Chapter 8. Data Normalisation

Table of contents

Objectives

At the end of this chapter you should be able to:

- Describe the process, strengths and weaknesses of data normalisation, and demonstrate an understanding of when and to what extent the technique should be applied in practice.
- Explain and apply the concepts of functional dependency and determinants through the understanding and construction of determinacy diagrams.

- Describe and apply understanding of three normal forms for relations.
- Convert un-normalised data into first normal form relations, so that data items contain only single, simple values.
- Derive second normal form relations by eliminating part-key dependencies.
- Derive third normal form relations by removing transitive dependencies.

Introduction

In parallel with this chapter, you should read Chapter 13 of Thomas Connolly and Carolyn Begg, "Database Systems A Practical Approach to Design, Implementation, and Management", (5th edn.).

Normalisation stands on its own as a well-founded approach to database design. In addition, normalisation links closely with the material covered in the previous two chapters on entity-relationship modelling. However, the additional flexibility of normalised designs comes at a price — a well-normalised design tends to perform poorly when subjected to large volumes of transactions. For this reason, there are trade-offs to be made between the extent to which a design is normalised and the performance response of the implemented system. The information in this chapter has to be applied carefully, in light of the information given in a later chapter on database design relating to de-normalisation and physical design.

Why should we attempt to normalise data? Un-normalised data often contains undesirable redundancy (and its associated 'costs' in storage, time and multiple updates), and different degrees of normalisation (i.e. different normal forms) can guarantee that certain creation, update and deletion anomalies can be avoided.

Context

This chapter covers the well-known approach to database design known as data normalisation. It introduces a bottom-up technique for the development of flexible database applications. This bottom-up approach complements the top-down entity-relationship technique presented in the first database design chapter, as the two approaches can be used to cross-check the extent to which the overall design satisfies the requirements of the application. By themselves, database designs arrived at through the normalisation process, while providing great flexibility, tend to perform very slowly. The complementary bottom-up and top-down methodologies, in practice, often reveal different information, and can be applied using different fact-finding techniques. For these reasons (of efficiency and the benefits of multiple viewpoints to get a better final design), a balanced approach to database design will use both approaches.

Determinacy diagrams

Determinants and determinacy diagrams

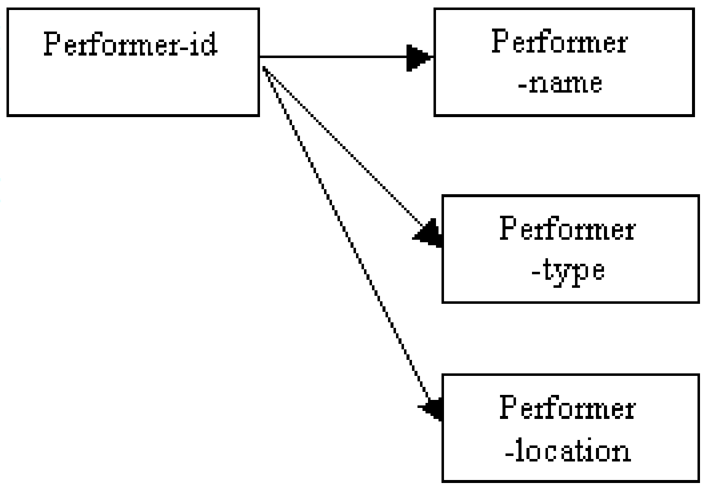
Diagrams can be used to indicate the dependencies between different attributes of an entity. We saw in the earlier chapter on entity-relationship modelling that one or more attributes could be identified as candidate keys before making a final selection of a primary key. When a primary key has been chosen, we may find that some attributes do not depend on the key, or some attributes depend only on part of the key. Determinacy diagrams offer the opportunity to examine the dependencies between attributes and the primary key in a visual representation.

Important

Determinant

When the value of one attribute allows us to identify the value of another attribute in the same relation, this first attribute is called a determinant. The determinant of a value might not be the primary key. This is true for groups of attributes as well, so if A is the determinant of B, A and B may either be single attributes, or more than one attribute.

In the diagram below, it can be seen that the name of a performer depends entirely on the performer-id (we know that this is a one-to-one relationship). We can say that performer-id functionally determines the performer-name, and this is shown by the arrow. In addition, the type and location of any particular performer are also determined by the performer-id.



It might be the case that there are performers who share the same family name (for example, a family of actors). Each member of the family who is an actor will have a unique performer-id (as the attribute performer-id is the primary key), but there may be more than one person with that particular name. The performer-name would not make a suitable choice for primary key for this reason. The performer-id uniquely determines the performer-name, but a performer-name may indicate more than one performer-id.

In a similar way, there may be more than one performer of a particular type; the performer-id will identify the performer-type for that specific individual. It is likely that any one location may have more than one performer based there; the location of any particular performer can be determined by means of the performer-id as the primary key. There are several possibilities for considering how the fee to a performer for a booking at a venue might be calculated, and these might include:

- flat rate fee for all performers for all venues
- fee negotiated with performer
- fee depends on performer's location
- fee depends on location of venue
- fee depends on performer type
- fee depends on date of booking
- fee depends on a combination of factors (e.g. performer and agent)

The method by which the fee is calculated will affect the way the data is modelled; this is because the value of the fee can be linked to a number of other attributes, and might not be determined by the performer-id alone as the primary key. The determinacy diagrams may be different depending on the particular method of calculating the fee.

If we consider some of the possibilities outlined above, we can identify the dependencies that affect the fee and create a determinacy diagram.

Direct dependencies

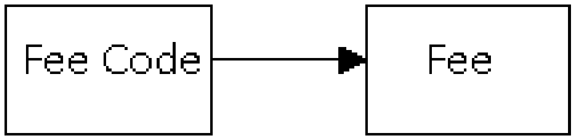
An example to illustrate direct dependencies might be: flat rate fee for all performers for all venues.

In this case, the fee could be regarded as another attribute of each performer, or could be linked to a performance (the number of performances determining the total amount earned). The fee could be regarded as an entity in its own right. We would need to take into account what would happen if the fees were to change. Would all fees change to the same new value? What would determine whether one performer earned a different fee from another? The answers to these questions would reveal underlying dependencies between the data.

If we assume that all performers are paid the same fee, and when the fee is changed it affects all performers in exactly the same way, we can identify the fee as a separate entity.

The value of the fee would then depend on the fee code. The fee is directly dependent on the fee code.

(Note that we would not want to insert the exact fee as a value for all performers because of the implications of updating the value when the fee changes.)



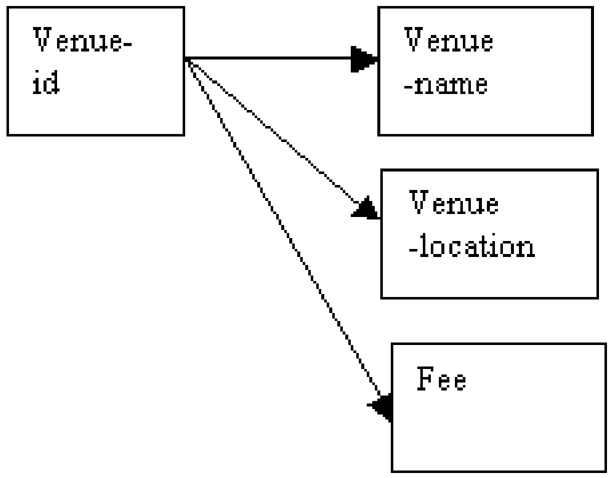
Transitive (indirect) dependencies

An example to illustrate transitive (also known as indirect) dependencies might be: fee depends on location of venue.

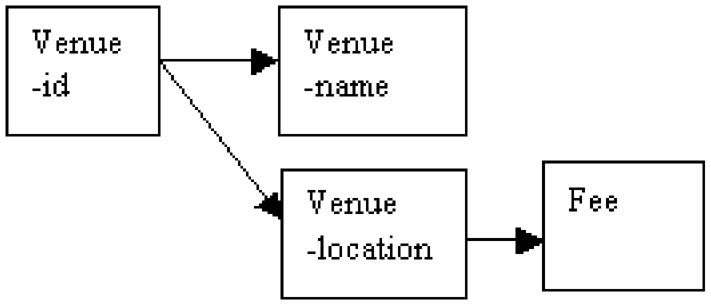
Where the value of the fee depends on the location of the venue, it is not possible to decide in advance what fee will be paid to a performer until details of the venue are known. This means that a booking must be made by an agent for a performer at a venue in order for the fee to be determined.

It will be necessary to find out whether the fee is determined by the specific venue, or whether all venues in the same location also attract the same fee.

If each venue has its own fee, then the fee will be determined by the venue-id, in the same way that other attributes of a particular venue, such as the name and location, are identified by venue-id as the key. This is a direct dependency.



On the other hand, if the fee applies to all venues in the same area, venues must be identified as belonging to specific areas in which a given fee applies. This is an indirect dependency, also known as a transitive dependency.



Important

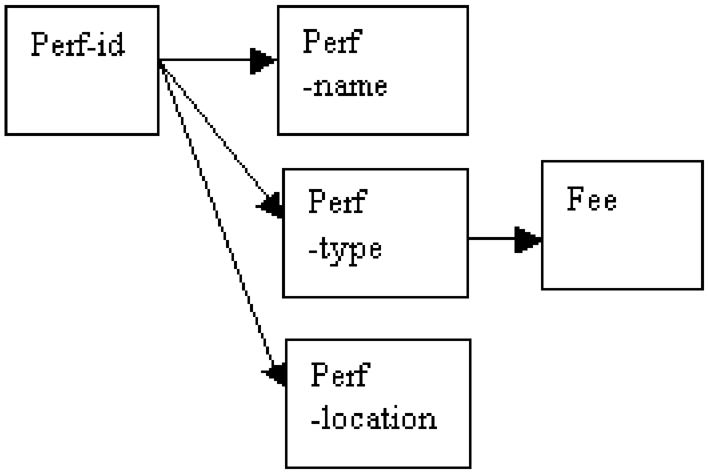
Transitive (indirect) dependency

Sometimes the value of an attribute is not determined directly from the primary key, but through the value of another attribute which is determined by the primary key; this relationship is known as a transitive dependency.

Another example of a transitive dependency

Consider the following attributes: fee depends on performer type.

Here the fee depends on whether the performer is an actor, dancer, singer or some other type of performer. The different types of performer need to be identified, and a fee specified in each case. The value of the fee does not depend directly on the performer-id, but is linked to the type of performer. This is another example of an indirect (or transitive) dependency.



Composite determinants and partial dependencies

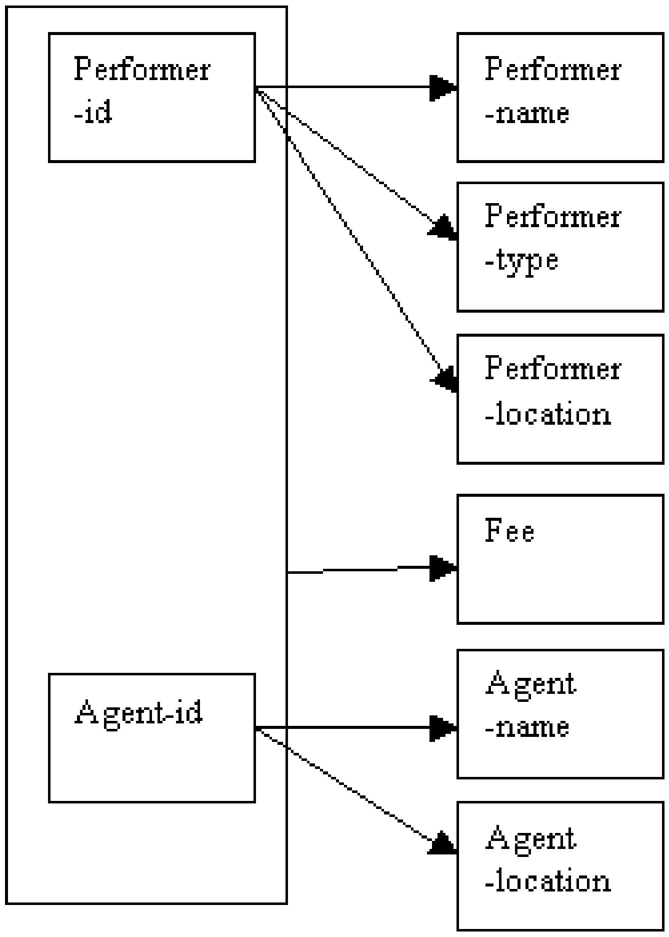
Sometimes the determinant is not a single attribute, but made up of two or more attributes. Consider the following: fee depends on a combination of factors (e.g. performer and agent).

Important

Composite determinant

If more than one value is required to determine the value of another attribute, the combination of values is known as a composite determinant.

If the fee is determined by more than one factor, both these elements must be taken into account. This is shown in the determinacy diagram on the right by the arrow including both the performer-id and the agent-id as the determinant items on which the fee depends. The attributes performer-id and agent-id are known as composite determinants.



Where every attribute in a primary key is required as a composite determinant for an attribute, the attribute is said to be fully functionally dependent on the key.

Note that the attributes that depend only on performer-id (such as the name, type and location of each performer), or agent-id (such as the agent and location of each agent) are shown linked directly to the appropriate key. If we take performer-id and agent-id as the key, we can say that the performer and agent details are partially dependent on the key. Partial dependency is when an attribute is functionally dependent on a proper subset of the key.

Important

Partial dependency

If the value of an attribute does not depend on an entire composite determinant, but only part of it, that relationship is known as a partial dependency.

Multiple determinants

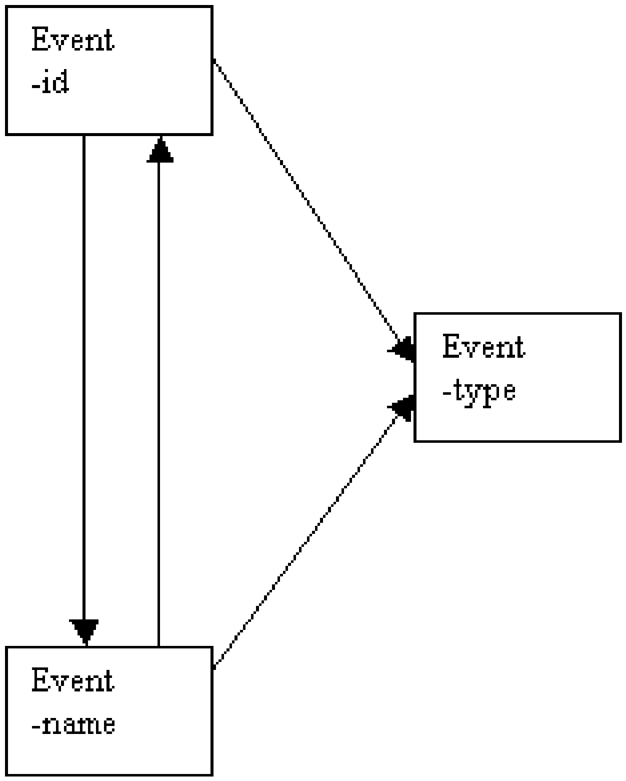
It is possible that there may be more than one attribute that can act as a determinant for other attributes. This is a slightly different situation from that of candidate keys, as not all determinants are necessarily candidate keys. If we wish to describe an event, we may find that there is a special relationship between the attributes event-id and event-name; each event will have a unique identification number, and also an unique name. The relationship between the event-id and the event-name is one-to-one. The better choice of primary key for the event would be event-id, which is a unique identification number.

The attribute event-name, while unique to each event, would not make such a good choice as the key because there can be problems in obtaining an exact match (e.g. “Quicktime”, “Quick time” and “Quick Time” would be regarded as different names).

We can show dependencies between the attributes event-id, event-name and event-type on a determinacy diagram.

Each event would have values for the attributes event-id, event-name and event-type.

In the determinacy diagram below, we can see that event-id is a determinant for the other two attributes, event-name and event-type.

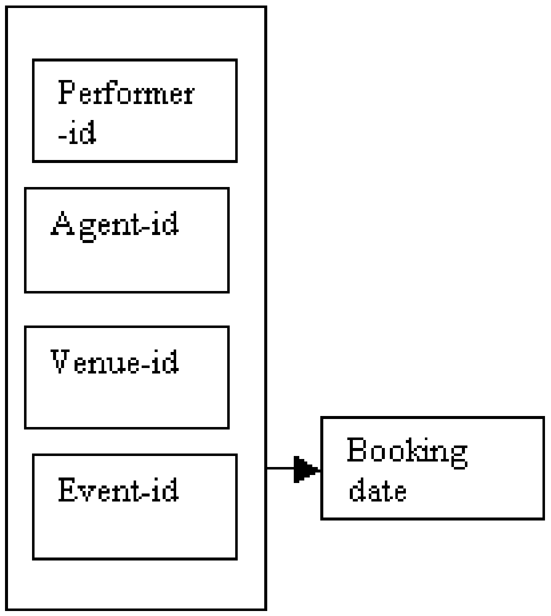


The determinacy diagram shows that the attribute event-name is also a determinant for the other two attributes, event-id and event-type. This is because there is a one-to-one relationship between event-id and event-name.

Overlapping determinants

There are sometimes cases where there is more than one combination of attributes that uniquely identifies a particular record. This means that the determinants have attributes in common. In certain circumstances, there may be a special relationship between the attributes, so that each uniquely determines the value of the other.

An example of this may be where each module in a degree programme has a unique module code and a unique module name. It would be possible to use either the module code or the module name as the determinant. In addition, the module code determines the module name, and the module name determines the module code.



In the context of our example relating to performers, agents, venues and events, we will also need to be able to identify bookings. We find that each booking can be identified by a different combination of attributes.

