

ACCA7002 Database Systems

Tutorial Tasks

Tutorial 5: Normalisation

The aim of this exercise is to help you practice database normalisation techniques.

Task 5.1¹: A company called *FastCabs* provides a taxi service to clients.

The table shown below displays some details of client bookings for taxis. Assume that a taxi driver is assigned to a single taxi, but a taxi can be assigned to one or more drivers.

Produce a set of 3NF relations (tables) from the table shown below.

DriverID	DriverName	TaxiID	JobID	JobDateTime	ClientID	ClientName	PickupAddress
D1	Joe Bull	T1	1	25/07/2019 10:00	C1	Anne Woo	1 Kings Road, Swansea
			2	29/07/2019 10:00			
D2	Tom Win	T2	3	30/07/2019 11:00	C1	Anne Woo	1 Kings Road, Swansea
D3	Jim Jones	T3	4	2/08/2019 13:00	C2	Mark Tin	10 High St, Swansea
D4	Stevin Win	T1	5	2/08/2019 13:00	C3	John Seal	22 Red Road, Swansea
D2	Tom Win	T2	6	25/08/2019 10:00	C4	Karen Bow	17 High St, Swansea

¹ Adapted from Database Systems: A Practical Approach to Design, Implementation and Management by Thomas Connolly & Carolyn Begg

Solution for Task 5.1

UNF to ONF

1NF: A relation in which the intersection of each row and column *contains one and only one value*. That is, there are no repeating groups.

Steps

- Identify Primary Key (PK): JobID
- Eliminate repeating groups either by flattening the table or creating a new table with the repeating groups.

Method 1: Flatten the Table

Simply fill down values from cell above in blank cells

DriverID	DriverName	TaxiID	JobID	JobDateTime	ClientID	ClientName	PickupAddress
D1	Joe Bull	T1	1	25/07/2019 10:00	C1	Anne Woo	1 Kings Road, Swansea
D1	Joe Bull	T1	2	29/07/2019 10:00	C1	Anne Woo	1 Kings Road, Swansea
D2	Tom Win	T2	3	30/07/2019 11:00	C1	Anne Woo	1 Kings Road, Swansea
D3	Jim Jones	T3	4	2/08/2019 13:00	C2	Mark Tin	10 High St, Swansea
D4	Stevin Win	T1	5	2/08/2019 13:00	C3	John Seal	22 Red Road, Swansea
D2	Tom Win	T2	6	25/08/2019 10:00	C4	Karen Bow	17 High St, Swansea

Method 2: Create a Separate Table

Create a separate table with the repeating groups + the PK (JobID)

Method 2 does not work well with this table data because using the PK (JobID) produces 6 rows which is the number of rows in the table.

1NF to 2NF

2NF: A relation that is in 1NF **and** every non-primary key attribute is fully-functionally dependent on the PK.

Steps

- Identify the PK: JobID
- Remove partial dependencies

The PK of this table is the JobID, which is made up of only one column **so there are no partial dependencies**, so this table is already in 2NF.

2NF to 3NF

3NF: A relation that is in both 1NF and 2NF **and** in which no non-primary key is transitively dependent on the primary key. That is identify relationships of the form $A \rightarrow B$ and $B \rightarrow C$.

Steps

- Identify the PK: JobID

¹ Adapted from Database Systems: A Practical Approach to Design, Implementation and Management by Thomas Connolly & Carolyn Begg

- Identify relationships of the form JobID $\rightarrow$ B and B $\rightarrow$ C

Table Columns

DriverID	DriverName	TaxiID	JobID	JobDateTime	ClientID	ClientName	PickupAddress
...	...	...	...	...	...	...	...

The functional dependencies are as follows:

- JobID $\rightarrow$ DriverID, DriverName, TaxiID, JobDateTime, ClientID, ClientName, PickupAddress

We already know that the JobID is the PK for this table, if you know the JobID, you can determine a combination of all the other columns

The other functional dependencies – make these new tables.

- ClientID $\rightarrow$ ClientName

If you know the ClientID, you can determine the ClientName.

Note: You may have been tempted to include the PickupAddress as part of this dependency, but the clue is in the name. It is just coincidental that Client C1 who used the Taxi service multiple times was picked up from the same address (let's assume her house). Next time around, Client C1 (Anne Woo) may have been asked to be picked up from the hospital, her office, a friend's place, etc. and then the PickupAddress would be different so it needs to go with the other Job Details.

- DriverID $\rightarrow$ DriverName, TaxiID

If you know the DriverID, you can determine the DriverName and TaxiID (as a taxi driver is assigned a single taxi)

There are no other functional dependencies. TaxiID does **not** determine any other columns because you can have more than one driver driving the same Taxi (T1 is driven by D1 and D4).

Make the other functional dependencies 2 new tables.

Client Table

ClientID $\rightarrow$ ClientName

Client (ClientID, ClientName)

ClientID (PK)	ClientName
C1	Anne Woo
C2	Mark Tin
C3	John Seal
C4	Karen Bow

Driver Table

DriverID $\rightarrow$ DriverName, TaxiID

Driver (DriverID, DriverName, TaxiID)

¹ Adapted from Database Systems: A Practical Approach to Design, Implementation and Management by Thomas Connolly & Carolyn Begg

DriverID (PK)	DriverName	TaxiID
D1	Joe Bull	T1
D2	Tom Win	T2
D3	Jim Jones	T3
D4	Stevin Win	T1

Job Table

For the original table (Delete all the columns on the right-side of the other functional dependencies, leaving only the PKs from the new tables as Foreign Keys (FKs).

