeds	10/06/2016	Henry	Cooper	henry.cooper@me.com
RO01	Rock On	BC1	Barbican Centre	London	06/06/2016	Alice	Jones	ajones@welovemusic.com
RO01	Rock On	O23	O2 Academy	Liverpool	10/06/2016	Alice	Jones	ajones@welovemusic.com
SF01	Stage Fright	RA1	Royal Albert Hall	London	20/06/2016	Bill	Richards	billyrich@amazingartists.com
JY01	Jimmy	RA1	Royal Albert Hall	London	28/06/2016	Henry	Cooper	henry.cooper@me.com
TD01	The Delights	O21	O2 Academy	Birmingham	13/06/2016	Alice	Jones	ajones@welovemusic.com
RO01	Rock On	HA1	Hammersmith	London	13/06/2016	Alice	Jones	ajones@welovemusic.com
JY01	Jimmy	BC1	Barbican Centre	London	30/06/2016	Henry	Cooper	henry.cooper@me.com

We have now made all the fields in the database atomic. We have done this by breaking down the venue name and location into two separate fields, and by breaking down the agent first and last name into two separate fields.

There is no field in the database that is unique for each record. To create a unique identifier for each record, we combine three fields together `ArtistID`, `VenueID` and `ConcertDate` to create a composite key. This composite key is a unique identifier for each record as only one artist will play at a particular venue on a given date.

The characteristics of 1NF are therefore:

- all fields in the database are atomic
- all repeated fields within each record have been removed
- a unique identifier has been created for each record.

Note, repeated fields within each record is not the same as repeated records.

5.3.3 Second Normal Form (2NF)

Now that we have taken our database to 1NF we can take it to 2NF. In 2NF our database looks like this:

ArtistDetails

ArtistID	ArtistName	AgentFirstName	AgentLastName	AgentEmail
RO01	Rock On	Alice	Jones	ajones@welovemusic.com
SF01	Stage Fright	Bill	Richards	billyrich@amazingartists.com
JS01	John Smith	Henry	Cooper	henry.cooper@me.com
JY01	Jimmy	Henry	Cooper	henry.cooper@me.com
TD01	The Delights	Alice	Jones	ajones@welovemusic.com

VenueDetails

VenueID	ConcertVenue	Location
O21	O2 Academy	Birmingham
HA1	Hammersmith	London
O22	O2 Academy	Leeds
BC1	Barbican Centre	London
O23	O2 Academy	Liverpool
RA1	Royal Albert Hall	London

ArtistsBookings

ConcertID	ArtistID	VenueID	ConcertDate
CON001	RO01	O21	01/06/2016
CON002	SF01	HA1	01/06/2016
CON003	JS01	O22	10/06/2016
CON004	RO01	BC1	06/06/2016
CON005	RO01	O23	10/06/2016
CON006	SF01	RA1	20/06/2016
CON007	JY01	RA1	28/06/2016
CON008	TD01	O21	13/06/2016
CON009	RO01	HA1	13/06/2016
CON010	JY01	BC1	30/06/2016

We have now separated our database into different tables.

We have introduced a primary key to each database:
- `ArtistID` for **ArtistDetails** table
- `VenueID` for **VenueDetails** table
- `ConcertID` for **ArtistBookings** table

The primary keys act as unique identifiers for each record in their respective tables. This means that data entered into that field for each record must be different for every record.

The data in each table is dependent on the primary key of that table:
- the **ArtistDetails** table contains details about each artist
- the **VenueDetails** table contains details about each venue
- the **ArtistsBookings** table contains details about each concert booking for each artist. An additional field `ConcertID` has been added to this table to create a primary key.

The fields that are in each of these tables are directly related to the primary key of that table.

Observe that the data is related in the following manner:
- the **ArtistDetails** table is related to the **ArtistBookings** table via `ArtistID`.
- the **VenueDetails** table is related to the **ArtistBookings** table via `VenueID`.

The characteristics of 2NF are therefore:
- the database has the characteristics of 1NF
- related data has been separated into different tables
- each table has a primary key
- the fields in each table are dependent on the primary key of that table.