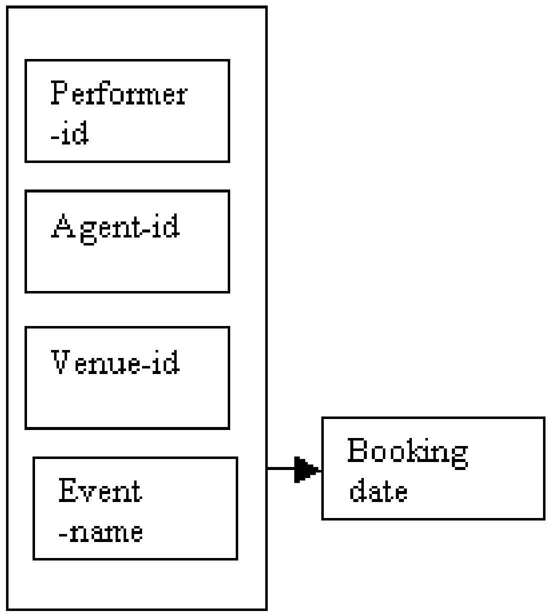
When a booking is made, the performer-id, agent-id, venue-id and event-id are all required in order to specify a particular event occurring on a given date. This also needs to be represented using a determinacy diagram.

Each booking can be identified by the primary key, which is shown on the right as a combination of the attributes performer-id, agent-id, venue-id and event-id.

Note that in this instance, the arrow (coming from the outer box) indicates that all four key attributes are used to identify the booking date.

We know that each event can be identified either by the event-id or the event-name; this means that we could have an alternative representation in the determinacy diagram, substituting the attribute event-name for event-id as part of the combined key.

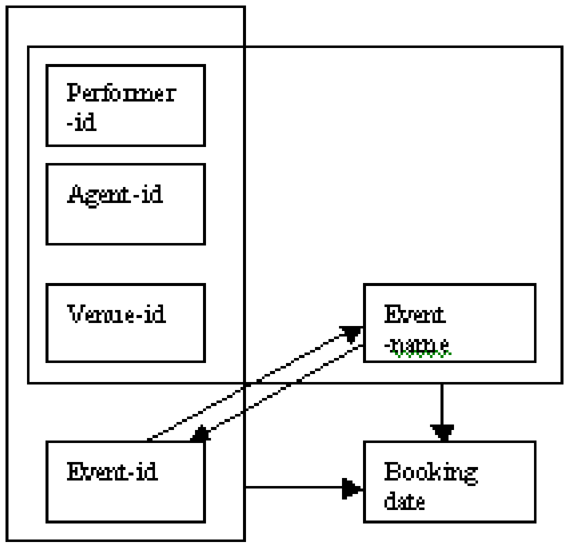
An alternative primary key for each booking would be a combination of performer-id, agent-id, venue-id and event-name.



Here again, the arrow (coming from the outer box) indicates that all four key attributes are used to identify the booking date.

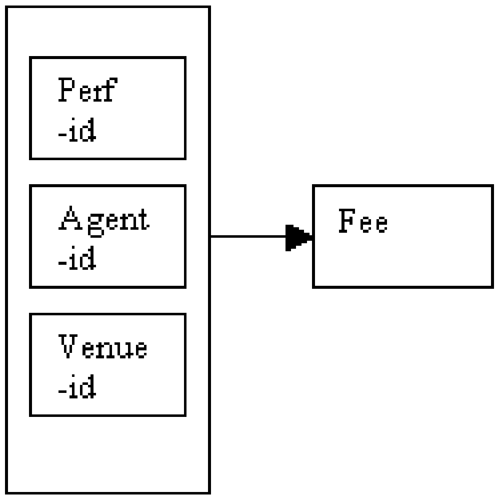
Here we have an overlapping key. The attribute event-name is a determinant, although it is not a candidate key for its own data. We would not want to use the event-name as a primary key, as it can present a problem in identifying the relevant tuple if the spelling is not exactly the same as in the relation.

The determinacy diagram also shows the relationship between the attributes event-id and event-name.



Exploring the determinant of ‘fee’ further

Consider the following determinacy diagram for attribute ‘fee’:

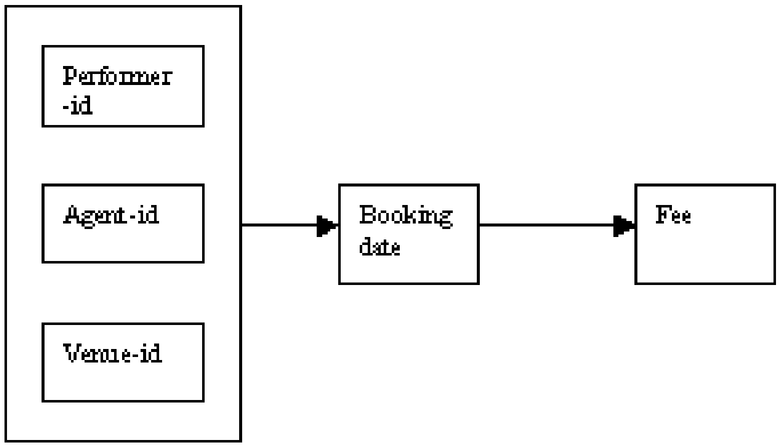


If a performer negotiates the same fee for all bookings, the fee depends on the performer-id, as each performer will have their own fee. This is a direct dependency.

Where the value of the fee depends the date of the booking, the value of the fee cannot be known until details of the booking are available.

This means that a booking must be made by an agent for a performer at a venue in order for the fee to be determined. It may be that a higher fee is paid in the summer months than at other times of the year.

The booking date will be determined by the composite determinant made up from the agent-id, performer-id and venue-id (as all three are involved). The booking date itself then determines the fee. There is therefore an indirect (or transitive) dependency between the composite key and the fee.



Finding keys using functional dependency

Functional dependency (FDs) helps to find keys for relations. To identify all candidate keys, check whether each determinant uniquely identifies tuples in the relation. Let's define another important concept called attribute closure.

Attribute closure

The closure of X, written X+, is all the attributes functionally determined by X. That is, X+ gives all the values that follow uniquely from X. Attribute closure is used to find keys and to see if a functional dependency is true or false.

To find the closure of X+, follow the following steps:

- ans = X
- For every Y→Z such that Y   ans, add Z to ans
- Repeat until no more changes to X+ are possible

For example, given a relation R, such that

R(S, C, P, M, G, L, T)

FDs {SC $\rightarrow$ PMG, SL $\rightarrow$ C, CT $\rightarrow$ L, TL $\rightarrow$ C, SP $\rightarrow$ C

Can we answer the following two questions?

Is SL a key for R?

- Start with ans = {SL}
- Using 2nd FD, SL functionally determines C, so we add C to the ans, ans = {SLC}
- Using 1st FD, SC functionally determines PMG, so we add PMG to the ans, ans = {SLCPMG}
- No more attributes can be added because no subset of the ans functionally determines other attributes, so (SL)+ is SLCPMG

Is SL a key for R? No, because the closure of SL is not equal to all the attributes in R

Does SL $\rightarrow$ PG?

Yes, because PG is in (SL)+

Normalisation

In the context of databases, normalisation is a process that ensures the data is structured in such a way that attributes are grouped with the primary key that provides unique identification. This means that some attributes, which may not depend directly on the primary key, may be extracted to form a new relation.

There are a number of reasons for performing normalisation; normalised data is resilient against anomalies that may occur in updating values by insertion, amendment or deletion, and other inconsistencies, and makes better use of storage space.

The process of normalisation does not alter the values associated with the attributes of an entity; rather, it develops a structure based upon the logical connections and linkages that exist between the data.

Important

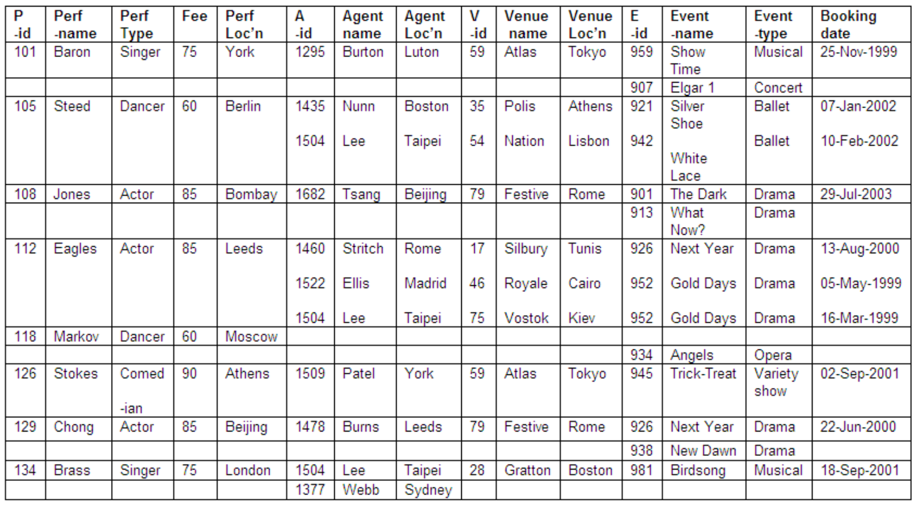
Normalisation

When a solution to a database problem is required, normalisation is the process which is used to ensure that data is structured in a logical and robust format. The most common transformations are from un-normalised data, through first and second, to third normal form. More advanced transformations are possible, including Boyce-Codd, fourth and fifth normal forms.

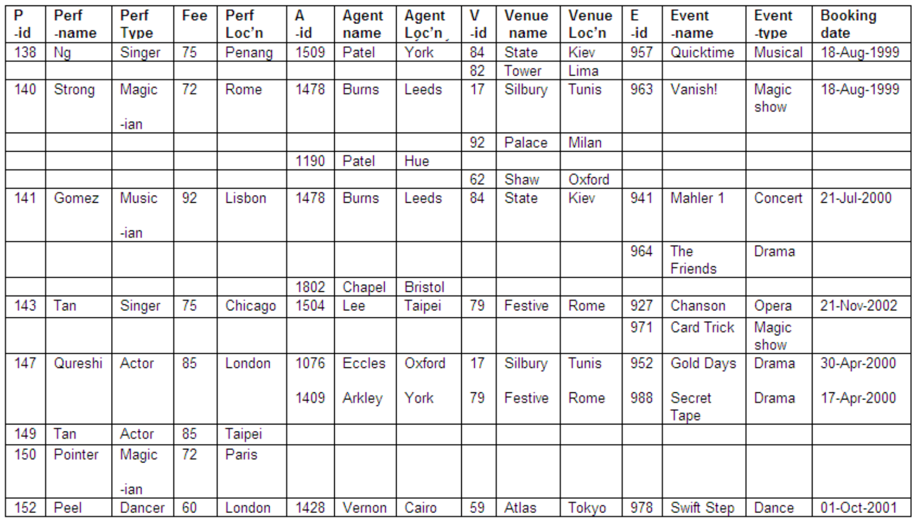
If we consider the data before it has undergone the normalisation process, we regard it as un-normalised.

Un-normalised data

In the table below we have details of performers, their agents, performance venues and booking dates in an un-normalised format. In this particular example, the fee paid to the performer depends on the performer-type (for example, the fee to all actors is 85).



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
101	Baron	Singer	75	York	1295	Burton	Luton	59	Atlas	Tokyo	959	Show Time	Musical	25-Nov-1999
											907	Elgar 1	Concert	
105	Steed	Dancer	60	Berlin	1435 1504	Nunn Lee	Boston Taipei	35 54	Polis Nation	Athens Lisbon	921 942	Silver Shoe White Lace	Ballet Ballet	07-Jan-2002 10-Feb-2002
108	Jones	Actor	85	Bombay	1682	Tsang	Beijing	79	Festive	Rome	901	The Dark	Drama	29-Jul-2003
											913	What Now?	Drama	
112	Eagles	Actor	85	Leeds	1460 1522 1504	Stritch Ellis Lee	Rome Madrid Taipei	17 46 75	Silbury Royale Vostok	Tunis Cairo Kiev	926 952 952	Next Year Gold Days Gold Days	Drama Drama Drama	13-Aug-2000 05-May-1999 16-Mar-1999
118	Markov	Dancer	60	Moscow										
											934	Angels	Opera	
126	Stokes	Comed -ian	90	Athens	1509	Patel	York	59	Atlas	Tokyo	945	Trick-Treat	Variety show	02-Sep-2001
129	Chong	Actor	85	Beijing	1478	Burns	Leeds	79	Festive	Rome	926	Next Year	Drama	22-Jun-2000
											938	New Dawn	Drama	
134	Brass	Singer	75	London	1504	Lee	Taipei	28	Gratton	Boston	981	Birdsong	Musical	18-Sep-2001
					1377	Webb	Sydney							



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
138	Ng	Singer	75	Penang	1509	Patel	York	84	State	Kiev	957	Quicktime	Musical	18-Aug-1999
								82	Tower	Lima				
140	Strong	Magic -ian	72	Rome	1478	Burns	Leeds	17	Silbury	Tunis	963	Vanish!	Magic show	18-Aug-1999
								92	Palace	Milan				
					1190	Patel	Hue							
								62	Shaw	Oxford				
141	Gomez	Music -ian	92	Lisbon	1478	Burns	Leeds	84	State	Kiev	941	Mahler 1	Concert	21-Jul-2000
											964	The Friends	Drama	
					1802	Chapel	Bristol							
143	Tan	Singer	75	Chicago	1504	Lee	Taipei	79	Festive	Rome	927	Chanson	Opera	21-Nov-2002
											971	Card Trick	Magic show	
147	Qureshi	Actor	85	London	1076 1409	Eccles Arkley	Oxford York	17 79	Silbury Festive	Tunis Rome	952 988	Gold Days Secret Tape	Drama Drama	30-Apr-2000 17-Apr-2000
149	Tan	Actor	85	Taipei										
150	Pointer	Magic -ian	72	Paris										
152	Peel	Dancer	60	London	1428	Vernon	Cairo	59	Atlas	Tokyo	978	Swift Step	Dance	01-Oct-2001

To accommodate the size of the table, some headings have been shortened as shown below:

- P-id: performer-id
- Perf-name: performer-name
- Perf-type: performer-type
- Perf-Loc'n: performer-location
- A-id: agent-id
- Agent-Loc'n: agent-location
- V-id: venue-id
- Venue-Loc'n: venue-location
- E-id: event-id

Problems with un-normalised data

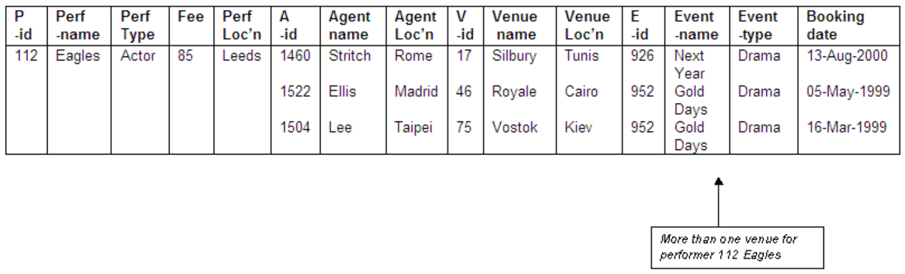
We can see from the table that some performers have more than one booking, whereas others have only a single booking, and some have none at all.

It is also shown in the table that agents are able to make bookings for different performers at different venues, but some agents have made no bookings, some venues have not been booked, and some events have not been scheduled.

The content of the table means that there is an inconsistent format, with multiple values for agents and venues associated with a single entry for some performers. The table as it stands would not be suitable for direct conversion into a relation.

Multiple venue bookings for Eagles

The performer Eagles (performer-id 112) has bookings at more than one venue, giving multiple rather than single entries for venue details.

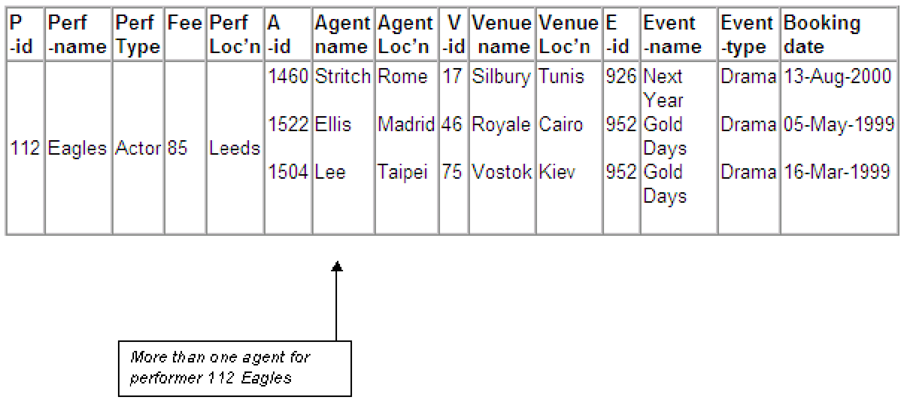


P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
112	Eagles	Actor	85	Leeds	1460	Stritch	Rome	17	Silbury	Tunis	926	Next Year	Drama	13-Aug-2000
					1522	Ellis	Madrid	46	Royale	Cairo	952	Gold Days	Drama	05-May-1999
					1504	Lee	Taipei	75	Vostok	Kiev	952	Gold Days	Drama	16-Mar-1999

More than one venue for performer 112 Eagles

Multiple agent bookings for Eagles

The performer Eagles (performer-id 112) has bookings made by more than one agent, and therefore there are multiple entries for agent details, rather than a single entry.

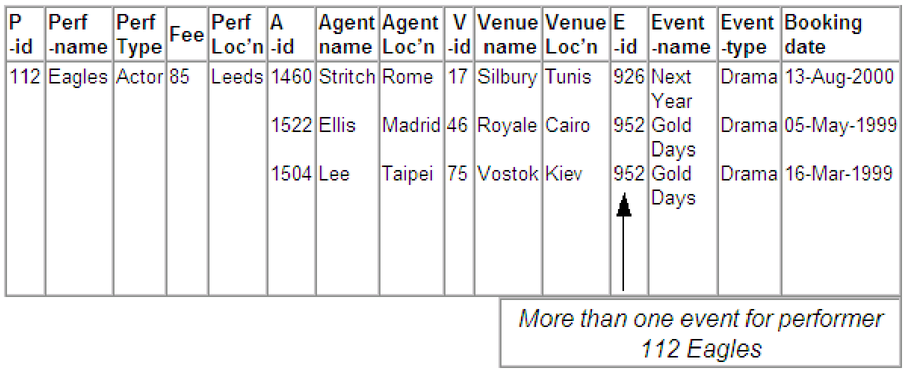


P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
					1460	Stritch	Rome	17	Silbury	Tunis	926	Next Year	Drama	13-Aug-2000
112	Eagles	Actor	85	Leeds	1522	Ellis	Madrid	46	Royale	Cairo	952	Gold Days	Drama	05-May-1999
					1504	Lee	Taipei	75	Vostok	Kiev	952	Gold Days	Drama	16-Mar-1999

More than one agent for performer 112 Eagles

Multiple event details for Eagles

The performer Eagles (performer-id 112) has bookings for more than one event, so that there are multiple entries for event details, rather than just one entry.



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
112	Eagles	Actor	85	Leeds	1460	Stritch	Rome	17	Silbury	Tunis	926	Next Year	Drama	13-Aug-2000
					1522	Ellis	Madrid	46	Royale	Cairo	952	Gold Days	Drama	05-May-1999
					1504	Lee	Taipei	75	Vostok	Kiev	952	Gold Days	Drama	16-Mar-1999

More than one event for performer 112 Eagles

Translating the table of un-normalised data into a relation, in what is called first normal form, will mean that the data contained in the table in represented in a more structured way. A relation in first normal form has only single entries for each attribute for every tuple. We shall now investigate how to perform this translation.

First normal form

The initial stage in the normalisation process is to convert a table of un-normalised data into a relation in first normal form. This means that we must extract the repeating groups of data that may appear in some rows of the table, and replace them with tuples where each attribute has only one value associated with it (at most).

Important

First normal form (1NF)

A relation is in first normal form if there is only one value at the intersection of each row and column.

Repeating groups in an un-normalised table of data are converted to first normal form by replacing them with tuples where each attribute has a single entry.

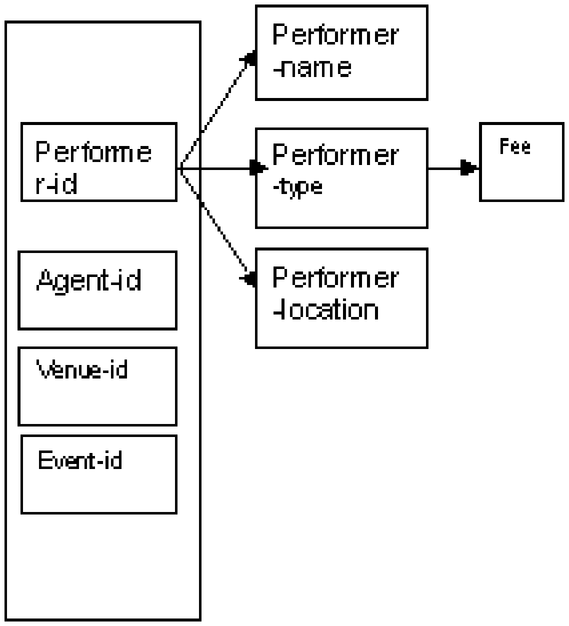
In order to convert an un-normalised relation into first normal form, we must identify the key attribute(s) involved. We can see from the table of un-normalised data that each performer has a code (performer-id), each agent is identified by an agent-id, each venue is determined by a venue-id and each event has an event-id.

Performer details

The details associated with each performer depend on the performer-id as the primary key.

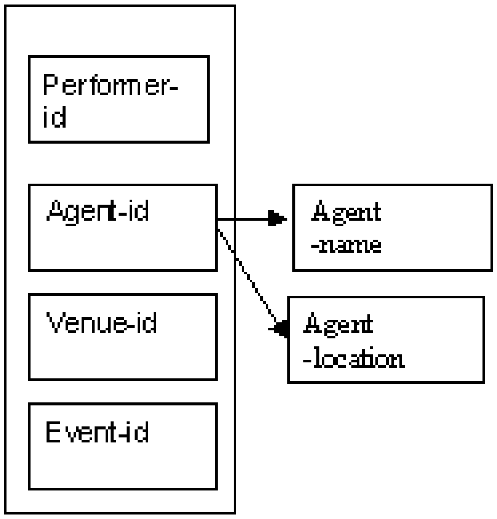
Note that the arrows coming directly from performer-id indicate that the performer attributes depend only on the key attribute performer-id, and not agent-id, venue-id or event-id.

We know that the fee in this case depends on the type of performer, and not directly on the primary key. This is shown in the diagram by the link between performer-type and fee.



Agent details

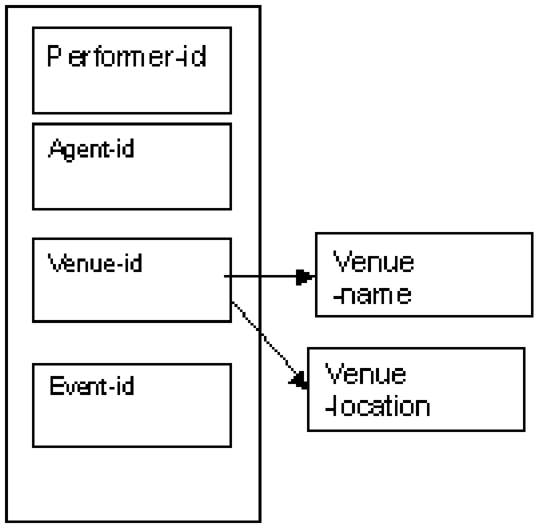
The information about each agent depends on the agent-id as the primary key.



Note that the arrow from agent-id indicates that the agent attributes depend only on agent-id as the key attribute, and not performer-id, venue-id or event-id.

Venue details

The primary key, venue-id, determines the name and location of each venue.



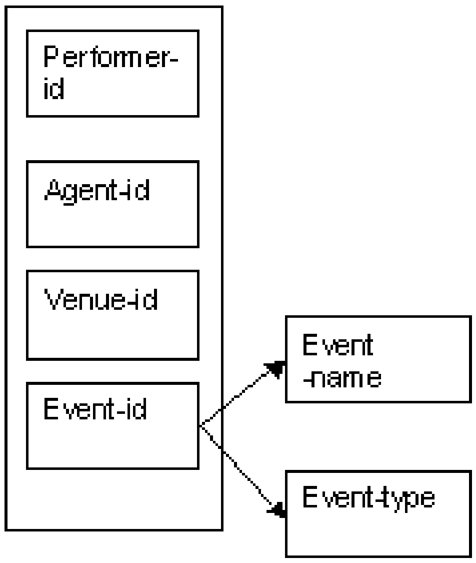
Note that the venue-name depends only on the venue-id as shown by the arrow in the diagram. The attributes performer-id, agent-id and event-id do not determine the venue-name.

Event details

We can consider the representation of events from two angles. We have two attributes which can be used as determinants: event-id and event-name. We can examine each in turn using a determinacy diagram, and then show the relationships between all three attributes (event-id, event-name and event-type) on a single determinacy diagram.

Event-id as the determinant

The primary key, event-id, determines the name and type of each event. There is a one-to-one relationship between event-id and event-name; either could be used to identify the other.



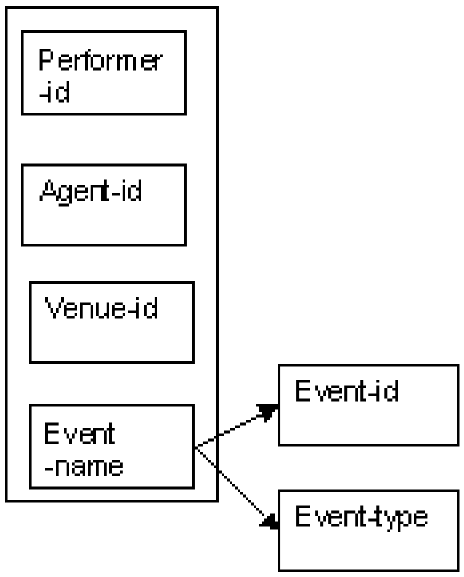
Note that the event-name depends only on the event-id as shown by the arrow in the diagram. The attributes performer-id, agent-id and venue-id do not determine the event-name.

Event-name as the determinant

There is a special relationship between the attributes event-id and event-name; each event-id and each event-name is unique.

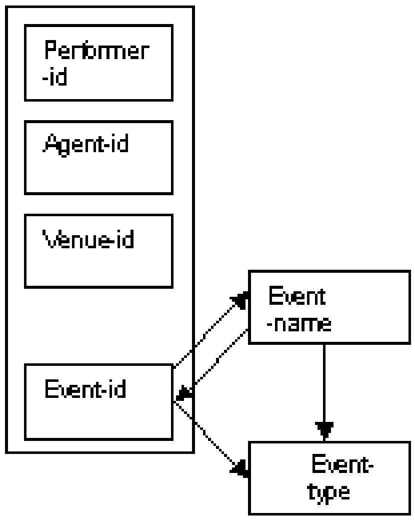
This means that we could use either the event-id or the event-name as the determinant for locating details about an event.

The determinacy diagram below shows the event-name being used as the determinant, although we would not want to use it as the primary key, as names can be difficult to get exactly right.



Event-id and event-name as determinants

We can show the special relationship between event-id and event-name by arrows illustrating the link in each direction.

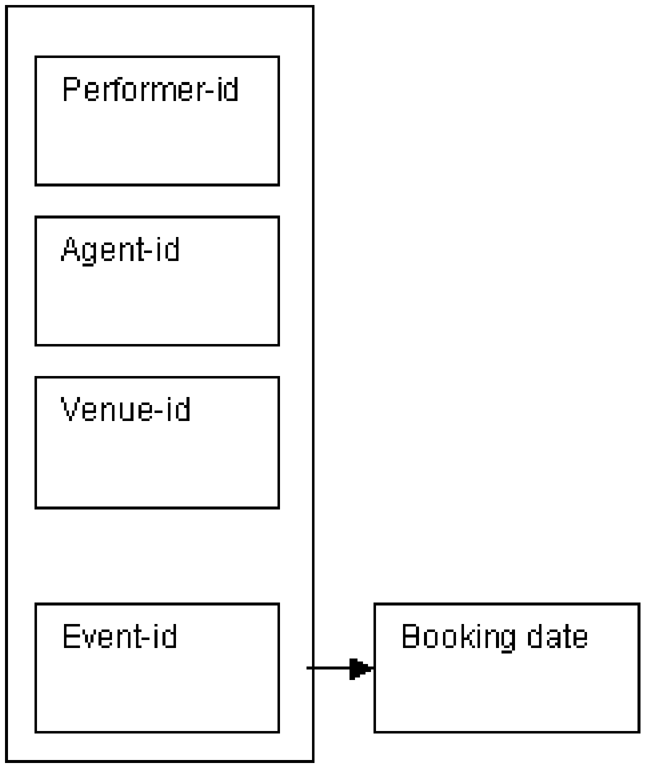


As either event-id or event-name can determine the event-type, there are links between event-id and event-type, and also between event-name and event-type.

Booking detail

In addition to the performers, agents and venues, we need to be able to identify the bookings that have been made. When a booking is made, the performer-id, agent-id, venue-id and event-id are all required in order to specify a particular event occurring on a given date. This also needs to be represented using a determinacy diagram.

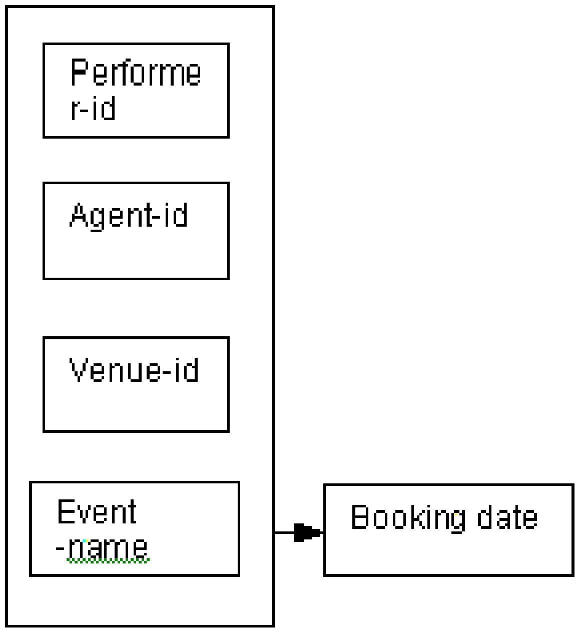
Each booking can be identified by the primary key, which is shown on the right as a combination of the attributes performer-id, agent-id, venue-id and event-id.



Note that in this instance, the arrow (coming from the outer box) indicates that all four key attributes are used to identify the booking date.

We know that each event can be identified either by the event-id or the event-name; this means that we could have an alternative representation in the determinacy diagram, substituting the attribute event-name for event-id as part of the combined key.

An alternative primary key for each booking would be a combination of performer-id, agent-id, venue-id and event-name.



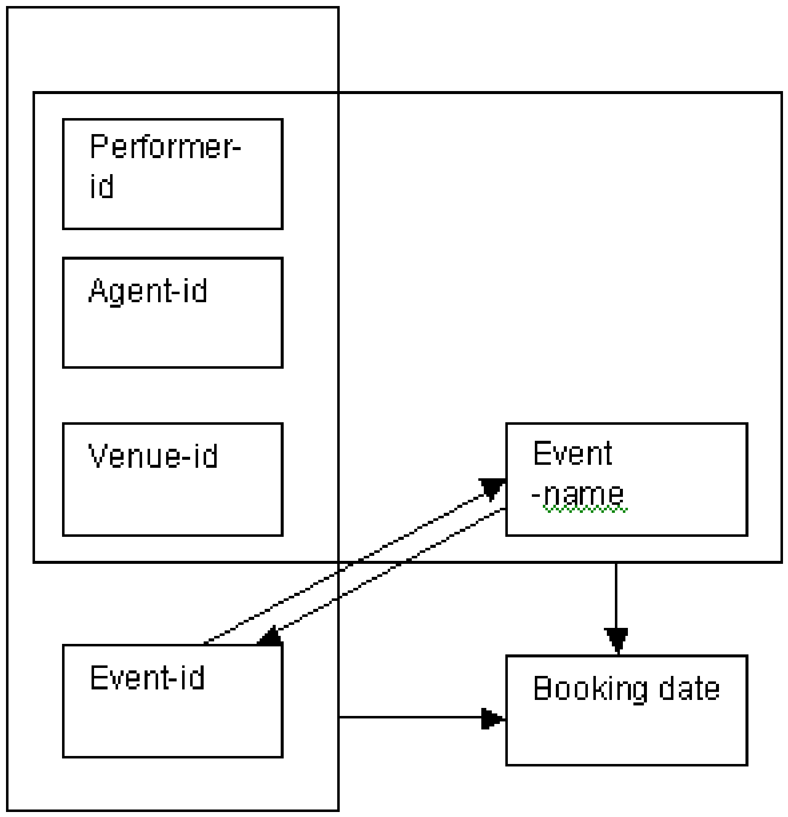
Here again, the arrow (coming from the outer box) indicates that all four key attributes are used to identify the booking date.

Here we have an overlapping key. The attribute event-name is a determinant, although it is not a candidate key for its own data. We would not want to use the event-name as a primary key, as it can present a problem in identifying the relevant tuple if the spelling is not exactly the same as in the relation.

We can show the overlapping nature of the keys for the booking details in a determinacy diagram.

The determinacy diagram below shows that the booking date could be located through a primary key constructed from the attributes performer-id, agent-id, venue-id and event-id, or by means of a primary key combining the attributes performer-id, agent-id, venue-id and event-name.

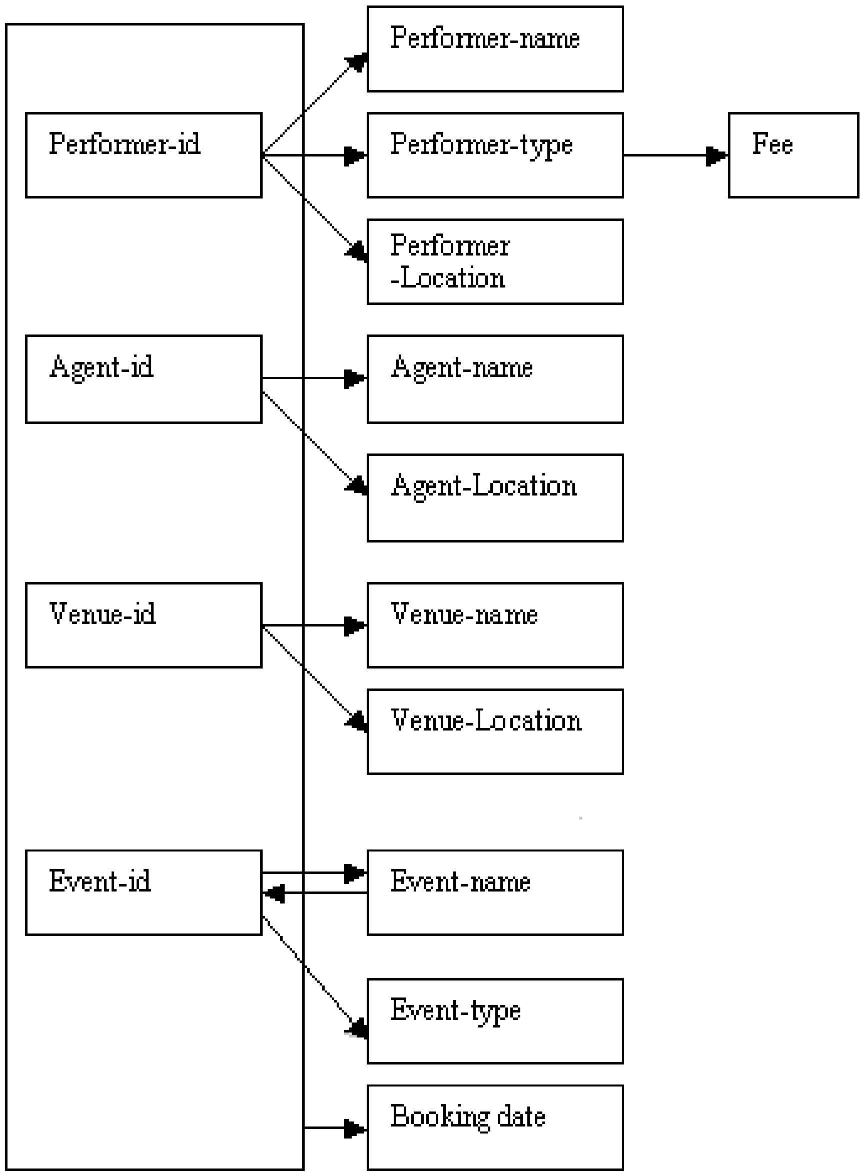
The determinacy diagram also shows the relationship between the attributes event-id and event-name.