5.3.4 Third Normal Form (3NF)

Now we have taken our database to 2NF we can take it to 3NF. In 3NF our database looks like this:

ArtistDetails

Artist ID*	Artist Name	Agent ID
RO01	Rock On	AGAJ01
SF01	Stage Fright	AGBR01
JS01	John Smith	AGHC01
JY01	Jimmy	AGHC01
TD01	The Delights	AGAJ01

AgentDetails

Agent ID*	Agent First Name	Agent Last Name	Agent Email
AGAJ01	Alice	Jones	ajones@welovemusic.com
AGBR01	Bill	Richards	billyrich@amazingartists.com
AGHC01	Henry	Cooper	henry.cooper@me.com

VenueDetails

Venue ID*	Concert Venue	Location of Venue
O21	O_2 Academy	Birmingham
HA1	Hammersmith Apollo	London
O22	O_2 Academy	Leeds
BC1	Barbican Centre	London
O23	O_2 Academy	Liverpool
RA1	The Royal Albert Hall	London

ArtistBookings

Concert ID*	Artist ID	Venue ID	Concert Date
CON001	RO01	O21	01/06/2016
CON002	SF01	HA1	01/06/2016
CON003	JS01	O22	10/06/2016
CON004	RO01	BC1	06/06/2016
CON005	RO01	O23	10/06/2016
CON006	SF01	RA1	20/06/2016
CON007	JY01	RA1	28/06/2016
CON008	TD01	O21	13/06/2016
CON009	RO01	HA1	13/06/2016
CON010	JY01	BC1	30/06/2016

We have now separated any fields in the table that are not directly related to each other.

In the **ArtistDetails** table, the fields relating to the artist were not directly related to the agent's details. We have therefore separated the agent details into another table.

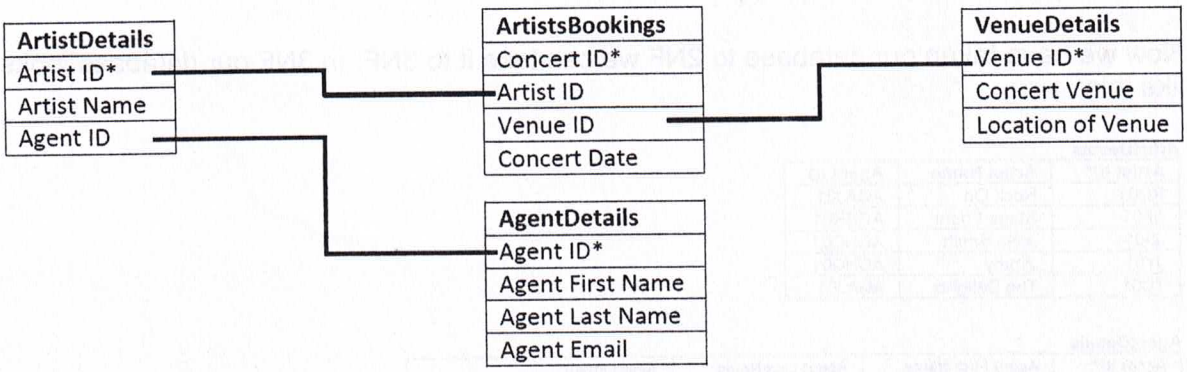
This table has also been given a primary key that is referenced in the artist's table.

The characteristics of 3NF are therefore:

- the database has the characteristics of 2NF
- fields have been further separated into tables that contain only fields that are directly related to each other.

By normalising our database we have removed any repeating fields and any repeating records, and have separated any data that is not directly related to the primary key of the table or the other fields within that table.

We have taken our database through the three different stages of normalisation and can now create links between the tables to create a relational database:



Our database now allows bookings to be made for each artist at a concert venue. The bookings could be made using the **ArtistsBookings** table.

The details for each artist and each venue are linked to the relevant table for those details. This is done through the use of their primary key appearing in this table as a foreign key.