It is not common to find overlapping keys; it is more usual to have a unique identifier which distinguishes between different items (for example, the performer-id will distinguish between different performers who may happen to have the same name). At this point in the normalisation process, overlapping keys do not present a problem, but they will be dealt with at a later stage. We will use the event-id in preference to the event-name for the time being, but we will need to remember the special relationship that exists between these two attributes.

Determinacy diagram for first normal form

The information represented in these four categories (performer, agent, venue and booking) can be displayed in a single diagram for first normal form (1NF):

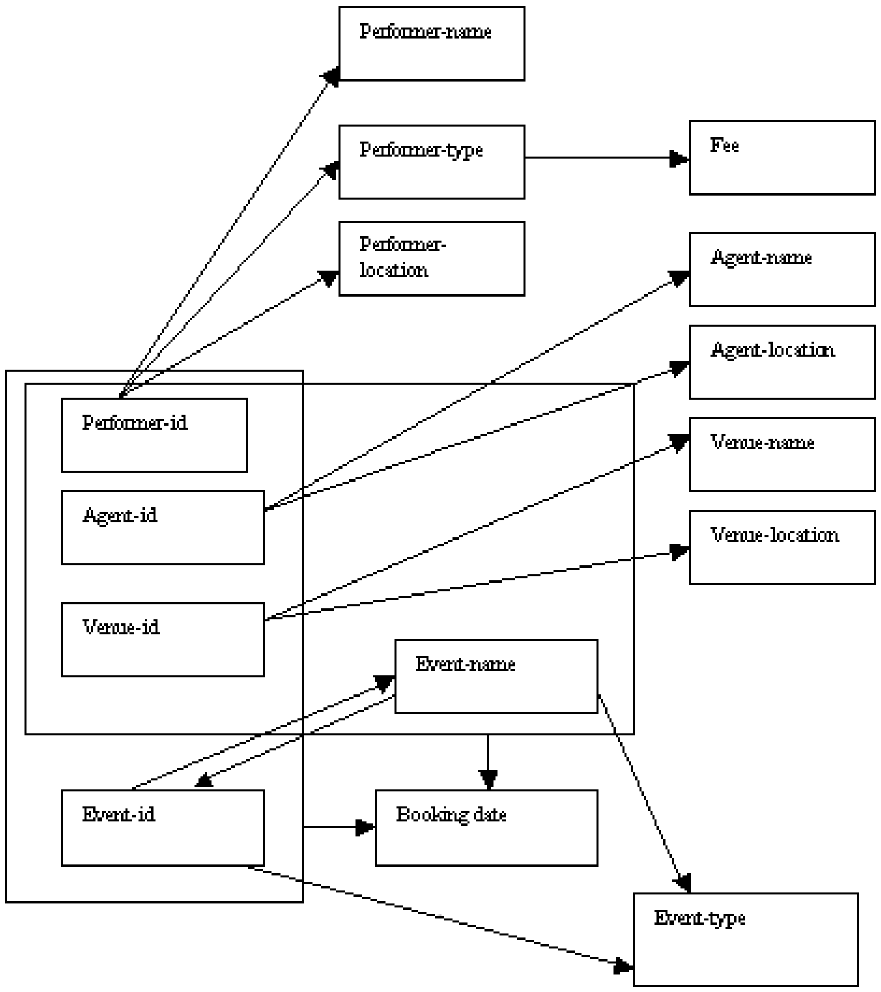


The combined determinacy diagram (above) for first normal form shows that:

- The performer attributes (name, type, location and fee) depend only on the key performer-id.
- The agent attributes (name and location) depend only on the key agent-id.

- The venue attributes (name and location) depend only on the key venue-id.
- The event attributes (name and type) depend only on the key event-id (we will examine the relationship between event-id and event-name later).
- The booking details depend on all four key attributes: performer-id, agent-id ,venue-id and event-id.

The full determinacy diagram for first normal form, showing the overlapping keys, is shown below:



The result of converting an un-normalised table of data into first normal form is to remove repeating values, so that each line in the table has the same format, with only one value in each column for each row. This means that there will be only one value for each attribute for each tuple in a relation in first normal form.

Where more than one booking has been made for a performer, each booking is now given as a separate entry.

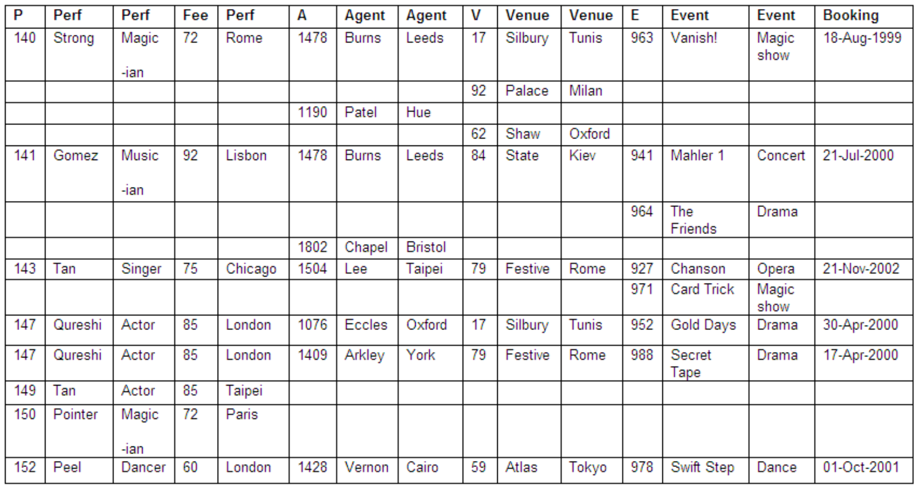
The original table of data has been converted into a relation in first normal form, as shown below. The relation has the same structure as the determinacy diagram, both being in first normal form, and exhibiting the following characteristics:

- All performers have a performer-id as the primary key.
- Details about agents can be determined from the primary key agent-id.
- Any venue can be identified by the venue-id as the primary key.
- All events can be determined by event-id as the primary key.
- Where a booking has been made, the key attributes performer-id, agent-id, venue-id and event-id all have values, which combine to identify each particular booking as a composite (or compound) primary key.

We can now convert our table of un-normalised data into a relation in first normal form (1NF). Note that there is at most a single value at the intersection of each row and column. This process is sometimes known as 'flattening' the table.

Table of relation in first normal form (1NF)

P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
101	Baron	Singer	75	York	1295	Burton	Luton	59	Atlas	Tokyo	959	Show Time	Musical	25-Nov-1999
											907	Elgar 1	Concert	
105	Steed	Dancer	60	Berlin	1435	Nunn	Boston	35	Polis	Athens	921	Silver Shoe	Ballet	07-Jan-2002
105	Steed	Dancer	60	Berlin	1504	Lee	Taipei	54	Nation	Lisbon	942	White Lace	Ballet	10-Feb-2002
108	Jones	Actor	85	Bombay	1682	Tsang	Beijing	79	Festive	Rome	901	The Dark	Drama	29-Jul-2003
											913	What Now?	Drama	
112	Eagles	Actor	85	Leeds	1460	Stritch	Rome	17	Silbury	Tunis	926	Next Year	Drama	13-Aug-2000
112	Eagles	Actor	85	Leeds	1522	Ellis	Madrid	46	Royale	Cairo	952	Gold Days	Drama	05-May-1999
112	Eagles	Actor	85	Leeds	1504	Lee	Taipei	75	Vostok	Kiev	952	Gold Days	Drama	16-Mar-1999
118	Markov	Dancer	60	Moscow										
											934	Angels	Opera	
126	Stokes	Comed -ian	90	Athens	1509	Patel	York	59	Atlas	Tokyo	945	Trick-Treat	Variety Show	02-Sep-2001
129	Chong	Actor	85	Beijing	1478	Burns	Leeds	79	Festive	Rome	926	Next Year	Drama	22-Jun-2000
											938	New Dawn	Drama	
134	Brass	Singer	75	London	1504	Lee	Taipei	28	Gratton	Boston	981	Birdsong	Musical	18-Sep-2001
					1377	Webb	Sydney							



P	Perf	Perf	Fee	Perf	A	Agent	Agent	V	Venue	Venue	E	Event	Event	Booking
140	Strong	Magic -ian	72	Rome	1478	Burns	Leeds	17	Silbury	Tunis	963	Vanish!	Magic show	18-Aug-1999
								92	Palace	Milan				
					1190	Patel	Hue							
								62	Shaw	Oxford				
141	Gomez	Music -ian	92	Lisbon	1478	Burns	Leeds	84	State	Kiev	941	Mahler 1	Concert	21-Jul-2000
											964	The Friends	Drama	
					1802	Chapel	Bristol							
143	Tan	Singer	75	Chicago	1504	Lee	Taipei	79	Festive	Rome	927	Chanson	Opera	21-Nov-2002
											971	Card Trick	Magic show	
147	Qureshi	Actor	85	London	1076	Eccles	Oxford	17	Silbury	Tunis	952	Gold Days	Drama	30-Apr-2000
147	Qureshi	Actor	85	London	1409	Arkley	York	79	Festive	Rome	988	Secret Tape	Drama	17-Apr-2000
149	Tan	Actor	85	Taipei										
150	Pointer	Magic -ian	72	Paris										
152	Peel	Dancer	60	London	1428	Vernon	Cairo	59	Atlas	Tokyo	978	Swift Step	Dance	01-Oct-2001

We can see that the relation in first normal form will still exhibit some problems when we try to insert new tuples, update existing values or delete existing tuples. This is because there is no primary key for the whole table, although each major component has its own key (performer, agent, venue, event and booking).

Insertion anomalies of first normal form

There is a problem in selecting a suitable key for the table in its current format.

If we wish to insert details for a new performer, agent, venue or booking, we need to be able to identify the key attribute and determine a value for the key for the new record, for it to be entered as a tuple in the relation.

There is no clear candidate for a key for the whole relation in first normal form. We cannot use the performer-id as a key, because not every record in the table has a performer specified. The following examples illustrate this: the venue 62 Shaw has no performer, no event and no agent; the agent 1377 Webb has made no bookings for performers, venues or events; and the event 938 New Dawn has no performer, agent or venue. A null value cannot be allowed in a key field (for reasons of entity integrity, as discussed in Chapter 2).

If we made up a fictitious performer-id value to use as the key when we wanted to insert a new agent, a new venue or a new event, we would then generate another set of problems, such as apparent double bookings.

We need to consider the possibilities for a key for the whole relation in first normal form, and identify any problems that might arise with each option. The use of the following attributes as the primary key will be considered in turn:

- Performer-id

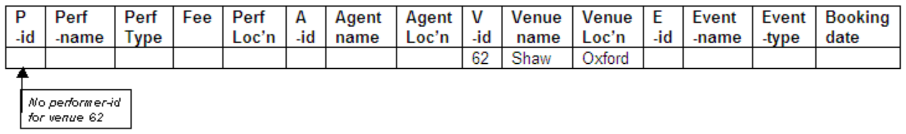
- Agent-id
- Venue-id
- Event-id
- Performer-id, agent-id, venue-id and event-id combined

Would the attribute performer-id make a suitable key for the relation in 1NF?

The attribute performer-id is the primary key for performers, but it cannot be used as the key for the whole relation in first normal form as there are some cases where there is no relevant value, as shown in the following examples:

No performer-id for Shaw

The venue Shaw (venue-id 62) has not been used for any bookings, and therefore has no performer-id associated with it that could be used as a key.

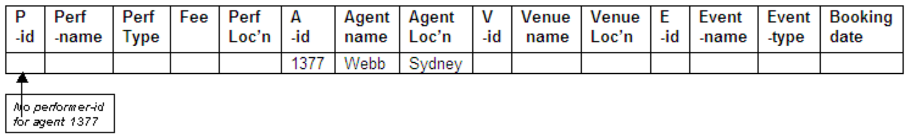


P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
								62	Shaw	Oxford				

No performer-id for venue 62

No performer-id for Webb

The agent Webb (agent-id 1377) has made no bookings for performers, and thus there is no appropriate performer-id that could be used as a key.



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
					1377	Webb	Sydney							

No performer-id for agent 1377

No performer-id for New Dawn

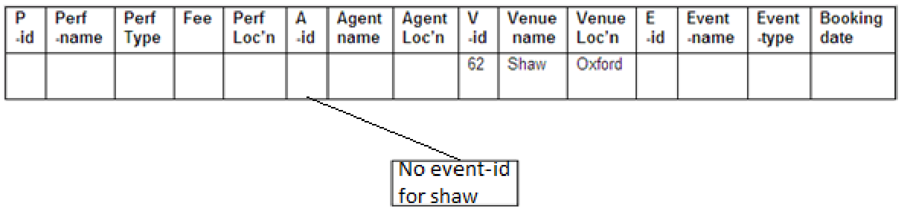
There are no bookings for the event New Dawn (event-id 938), and therefore there is no associated performer-id that could be used as a key.

Would the attribute agent-id make a suitable key for the relation in 1NF?

While it is the primary key for agents, the attribute agent-id would not make a good choice as the key for the whole relation in first normal form as here, too, there are times where there is no value present. This is illustrated below.

No agent-id for Shaw

No bookings have been made for the venue Shaw (venue-id 62), and therefore no agent-id is available to be used as a key.

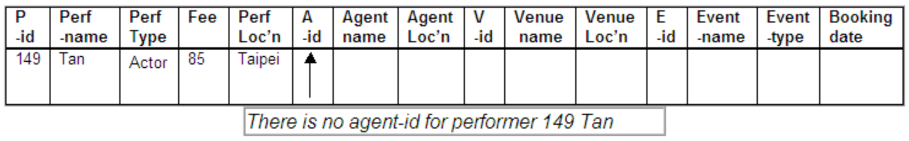


P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
								62	Shaw	Oxford				

No event-id for shaw

No agent-id for Tan

The actor Tan (performer-id 149) has no bookings and therefore no agent-id is available to be used as a key.



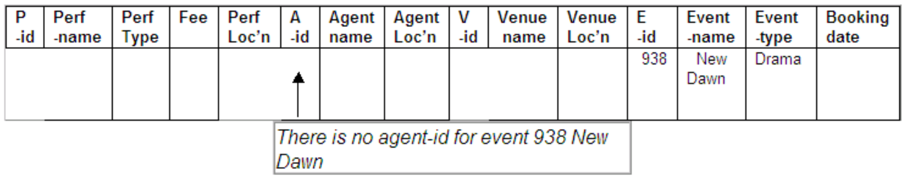
P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
149	Tan	Actor	85	Taipei										

There is no agent-id for performer 149 Tan

Note that the performer-id as primary key for performers distinguishes between 149 Tan the actor, and 143 Tan the singer (who does have a booking).

No agent-id for New Dawn

There are no bookings for the event New Dawn (event-id 938), and therefore there is no agent-id that could be used as a key.



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
											938	New Dawn	Drama	

There is no agent-id for event 938 New Dawn

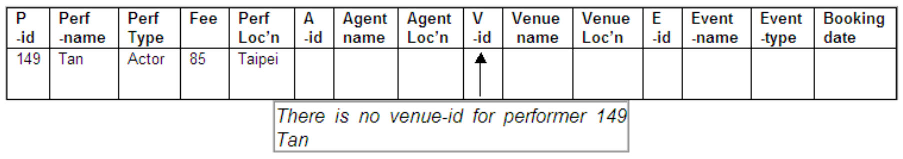
We can conclude that the attribute agent-id would not make a suitable key for the relation in first normal form.

Would the attribute venue-id make a suitable key for the relation in 1NF?

The attribute venue-id is the primary key for all venues, but it cannot be employed as the key for the whole relation in first normal form as there are instances where no value has been allocated, for example:

No venue-id for Tan

The actor Tan (performer-id 149) has no bookings at a venue and therefore there is no venue-id that can be used as a key.

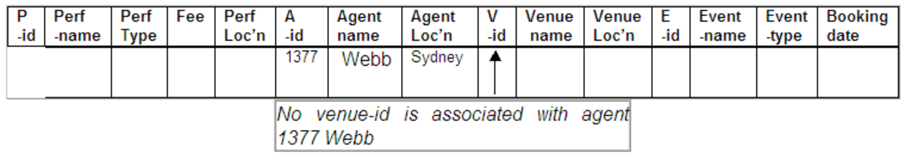


P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
149	Tan	Actor	85	Taipei										

There is no venue-id for performer 149 Tan

No venue-id for Webb

The agent Webb (agent-id 1377) has made no bookings, and is therefore not associated with any venue-id that could be used as a key.

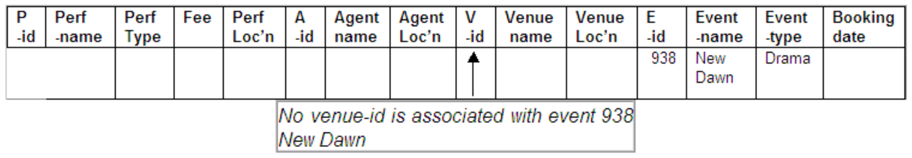


P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
					1377	Webb	Sydney							

No venue-id is associated with agent 1377 Webb

No venue-id for New Dawn

There are no bookings for the event New Dawn (event-id 938), and therefore there is no venue-id that could be used as a key.



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
											938	New Dawn	Drama	

No venue-id is associated with event 938 New Dawn

We can conclude that the attribute venue-id would not make a suitable key for the relation in first normal form.

Would the attribute event-id make a suitable key for the relation in 1NF?

The attribute event-id is the primary key for events (although the event-name could also be used as the primary key). The examples below demonstrate that the event-id cannot be used as the key for the whole relation in first normal form, as there are cases where there is no value for the event-id.

No event-id for Tan

The actor Tan (performer-id 149) has no bookings at an event and therefore there is no event-id that can be used as a key.

P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
149	Tan	Actor	85	Taipei										

There is no event-id for performer 149 Tan

No event-id for Shaw

The venue Shaw (venue-id 62) has not been used for any bookings, and therefore there is no event-id associated with it that could be used as a key.

P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
								62	Shaw	Oxford				

No event-id for venue 62

No event-id for Webb

The agent Webb (agent-id 1377) has made no bookings, and thus there is no appropriate event-id that could be used as a key.

P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
					1377	Webb	Sydney							

No event-id for agent 1377

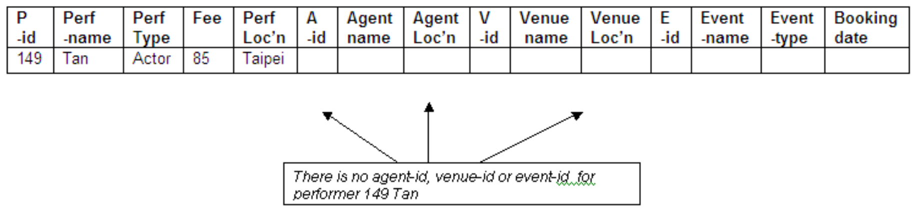
We can conclude that the attribute event-id would not make a suitable key for the relation in first normal form.

Would the combined attributes performer-id, agent-id, venue-id and event-id make a suitable key for the relation in 1NF?

The combined attributes performer-id, agent-id, venue-id and event-id serve as the primary key for all bookings, but this combination cannot be employed as the key for the whole relation in first normal form as there are entries where the key would be incomplete, for example:

No agent-id, venue-id or event-id for Tan

The actor Tan (performer-id 149) has no bookings made by an agent at a venue for an event and therefore there is no complete combined key value.

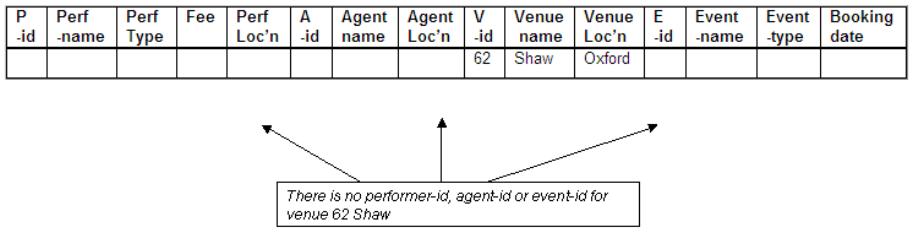


P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
149	Tan	Actor	85	Taipei										

There is no agent-id, venue-id or event-id for performer 149 Tan

No performer-id, agent-id or event-id for Shaw

No bookings have been made for the venue Shaw (venue-id 62), and therefore no complete combined key is available, as there is no performer, agent or event associated with the venue.

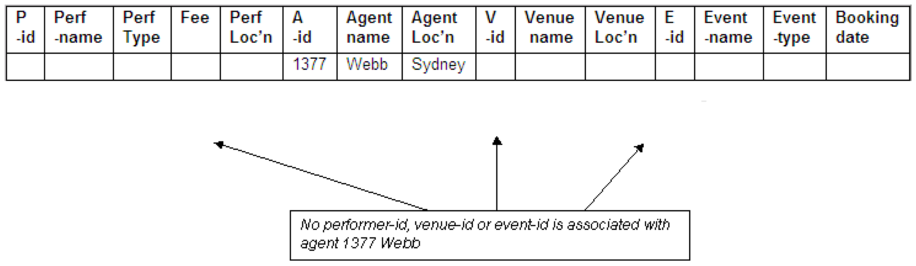


P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
								62	Shaw	Oxford				

There is no performer-id, agent-id or event-id for venue 62 Shaw

No performer-id, venue-id or event-id for Webb

The agent Webb (agent-id 1377) has made no bookings, and there is therefore an incomplete combined key value for Webb (no performer, venue or event).

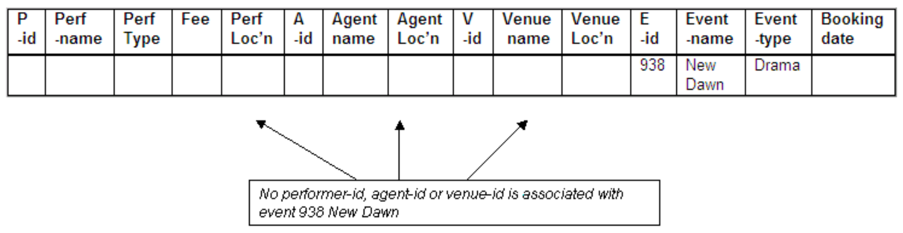


P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
					1377	Webb	Sydney							

No performer-id, venue-id or event-id is associated with agent 1377 Webb

No performer-id, agent-id or venue-id for New Dawn

The event New Dawn has not been booked, and therefore there is no complete combined key available as there is no performer, agent or venue associated with the event.



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
											938	New Dawn	Drama	

No performer-id, agent-id or venue-id is associated with event 938 New Dawn

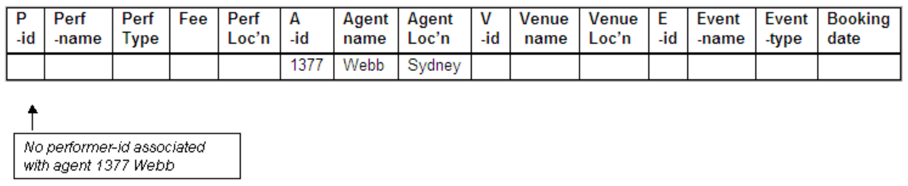
We can conclude that the combination of the attributes performer-id, agent-id, venue-id and event-id would not make a suitable key for the relation in first normal form.

There is no obvious choice for a primary key. The attributes that we might expect to be able to use as a key (such as performer-id, agent-id, venue-id and event-id) are unsuitable because a value is not always available, and it is not possible to have a key field with a null (or empty) value (because of the requirements of entity integrity).

Arbitrary selection of a primary key for relation in 1NF

If we take an alternative approach and arbitrarily select the performer-id as the key field, this will also lead to problems.

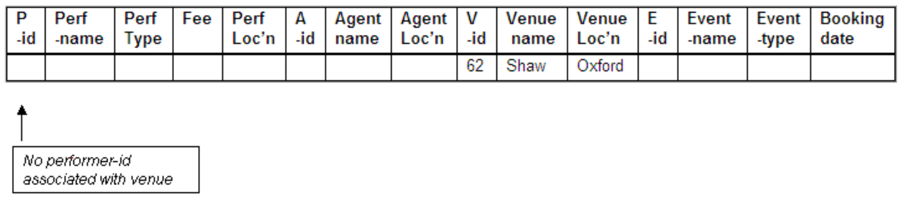
We would not be able to insert details about new agents who have yet to make a booking, as they will not have a performer-id associated with them. Neither would it be possible to retain the tuple on agent Webb (agent-id 1377), who has yet to make a booking.



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
					1377	Webb	Sydney							

No performer-id associated with agent 1377 Webb

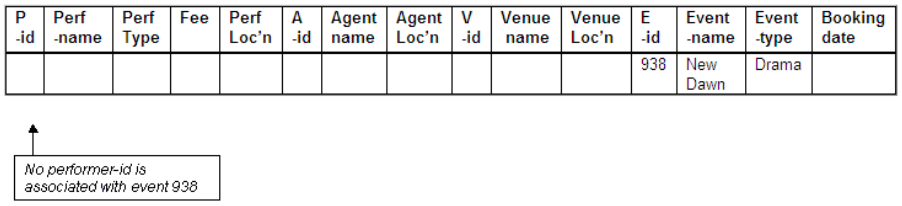
We would not be able to insert details about new venues that have not yet been used for a booking, as they too will not have a performer-id associated with them. In this instance, it would not be possible to retain the tuple on the venue Shaw (venue-id 62).



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
								62	Shaw	Oxford				

No performer-id associated with venue

We would not be able to insert details about new events that had not yet been booked, as any such event will not have a performer-id associated with it. This means that we would not be able to retain the tuple on the event New Dawn, as it has not been used for a booking.



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
											938	New Dawn	Drama	

No performer-id is associated with event 938

We can see that there is no single attribute, or combination of attributes, that could be used successfully to identify any record in the table; this implies that there will be difficulties when it comes to inserting new data as well as manipulating data already in the table.

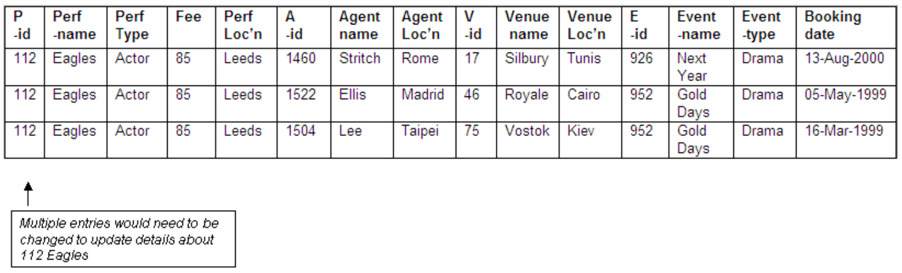
We will see that the problem of not being able to find a key for the relation in first normal form will lead us into the creation of an improved structure for representing data, so that there will be no ambiguity or loss of information.

Amendment anomalies of first normal form

There is a problem in updating values in a table in first normal form. If there is more than one entry in the relation (for example, a performer who has several bookings), any change to that individual's details must be reflected in all such entries, otherwise the data will become inconsistent.

Problems if performer changes location

What would happen if a performer moved to another location, or changed name through marriage (or both)? In first normal form, the full details for a performer are repeated every time a booking is made, and each such entry would need to be updated to reflect the change in name or location. The performer 112 Eagles already has three bookings; if there is any change to the performer details, all three entries would need to be updated. If this is not done, and a further booking is made with the updated performer details, the data in the relation will become inconsistent.

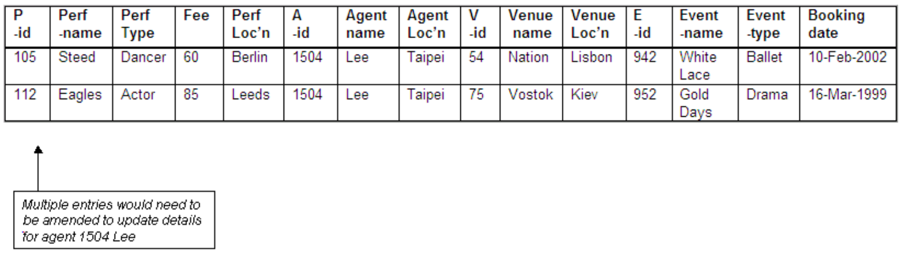


P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
112	Eagles	Actor	85	Leeds	1460	Stritch	Rome	17	Silbury	Tunis	926	Next Year	Drama	13-Aug-2000
112	Eagles	Actor	85	Leeds	1522	Ellis	Madrid	46	Royale	Cairo	952	Gold Days	Drama	05-May-1999
112	Eagles	Actor	85	Leeds	1504	Lee	Taipei	75	Vostok	Kiev	952	Gold Days	Drama	16-Mar-1999

Multiple entries would need to be changed to update details about 112 Eagles

Problems if agent changes location

The agent Lee (agent-id 1504) has made bookings for more than one performer, at more than one location, so if agent Lee were to move to another location it would be necessary to change details of the agent location in more than one place.

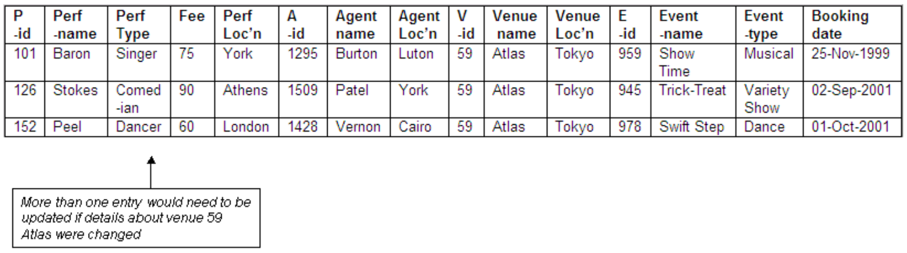


P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
105	Steed	Dancer	60	Berlin	1504	Lee	Taipei	54	Nation	Lisbon	942	White Lace	Ballet	10-Feb-2002
112	Eagles	Actor	85	Leeds	1504	Lee	Taipei	75	Vostok	Kiev	952	Gold Days	Drama	16-Mar-1999

Multiple entries would need to be amended to update details for agent 1504 Lee

Problems if agent venue details change

The venue Atlas (venue-id 59) has been booked for more than one performer, and by more than one agent; this means that there are several entries relating to this venue. Any change to the details of the venue (perhaps a change of name following a change of ownership) would need to be made to every entry that included the venue Atlas, in order to avoid inconsistencies in the data.

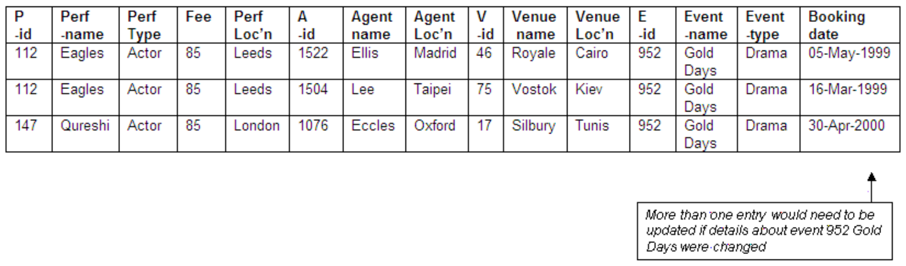


P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
101	Baron	Singer	75	York	1295	Burton	Luton	59	Atlas	Tokyo	959	Show Time	Musical	25-Nov-1999
126	Stokes	Comed -ian	90	Athens	1509	Patel	York	59	Atlas	Tokyo	945	Trick-Treat	Variety Show	02-Sep-2001
152	Peel	Dancer	60	London	1428	Vernon	Cairo	59	Atlas	Tokyo	978	Swift Step	Dance	01-Oct-2001

More than one entry would need to be updated if details about venue 59 Atlas were changed

Problems if event details change

If one of the events were to be changed, this could affect a number of tuples in the relation in first normal form. If the drama 952 Gold Days were to be rewritten to include songs, it would then need to be reclassified as a musical, and this information would need to be updated for every booking for that event. Even if the new musical production were allocated a new event-id, the change would still need to be reflected in every booking of the event.



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
112	Eagles	Actor	85	Leeds	1522	Ellis	Madrid	46	Royale	Cairo	952	Gold Days	Drama	05-May-1999
112	Eagles	Actor	85	Leeds	1504	Lee	Taipei	75	Vostok	Kiev	952	Gold Days	Drama	16-Mar-1999
147	Qureshi	Actor	85	London	1076	Eccles	Oxford	17	Silbury	Tunis	952	Gold Days	Drama	30-Apr-2000

More than one entry would need to be updated if details about event 952 Gold Days were changed

Deletion anomalies of first normal form

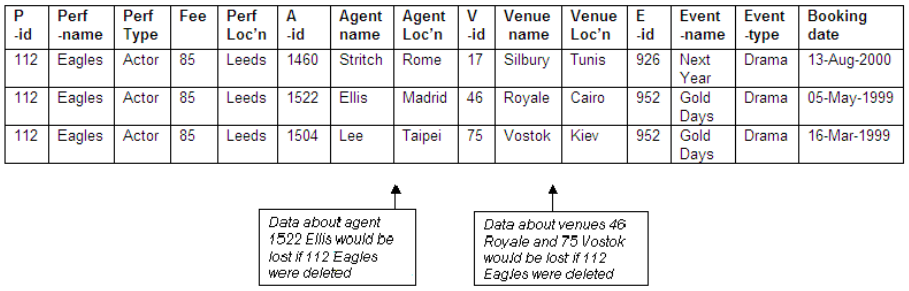
Problems if an actor is deleted

What if we were to delete the record for the actor Eagles (performer-id 112)? In this case, Eagles has three bookings, at the venues Silbury (venue-id 17), Royale (venue-id 46) and Vostok (venue-id 75). Eagles is the only performer to have a booking at venues Royale and Vostok. The agent Ellis (agent-id 1522), who made the booking for Eagles at the venue Royale, has made no other bookings. The agent Stritch (agent-id 1460), who booked Eagles into the venue Silbury, has made no other bookings, although the venue has been booked by other agents for other performers.

The events for which Eagles has been booked include two bookings for 952 Gold Days (one by agent 1522 Ellis for venue 46 Royale, the other by agent 1504 Lee for venue 75 Vostok), and a booking for event 926 Next Year (made by agent 1460 Stritch for venue 17 Silbury). As both events have also been booked for other performers, we would not lose details of the events themselves if Eagles is deleted from the relation. If Eagles had been the only performer for either one of these events, the result would have been the loss of these details when Eagles had been deleted.