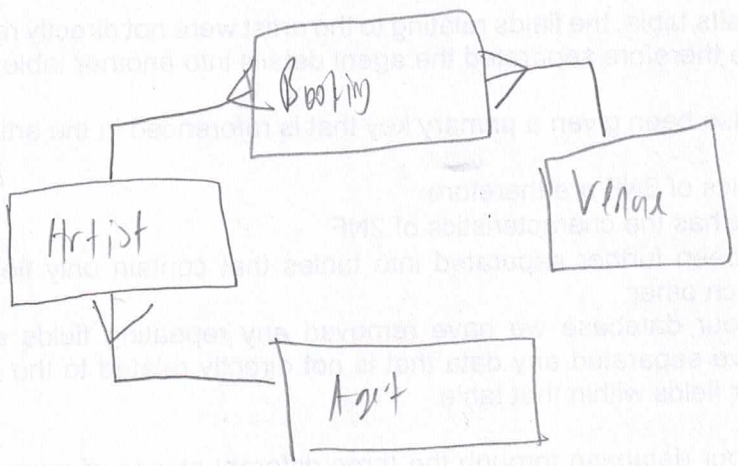
A standard notation is used to represent the data structures of table in normalisation.

The name of the table appears in capital letters and the field names that relate to it appear in brackets alongside, each one separated by a comma.

The primary key of the table is underlined in full and any foreign keys are underlined in dash.

In this notation our data structure would be:

ARTISTDETAILS(Artist ID, Artist Name, Agent ID)
ARTISTSBOOKINGS(Concert ID, Artist ID, Venue ID, Concert Date)
VENUEDETAILS(Venue ID, Concert Venue, Location of Venue)
AGENTDETAILS(Agent ID, Agent First Name, Agent Last Name, Agent Email)



5.4 Advantages and Disadvantages of Normalisation

Advantages	Disadvantages
The storage capacity needed for the database may be smaller as there is no repeating data.	The process of normalising a database can be very complex and is a specialist skill.
The database is easy to search as there is less data to search through, due to the removal of repeated records.	The process of searching the database may be slower. This is because a normalised database requires greater usage of a device's central processing unit (CPU), due to the links between tables.
If the data is amended in a record this will be amended in all matching records in the database, due to the links between the tables.	If data is amended in a normalised database then all matching records are amended. This means that the previous data used is deleted and can no longer be seen in any records.
The integrity of the data in the database is increased as there are no repeated records of data that could cause it to be inconsistent.	

Hire(Hire_ID, Hire_Date, Hire_Cost, Sales Man_ ID, Cust_ID)
Car (Car_Reg, Car_Make, Car_Model)
HireCar(Hire ID, Car_Reg)
Salesman(Salesman_ID, Salesm_First Name)
Customer(Customer ID, Cust_First Name, Cust_Last Name, Cust_Email)

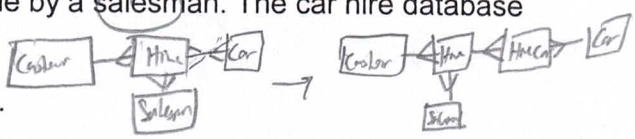
Practice Questions

1. A car hire firm wants to store data about its cars and the customers that hire them.

   When a customer hires a car, the hire is given a Hire_ID.

   A customer can hire more than one car under the same Hire_ID. Each car can only be hired to one customer at a time. Each hire is made by a salesman. The car hire database is currently in 1NF.

   Convert it to 3NF and write the table descriptions.

   Use the source file `Hire_Cars.csv`.

2. A veterinary surgery treats a variety of pets. The surgery has several vets. Each pet has one owner, but an owner may have several pets. When a pet needs treatment, its owner makes an appointment. Each appointment has a time and date, a vet, a pet and an owner.

   The unnormalised data for the surgery's database is as follows:

   ```
   APPOINTMENTS(Appointment_ID, Appointment_Time, Appointment_Date,
   Vet_Name,Owner_ID, Owner_Name, Owner_Email, Pet_Name, Type_of_Pet)
   ```

   Take this unnormalised data through to 3NF.

   Use the source file `Appointments.csv`.

Appt (Appt_ID, Appt_Time, Appt_Pet, Vet_Name, Owner_ID, Pet_ID)
Owner (Owner_ID, Owner_Name, Owner_Email)
Pet (Pet ID, Owner_ID, Pet_Name, Type_Of_Pet)