If the details for performer Eagles are deleted, not only will we lose the data about agents Ellis and Stritch, but we will also lose details of the venues Royale and Vostok. The performer Eagles has three bookings, which involve two events, Gold Days (which Eagles performs twice), and Next Year. As both these events are also performed by other individuals, the deletion of data relating to Eagles means that in this case we will not lose data about these two events. If, however, Eagles had been the only performer booked for either of these events, the event details would have been lost after the deletion of the performer Eagles.

It is worth noting that if the details for Eagles are removed from the relation, all three occurrences would have to be removed; there would be problems of data integrity and consistency if some were omitted.



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
112	Eagles	Actor	85	Leeds	1460	Stritch	Rome	17	Silbury	Tunis	926	Next Year	Drama	13-Aug-2000
112	Eagles	Actor	85	Leeds	1522	Ellis	Madrid	46	Royale	Cairo	952	Gold Days	Drama	05-May-1999
112	Eagles	Actor	85	Leeds	1504	Lee	Taipei	75	Vostok	Kiev	952	Gold Days	Drama	16-Mar-1999

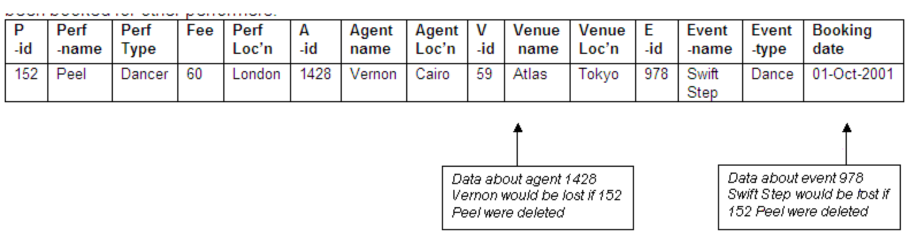
Data about agent 1522 Ellis would be lost if 112 Eagles were deleted

Data about venues 46 Royale and 75 Vostok would be lost if 112 Eagles were deleted

Problems if a performer is deleted

What if we were to delete the record for 152 Peel, the dancer? This may happen if Peel retires as a dancer.

The problem would be that not only would we remove the data related to Peel (which is our intention), but we would also unintentionally lose the data associated with the agent Vernon, as this is the only booking Vernon has made. We would also lose information stored about the event 978 Swift Step, as this is the only booking made that involves this event. Note that we would not lose details relating to the venue 59 Atlas, as this venue has also been booked for other performers.



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
152	Peel	Dancer	60	London	1428	Vernon	Cairo	59	Atlas	Tokyo	978	Swift Step	Dance	01-Oct-2001

Data about agent 1428 Vernon would be lost if 152 Peel were deleted

Data about event 978 Swift Step would be lost if 152 Peel were deleted

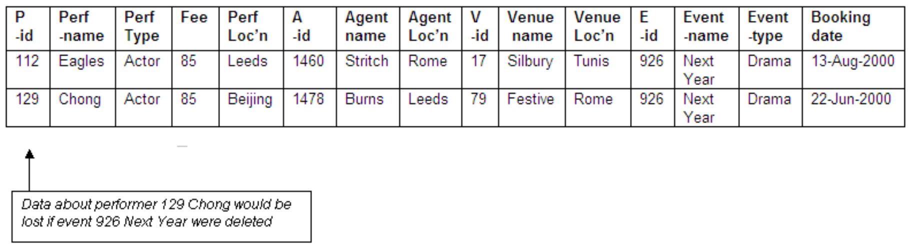
Problems if an event is deleted

What would happen if the event 926 Next Year were to be withdrawn, and all tuples containing that event deleted?

The event Next Year is involved in two bookings, one for performer 112 Eagles, and another for performer 129 Chong.

The booking for Eagles was made by agent 1504 Lee for venue 17 Silbury. Eagles has other bookings, agent Lee has made bookings for other performers, and the venue Silbury has been booked for other events, so the deletion of this tuple will not cause a loss of data about performers, agents or venues.

The other booking for event 926 New Year for performer Chong was made by agent 1478 Burns at venue 79 Festive. The agent Burns and the venue Festive are also involved in other bookings, but this was the only booking for performer Chong. If this tuple is deleted, we will lose all details concerning the performer 129 Chong.



P -id	Perf -name	Perf Type	Fee	Perf Loc'n	A -id	Agent name	Agent Loc'n	V -id	Venue name	Venue Loc'n	E -id	Event -name	Event -type	Booking date
112	Eagles	Actor	85	Leeds	1460	Stritch	Rome	17	Silbury	Tunis	926	Next Year	Drama	13-Aug-2000
129	Chong	Actor	85	Beijing	1478	Burns	Leeds	79	Festive	Rome	926	Next Year	Drama	22-Jun-2000

Data about performer 129 Chong would be lost if event 926 Next Year were deleted

These examples show that we need to store information about performers, agents, venues and events independently of each other, so that we do not risk losing data. The solution is to convert the relation in first normal form into a number of relations in second normal form.

Second normal form

The process of converting a relation from first normal form into second normal form is the identification of the primary keys, and the grouping together of attributes that relate to the key. This means that attributes that depend on different keys will now appear in a separate relation, where each attribute depends only on the key, whether directly or indirectly. The purpose of converting the relation into second normal form is to resolve many of the problems identified with first normal form.

Important

Second normal form (2NF)

For a relation to be in second normal form, all attributes must be fully functionally dependent on the primary key. Data items which are only partial dependencies (as they are not fully functionally dependent on the primary key) need to be extracted to form new relations.

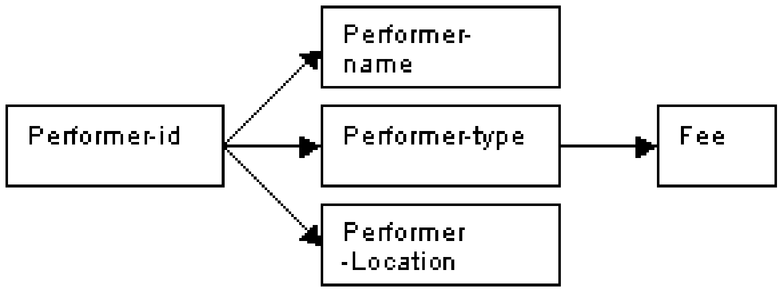
For our performer case study, the single relation in first normal form (1NF) is transformed into four relations in second normal form (working from the 1NF determinacy diagram): performers, agents, venues and bookings.

Performer details

All data relating to performers is now grouped separately from agents, venues, events and bookings. The determinacy diagram for performer details gives us a performer relation in second normal form. The primary key for the performer relation is performer-id, and the other attributes are names, performer-type, fee and location.

The creation of an independent new relation for performers has the following benefits, which resolve the problems encountered with the single relation in first normal form:

- New performers can be inserted even if they have no bookings.
- A single amendment will be sufficient to update performer details even if several bookings are involved.
- The deletion of a performer record will not result in the loss of details concerning agents, venues or events, as performers, agents, venues and events are now stored independently of each other.



Relation in second normal form: Performers

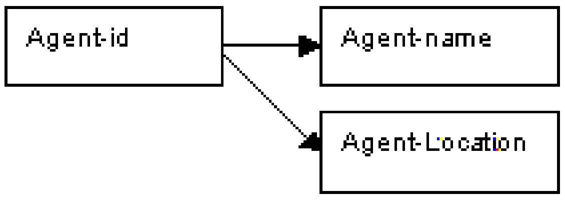
Performer-id	Performer-name	Performer-type	Fee	Performer-location
101	Baron	Singer	75	York
105	Steed	Dancer	60	Berlin
108	Jones	Actor	85	Bombay
112	Eagles	Actor	85	Leeds
118	Markov	Dancer	60	Moscow
126	Stokes	Comedian	90	Athens
129	Chong	Actor	85	Beijing
134	Brass	Singer	75	London
138	Ng	Singer	75	Penang
140	Strong	Magician	72	Rome
141	Gomez	Musician	92	Lisbon
143	Tan	Singer	75	Chicago
147	Qureshi	Actor	85	London
149	Tan	Actor	85	Taipei
150	Pointer	Magician	72	Paris
152	Peel	Dancer	80	London

Agent details

The information concerning agents is now stored separately from that of performers, venues and bookings. The determinacy diagram for agents gives us a relation for agents in second normal form. The primary key for the agents relation is agent-id, and the remaining attributes are name and location.

The new relation for agents has the following benefits, which resolve the problems encountered with the single relation in first normal form because the new relation is independent from performers, venues and bookings:

- New agents can be inserted even if they have made no bookings.
- A single change will be enough to update agent details, even if several bookings are involved.
- Agent details will now no longer be lost if a performer is deleted, as performers, agents, venues and events are now stored independently of each other.



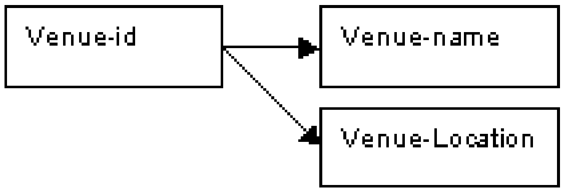
Relation in second normal form: Agents

Agent-id	Agent-name	Agent-location
1295	Burton	Luton
1435	Nunn	Boston
1504	Lee	Taipei
1682	Tsang	Beijing
1460	Stritch	Rome
1522	Ellis	Madrid
1478	Burns	Leeds
1377	Webb	Sydney
1509	Patel	York
1190	Patel	Hue
1802	Chapel	Bristol
1076	Eccles	Oxford
1409	Arkley	York
1428	Vernon	Cairo

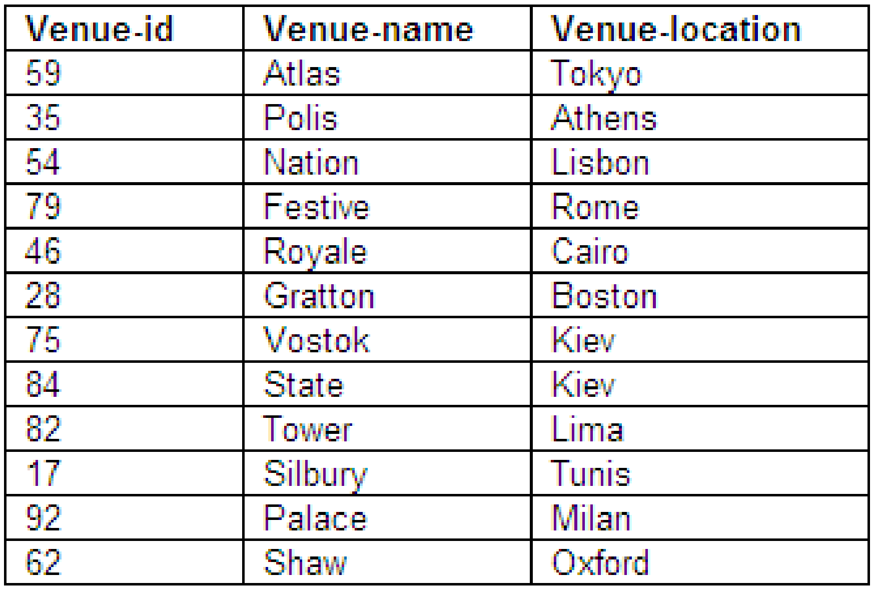
Venue details

The creation of a new relation solely to store the details of venues has the following effects, which resolve the problems identified with the single relation in first normal form:

- Details of a new venue can be inserted, whether or not it has been booked.
- If the name of the venue is changed, the alteration only needs to be made once, in the venue relation, not for every booking of that venue.
- If details of a performer are deleted, and the performer had the only booking at a particular venue, details of the venue will not be lost.



Relation in second normal form: Venues

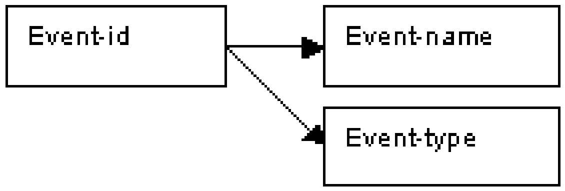


Venue-id	Venue-name	Venue-location
59	Atlas	Tokyo
35	Polis	Athens
54	Nation	Lisbon
79	Festive	Rome
46	Royale	Cairo
28	Gratton	Boston
75	Vostok	Kiev
84	State	Kiev
82	Tower	Lima
17	Silbury	Tunis
92	Palace	Milan
62	Shaw	Oxford

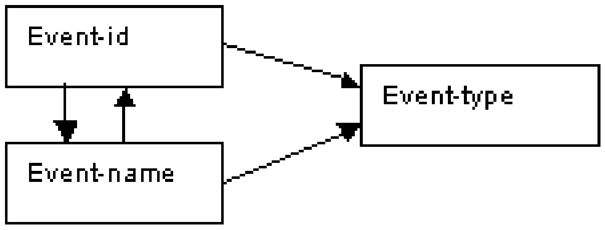
Event details

- A new relation is created to hold details of individual events.
- Details of a new event can be inserted, whether or not it has been booked.
- If the name of the event is changed, the alteration only needs to be made once, in the event relation, not for every booking of that event.
- If details of a performer are deleted, and the performer had the only booking of a particular event, details of the event will not be lost.

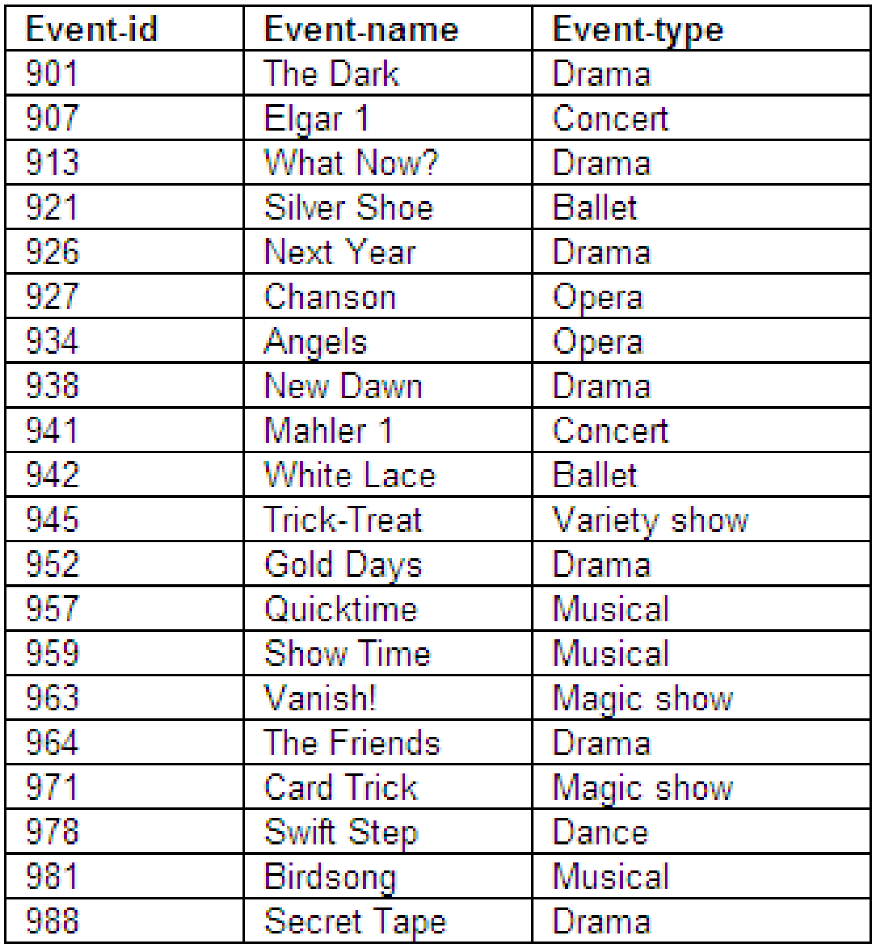
The determinacy diagram could be represented as follows:



An alternative representation of the determinacy diagram illustrates that the attribute event-name is also a determinant, although it is not a candidate key:



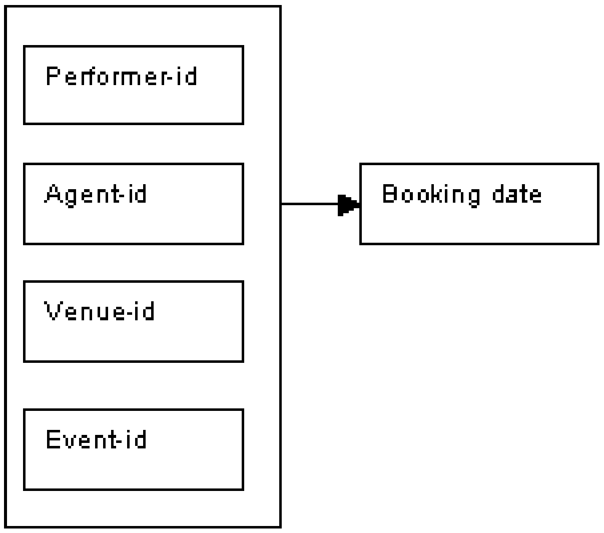
Relation in second normal form: Events



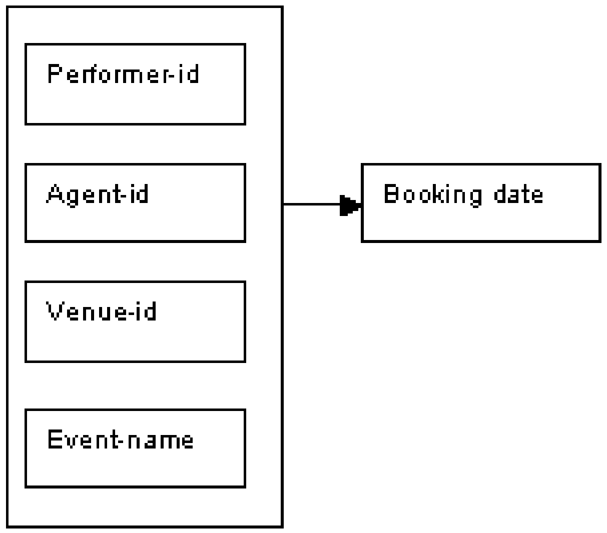
Event-id	Event-name	Event-type
901	The Dark	Drama
907	Elgar 1	Concert
913	What Now?	Drama
921	Silver Shoe	Ballet
926	Next Year	Drama
927	Chanson	Opera
934	Angels	Opera
938	New Dawn	Drama
941	Mahler 1	Concert
942	White Lace	Ballet
945	Trick-Treat	Variety show
952	Gold Days	Drama
957	Quicktime	Musical
959	Show Time	Musical
963	Vanish!	Magic show
964	The Friends	Drama
971	Card Trick	Magic show
978	Swift Step	Dance
981	Birdsong	Musical
988	Secret Tape	Drama

Booking details

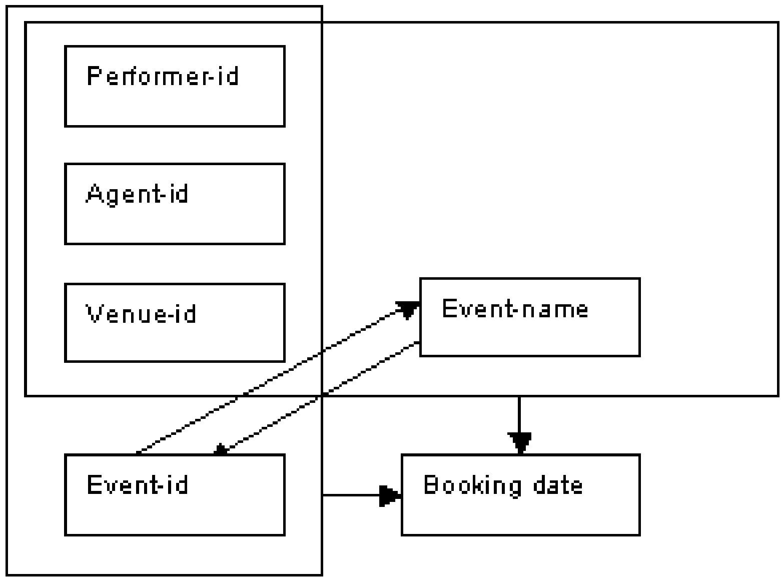
Every time a booking is made, the details are recorded in the relation called Bookings. There is no need to store all the details of the performer, agent, venue and event for each booking that is made, as this information can be acquired from the relevant relation for performers, agents, venues and event. The information that is needed for the Booking relation is the performer-id, agent-id, venue-id and event-id (these four attributes together form the key for this relation), and the booking date.



Another possible key for the Booking relation involves the attributes performer-id, agent-id, venue-id and event-name; as three of the four attributes in this key are the same as the first key described for this relation, we have an example of overlapping keys. Note that the overlapping keys are not resolved in the transformation from first normal form to second normal form, as event-id and event-name are part of each key. Conversion from first to second normal form extracts all non-key attributes which are only partially dependent on the key, and as such event-id and event-name remain as they are part of the key.



The determinacy diagram below shows the overlapping keys for the Bookings relation, and also illustrates the dependencies between the attributes event-id and event-name:



The details of the Bookings relation are shown below.

Relation in second normal form: Bookings

Performer -id	Agent -id	Venue -id	Event -id	Event -name	Booking date
101	1295	59	959	Show Time	25-Nov-1999
105	1435	35	921	Silver Shoe	07-Jan-2002
105	1504	54	942	White Lace	10-Feb-2002
108	1682	79	901	The Dark	29-Jul-2003
112	1460	17	926	Next Year	13-Aug-2000
112	1522	46	952	Gold Days	05-May-1999
112	1504	75	952	Gold Days	16-Mar-1999
126	1509	59	945	Trick-Treat	02-Sep-2001
129	1478	79	926	Next Year	22-Jun-2000
134	1504	28	981	Birdsong	18-Sep-2001
138	1509	84	957	Quicktime	18-Aug-1999
140	1478	17	963	Vanish!	18-Aug-1999
141	1478	84	941	Mahler 1	21-Jul-2000
143	1504	79	927	Chanson	21-Nov-2002
147	1076	17	952	Gold Days	30-Apr-2000
147	1409	79	988	Secret Tape	17-Apr-2000
152	1428	59	978	Swift Step	01-Oct-2001

Insertion anomalies of second normal form

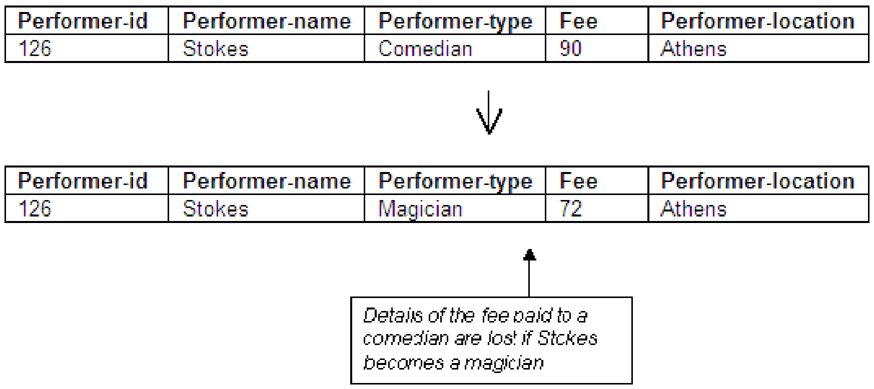
We cannot enter a fee for a type of performer unless there is a performer of that type already present in the relation in second normal form. The reason for this is that if there is no existing performer of that type, there will be no performer-id value available as a key. If we want to add that acrobats are paid 65 (in whatever currency), we cannot do so unless we are able to enter complete details for a specific individual. Note that this performer would not have to have a booking, but there must be at least one person associated with a performer-type before.

Performer-id	Performer-name	Performer-type	Fee	Performer-location
		Acrobat	65	

Details of the fee paid to an acrobat cannot be entered unless there is a performer

Amendment anomalies of second normal form

If performer Stokes (performer-id 126), who is the only comedian in the relation, retrains and changes career to become a magician, we will then lose the information that comedians are paid a fee of 90 (in whatever currency is used). Stokes will then be paid 72, which is the fee for all magicians.



Performer-id	Performer-name	Performer-type	Fee	Performer-location
126	Stokes	Comedian	90	Athens

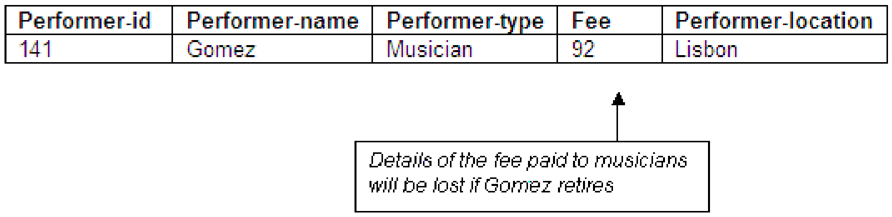
Performer-id	Performer-name	Performer-type	Fee	Performer-location
126	Stokes	Magician	72	Athens

Details of the fee paid to a comedian are lost if Stokes becomes a magician

We would also find an amendment anomaly if the fee paid to a particular type of performer changed. If all singers were granted a new rate, all tuples relating to singers would need to be updated, otherwise the data would become inconsistent.

Deletion anomalies of second normal form

If Gomez (performer-id 141), the only musician in the relation, decides to retire, we will lose the information regarding the fee of 92 paid to musicians.



Performer-id	Performer-name	Performer-type	Fee	Performer-location
141	Gomez	Musician	92	Lisbon

Details of the fee paid to musicians will be lost if Gomez retires

All these anomalies are caused by the fee paid to the performer being dependent on the performer-type, and not directly on the primary key performer-id. This indirect, or transitive, dependency can be resolved by transforming the relations in second normal form into third normal form, by extracting the attributes involved in the indirect dependency into a separate new relation.

Third normal form

The reason for converting a table from second normal form into third normal form is to ensure that data depends directly on the primary key, and not through some other relationship with another attribute (known as an indirect, or transitive, dependency).

Important

Third normal form (3NF)

A relation is in third normal form if there are no indirect (or transitive) dependencies between the attributes; for a relation to be in third normal form, all attributes must be directly dependent on the primary key.

An indirect dependency is resolved by creating a new relation for each entity; these new relations contain the transitively dependent attributes together with the primary key.

The conversion of a relation into third normal form will resolve anomalies identified in second normal form.

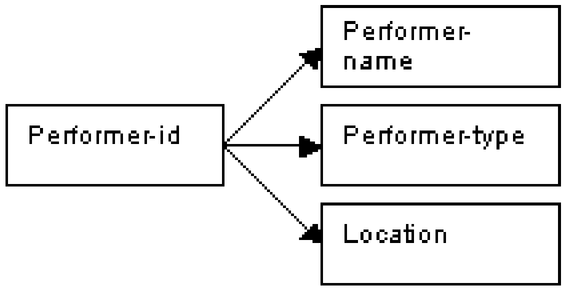
We now have six relations in third normal form: Performers, Fees, Agents, Venues, Events and Bookings.

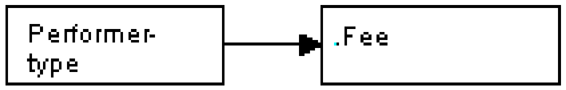
Performer details

As before, the name and location of each performer depends on the performer-id. We noticed in second normal form that there were problems associated with having the fee contained within the performer relation, as the value of the fee depended on the performer-type and not on performer-id, demonstrating a transitive dependency.

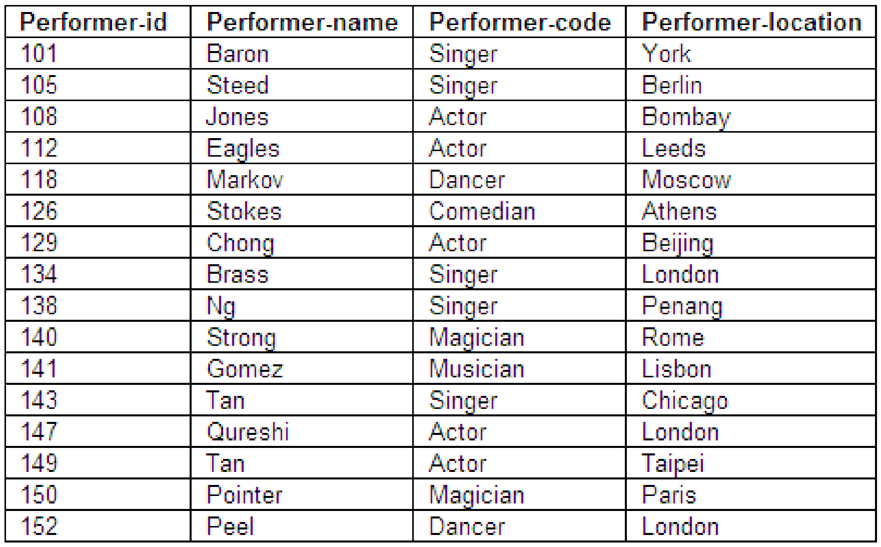
One solution would be to create a new relation with performer-type as the key, and the fee as the other attribute; performer-type would also remain in the relation Performers, but the fee would be removed.

The relations for Performer and Fees follow the determinacy diagrams below.





Relation in third normal form: Performers



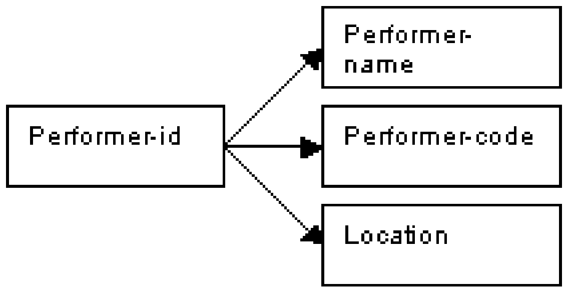
Performer-id	Performer-name	Performer-code	Performer-location
101	Baron	Singer	York
105	Steed	Singer	Berlin
108	Jones	Actor	Bombay
112	Eagles	Actor	Leeds
118	Markov	Dancer	Moscow
126	Stokes	Comedian	Athens
129	Chong	Actor	Beijing
134	Brass	Singer	London
138	Ng	Singer	Penang
140	Strong	Magician	Rome
141	Gomez	Musician	Lisbon
143	Tan	Singer	Chicago
147	Qureshi	Actor	London
149	Tan	Actor	Taipei
150	Pointer	Magician	Paris
152	Peel	Dancer	London

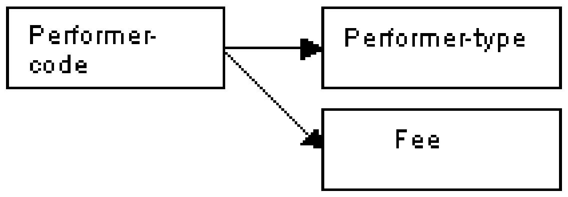
Relation in third normal form: Fees

Performer-type	Fee
Singer	75
Dancer	60
Actor	85
Comedian	90
Magician	84
Musician	92

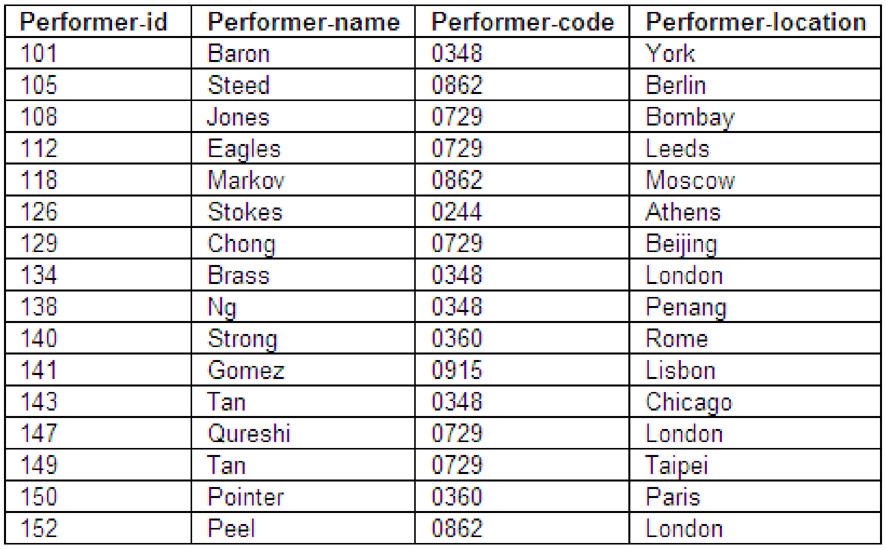
A possible problem with this approach is the format of data entry of new performers; if "ACROBAT", "Acrobat" or "acrobat" are entered, they might not be recognised as the same performer-type. In addition, if an error is made and "arcobat" is entered, this may not be recognised. To deal with this problem, we have used a code for performer-type in the Performer relation. This code is then used as the key in the Fees relation, and the other attributes are performer-type and the fee, both of which depend on the performer-code as primary key. (We could have introduced the performer-code into the table of un-normalised data.)

The relations for Performer and Fees follow the determinacy diagrams below.



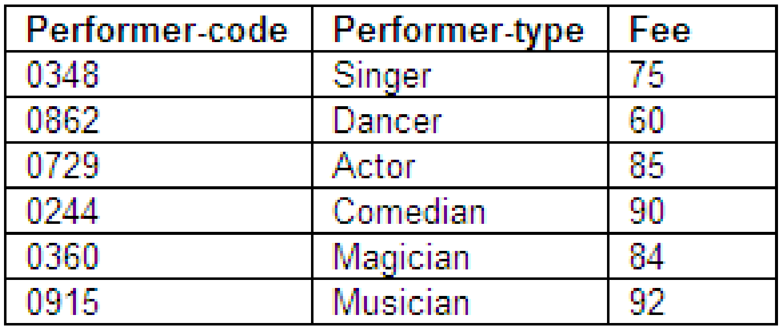


Relation in third normal form: Performers



Performer-id	Performer-name	Performer-code	Performer-location
101	Baron	0348	York
105	Steed	0862	Berlin
108	Jones	0729	Bombay
112	Eagles	0729	Leeds
118	Markov	0862	Moscow
126	Stokes	0244	Athens
129	Chong	0729	Beijing
134	Brass	0348	London
138	Ng	0348	Penang
140	Strong	0360	Rome
141	Gomez	0915	Lisbon
143	Tan	0348	Chicago
147	Qureshi	0729	London
149	Tan	0729	Taipei
150	Pointer	0360	Paris
152	Peel	0862	London

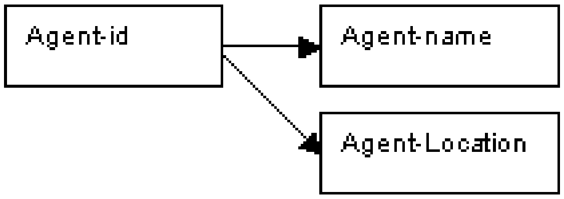
Relation in third normal form: Fees



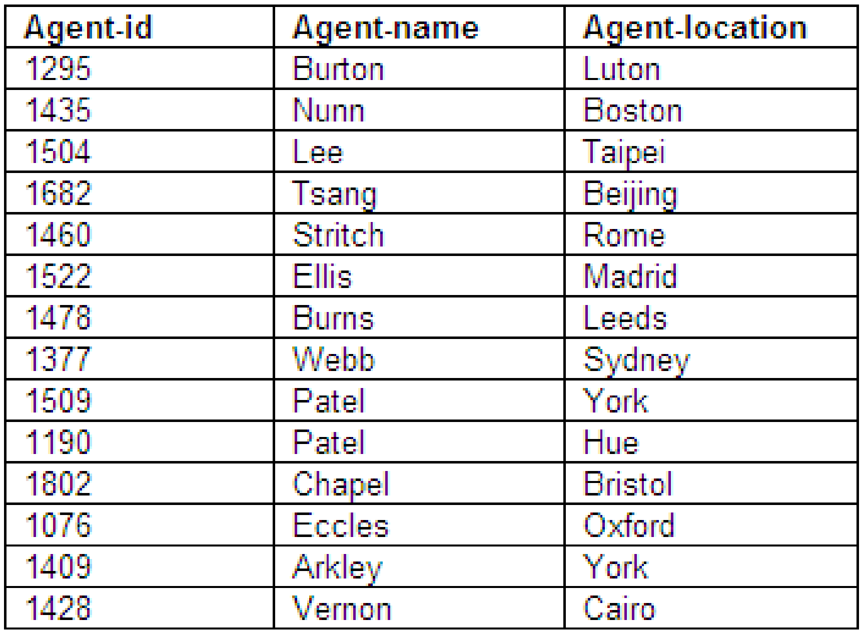
Performer-code	Performer-type	Fee
0348	Singer	75
0862	Dancer	60
0729	Actor	85
0244	Comedian	90
0360	Magician	84
0915	Musician	92

Agent details

There is no change to the determinacy diagram for Agents, as this is already in third normal form (there were no transitive dependencies). The relation follows the determinacy diagram below.



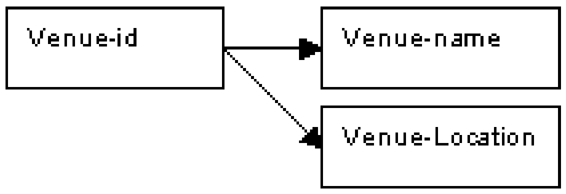
Relation in third normal form: Agents



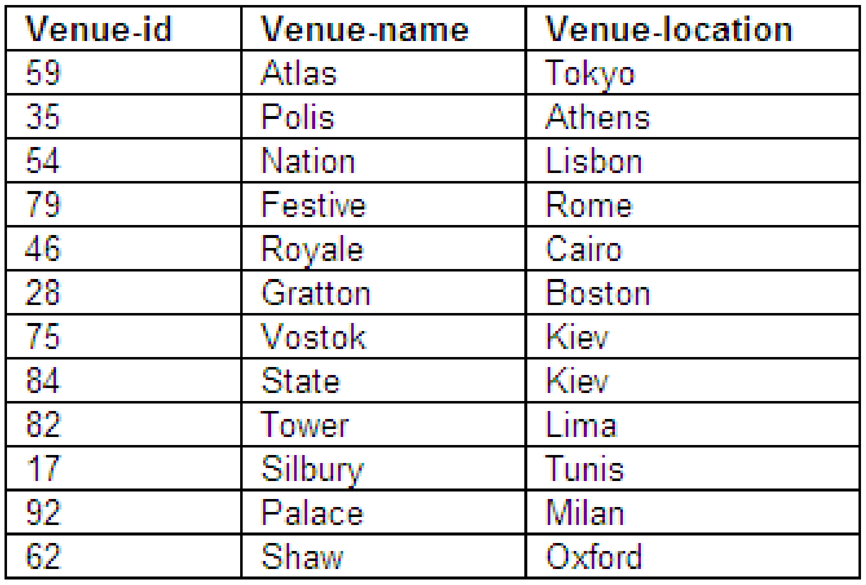
Agent-id	Agent-name	Agent-location
1295	Burton	Luton
1435	Nunn	Boston
1504	Lee	Taipei
1682	Tsang	Beijing
1460	Stritch	Rome
1522	Ellis	Madrid
1478	Burns	Leeds
1377	Webb	Sydney
1509	Patel	York
1190	Patel	Hue
1802	Chapel	Bristol
1076	Eccles	Oxford
1409	Arkley	York
1428	Vernon	Cairo

Venue details

The data on Venues is already in third normal form as there were no transitive dependencies; there are therefore no changes to the determinacy diagram shown below, and the relation which follows.



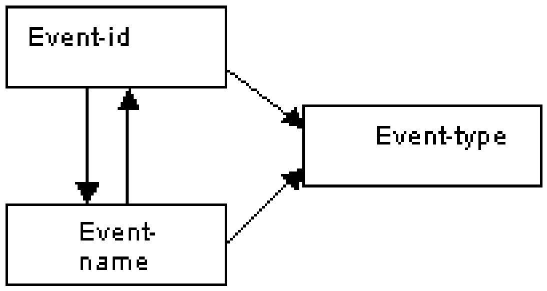
Relation in third normal form: Venues



Venue-id	Venue-name	Venue-location
59	Atlas	Tokyo
35	Polis	Athens
54	Nation	Lisbon
79	Festive	Rome
46	Royale	Cairo
28	Gratton	Boston
75	Vostok	Kiev
84	State	Kiev
82	Tower	Lima
17	Silbury	Tunis
92	Palace	Milan
62	Shaw	Oxford

Event details

The Events relation is already in third normal form as there are no transitive dependencies. There is the special relationship that exists between the attributes event-id and event-name, which does not present a problem within the Events relation itself, but creates difficulties in the Bookings relation because of the overlapping key which results.



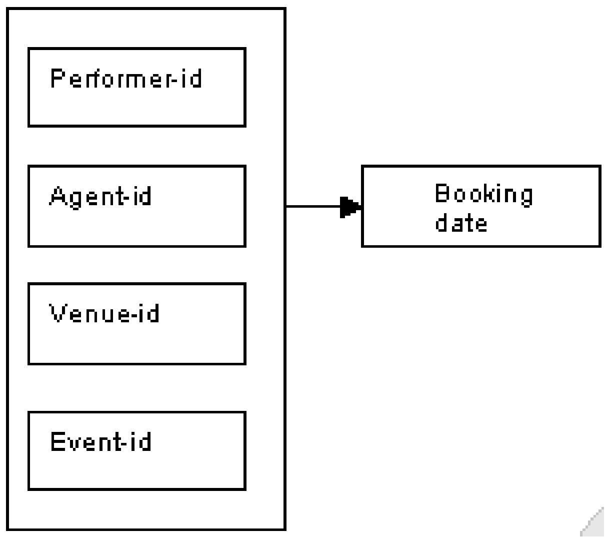
Relation in third normal form: Events

Event-id	Event-name	Event-type
901	The Dark	Drama
907	Elgar 1	Concert
913	What Now?	Drama
921	Silver Shoe	Ballet
926	Next Year	Drama
927	Chanson	Opera
934	Angels	Opera
938	New Dawn	Drama
941	Mahler 1	Concert
942	White Lace	Ballet
945	Trick-Treat	Variety show
952	Gold Days	Drama
957	Quicktime	Musical
959	Show Time	Musical
963	Vanish!	Magic show
964	The Friends	Drama
971	Card Trick	Magic show
978	Swift Step	Dance
981	Birdsong	Musical
988	Secret Tape	Drama

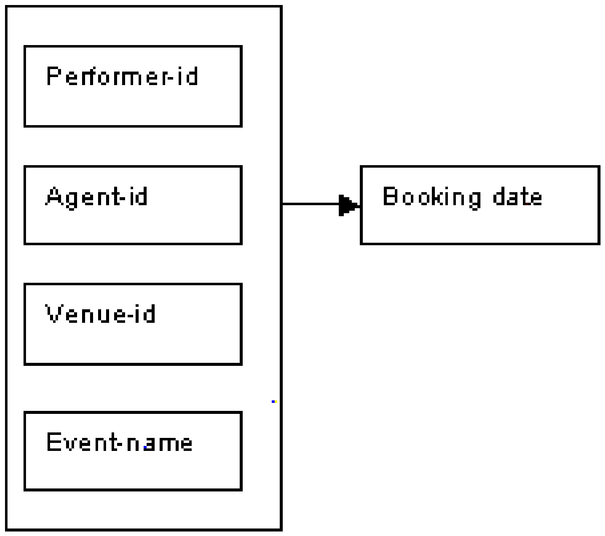
Booking details

The relation Bookings, with its composite determinants of performer-id, agent-id, venue-id and event-id, or performer-id, agent-id, venue-id and event-name, is already in third normal form as there are no transitive dependencies. The determinacy diagrams and the associated relation are illustrated below.

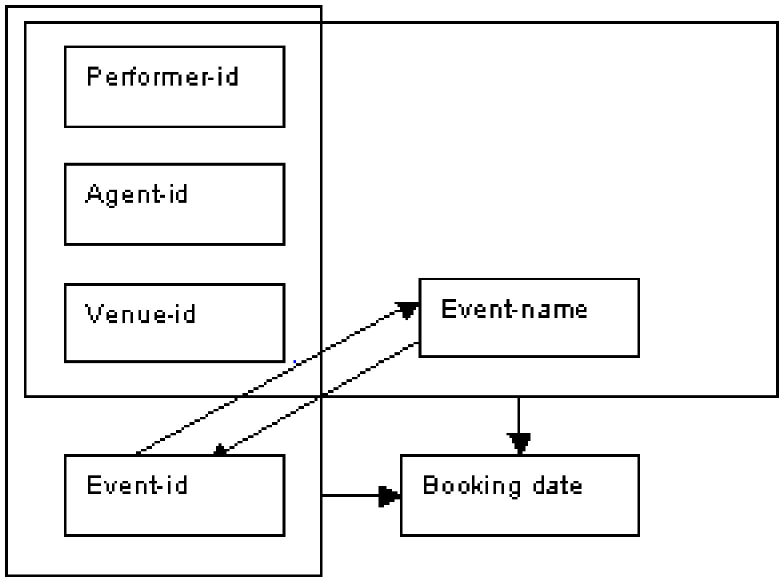
This determinacy diagram illustrates the combination of performer-id, agent-id, venue-id and event-id used as the determinant for the Bookings relation:



The next determinacy diagram shows the choice of performer-id, agent-id, venue-id and event-name as the determinants for the Bookings relation.

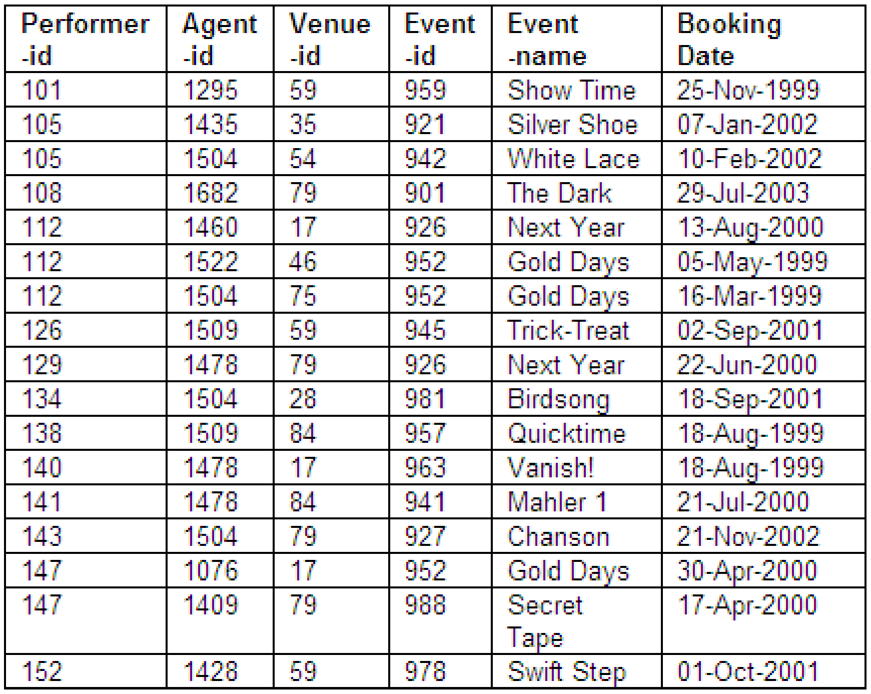


The determinacy diagram below combines the previous two determinacy diagrams to show the overlapping keys for the Bookings relation, and illustrates the dependencies between the attributes event-id and event-name.



The details of the Bookings relation are shown below.

Relation in third normal form: Bookings



Performer -id	Agent -id	Venue -id	Event -id	Event -name	Booking Date
101	1295	59	959	Show Time	25-Nov-1999
105	1435	35	921	Silver Shoe	07-Jan-2002
105	1504	54	942	White Lace	10-Feb-2002
108	1682	79	901	The Dark	29-Jul-2003
112	1460	17	926	Next Year	13-Aug-2000
112	1522	46	952	Gold Days	05-May-1999
112	1504	75	952	Gold Days	16-Mar-1999
126	1509	59	945	Trick-Treat	02-Sep-2001
129	1478	79	926	Next Year	22-Jun-2000
134	1504	28	981	Birdsong	18-Sep-2001
138	1509	84	957	Quicktime	18-Aug-1999
140	1478	17	963	Vanish!	18-Aug-1999
141	1478	84	941	Mahler 1	21-Jul-2000
143	1504	79	927	Chanson	21-Nov-2002
147	1076	17	952	Gold Days	30-Apr-2000
147	1409	79	988	Secret Tape	17-Apr-2000
152	1428	59	978	Swift Step	01-Oct-2001

Summary of the first three normal forms

We have seen how the original set of data items has been transformed through the initial process of identifying dependencies between data items, the formulation of successively higher normal-form collections of relations, each of which has represented an increasingly flexible design. The steps used to derive each successive normal form are summarised below:

- Identify data items which are the determinants of other data items, and through the removal of any repeating groups, form the data items into an initial first normal form relation.
- Identify any attributes that are not included in the primary key of the relation, which are not dependent on all of the primary key (this is sometimes called 'removing part-key dependencies'). It is also worth bearing in mind that this step does not arise for relations that have a single-attribute primary key.

- Identify any attributes that are not directly determined by the key (this is sometimes called 'removing transitive dependencies').

We shall see in a later chapter on database design that there is further work that can be done to normalise sets of relations, and alternative approaches to reaching third normal form (3NF). However, 3NF represents a point where we have gained a significant degree of flexibility in the design of a database application, and it is a point at which normalisation of many applications is considered to be complete.

Review questions

One of the biggest challenges when designing a database system is to obtain a correct and complete set of requirements from the prospective users of the system. Modern development methods place a strong emphasis on the need to develop prototypes of the system, so that these can be demonstrated to future users to clarify that what is being developed is what is actually required. Information gathering about the way in which an application is to work is a vital process which requires much attention to detail. This question provides an exercise in formulating the questions to be used in a data-analysis scenario. The importance of preparing for discussions about system requirements cannot be over-emphasised, as users often are short of time, have other commitments, and require guidance in describing the information required for a design.

Review question 1

Imagine that you have been commissioned by the owner of a small business to develop a database of the projects he is running. You know that the database is required to store details of the following:

1. The projects being undertaken, including expected start and finish dates.
2. Tasks required to complete each project.
3. Contract staff recruited to assist with the projects.
4. The budgets for projects.
5. The resources being used in projects and their costs.

Design a questionnaire you might use to assist you in obtaining the details of dependencies between data items when discussing the database with the business owner.

Review question 2

Given below is a possible series of answers to the questions in the previous question. Given these responses, formulate the data items mentioned into a first normal form relation.

1. How is each project identified?

   Each project is to be allocated a unique project number. This number need only be two digits long, as there will never be more that 99 projects to be stored at one time.

2. Is it required to store both expected and actual completed start and finish dates for projects?

   Yes, all four data items are required, and the same four data items are required for tasks as well.

3. How are tasks identified?

   They also have a unique task number, which again can be safely limited to two digits. So each task is identified by the combination of the project number of the project within which it occurs, and its own task number.

4. Do projects have many tasks?

   Yes, each project typically consists of about 10 tasks.

5. Can a task be split between more than one project?

   No, a task is always considered to take place within one project only.

6. Are employees assigned to projects, or to specific tasks within projects? How many tasks can an employee work on at one time?

   Employees are assigned to specific tasks within projects, so each employee can work on a number of tasks at one time. Furthermore, each task has an employee allocated to it who is specifically responsible for its successful completion. Each project has a project leader responsible for that project's successful completion.

7. What is required to be stored about contract staff pay?

   Each staff member is paid at a monthly rate, that rate being determined entirely by the highest qualification held by the staff member. We simply need to record the appropriate qualification for each staff member, and the monthly rate at which they are paid, plus the start and end dates of their current contact.

Review question 3

Remove any part-key dependencies from the relation produced in question 2 to produce a set of second normal form relations.

Review question 4

From the second normal form design in the previous question, produce a set of third normal form relations, by removing any indirect or transitive dependencies.

Review question 5

Explain the role of determinacy diagrams in database application development.

Review question 6

What is a repeating group? Why is it necessary to remove repeating groups in Relational database design?

Review question 7

Explain the term 'part-key dependency', and its role in normalisation.

Review question 8

What is the difference between second and third normal form relations?

Discussion topic

As mentioned at the start of the review questions, the process of eliciting information about the requirements of computer applications is an extremely important and potentially difficult one. Among the techniques that are commonly used to capture the requirements of users and other stakeholders in the system are:

- Interviews, which vary in different organisations and between individuals in the amount of planning and pre-determined questions
- Questionnaire surveys, in the following formats: written, e-mail or web-based
- Brainstorming
- Direct observation of users carrying out tasks

All of these techniques and more can play a useful role in capturing requirements, and each technique has particular strengths and weaknesses. You are encouraged to discuss with other students the strengths and weaknesses you consider each of the techniques listed above have in obtaining accurate and comprehensive information about the requirements for a new computer application. You should include in the discussion any experiences you have had yourself of good or bad practice in the process of requirements capture.

Application and further work

You are encouraged to consider the strengths and weaknesses of the application developed in the review questions.

Firstly, identify the additional flexibility gained by each successive stage of the normalisation process. That is, clarify the sorts of data manipulation that can be carried out in the more normalised versions of the design, compared to the un-normalised design.

Secondly, consider to what extent this extra flexibility is likely to be useful to the business owner, and whether it is worth the overhead of managing the additional tables.