Job (JobID, DriverID, DriverName, TaxiID, JobDateTime, ClientID, ClientName, PickupAddress)

JobID (PK)	DriverID (FK)	JobDateTime	ClientID (FK)	PickupAddress
1	D1	25/07/2019 10:00	C1	1 Kings Road, Swansea
2	D1	29/07/2019 10:00	C1	1 Kings Road, Swansea
3	D2	30/07/2019 11:00	C1	1 Kings Road, Swansea
4	D3	2/08/2019 13:00	C2	10 High St, Swansea
5	D4	2/08/2019 13:00	C3	22 Red Road, Swansea
6	D2	25/08/2019 10:00	C4	17 High St, Swansea

¹ Adapted from Database Systems: A Practical Approach to Design, Implementation and Management by Thomas Connolly & Carolyn Begg

Task 5.2¹: Students can lease university flats and some of the details of leases held by students for places in university flats are shown in the table structure below. A room flat number (RoomFlatNo) uniquely identifies each single room in all flats and is used when leasing a room to a student.

- Identify the functional dependencies that exist between the columns of the table and identify the primary key and any alternate key(s) (if present) for the table.
- Produce a set of 3NF tables from this table.

Student Accommodation Leases Table

StudentAccommodationLease(LeaseNo, StudentNo, FlatRoomNo, FirstName, Surname, StartDate, FinishDate, FlatNo, FlatAddress)

StudentAccommodationLease
LeaseNo
StudentNo
FlatRoomNo
FirstName
Surname
StartDate
FinishDate
FlatNo
FlatAddress

¹ Adapted from Database Systems: A Practical Approach to Design, Implementation and Management by Thomas Connolly & Carolyn Begg

Solution for Task 5.2

StudentAccommodationLease (LeaseNo, StudentNo, FlatRoomNo, FirstName, Surname, StartDate, FinishDate, FlatNo, FlatAddress)

UNF to ONF

1NF: A relation in which the intersection of each row and column *contains one and only one value*. That is, there are no repeating groups.

Steps

- Identify Primary Key (PK): LeaseNo
Note: no two lease agreements will have the same lease number.
- Eliminate repeating groups either by flattening the table or creating a new table with the repeating groups.

This table has **no repeating groups** (no data is shown). **As such, the table is already in 1NF.**

1NF to 2NF

2NF: A relation that is in 1NF **and** every non-primary key attribute is fully-functionally dependent on the PK.

Steps

- Identify the PK: LeaseNo
- Remove partial dependencies

The PK of this table is the LeaseNo, which is made up of only one column **so there are no partial dependencies**, so this table is already in 2NF.

2NF to 3NF

3NF: A relation that is in both 1NF and 2NF **and** in which no non-primary key is transitively dependent on the primary key. That is identify relationships of the form $A \rightarrow B$ and $B \rightarrow C$.

Steps

- Identify the PK: LeaseNo
- Identify relationships of the form $\text{LeaseNo} \rightarrow B$ and $B \rightarrow C$

Table Columns

StudentAccommodationLease (LeaseNo, StudentNo, FlatRoomNo, FirstName, Surname, StartDate, FinishDate, FlatNo, FlatAddress)

Lease No	Student No	FlatRoom No	FirstName	Surname	Start Date	Finish Date	FlatNo	FlatAddress
...	...	...	...	...	...	...		...

First my apologies: FlatNo is actually an ID value rather than Flat Number, say 1, in a block of 10 flats.

¹ Adapted from Database Systems: A Practical Approach to Design, Implementation and Management by Thomas Connolly & Carolyn Begg

The functional dependencies are as follows:

- LeaseNo → StudentNo, FlatRoomNo, FirstName, Surname, StartDate, FinishDate, FlatNo, FlatAddress

We already know that the LeaseNo is the PK for this table, if you know the LeaseNo, you can determine a combination of all the other columns

The other functional dependencies

- StudentNo → FirstName, Surname

If you know the StudentNo, you can determine the FirstName and Surname.

- FlatNo → FlatAddress

If you know the FlatNo (it is an ID value), you can determine the FlatAddress.

- FlatNo, FlatRoomNo → FlatAddress

Note: This is a tricky one.

- If you know the FlatNo (e.g. F0001), you cannot determine the FlatRoomNo as it may have more than one room (FlatRoomNo 1 – 5). Similarly, if you know the FlatRoomNo (e.g. Flat Room No 5), you cannot determine the Flat No as two different flats may have a room with FlatRoomNo (5).
- In other words, you cannot determine one without the other and hence the composite key.

Make the other functional tables new tables

Student Table

StudentNo → FirstName, Surname

Student (StudentNo, FirstName, Surname)

StudentNo (PK)	FirstName	Surname
...	...	...

Flat Table

FlatNo → FlatAddress

Flat (FlatNo, FlatAddress)

FlatNo (PK)	FlatAddress
...	...

FlatRoom Table

FlatNo, FlatRoomNo → FlatAddress

A table with these three columns, however, is **not** in 3NF.

Note:

¹ Adapted from Database Systems: A Practical Approach to Design, Implementation and Management by Thomas Connolly & Carolyn Begg

- Every time you create a new table it is good practice to check whether the new tables being created are in 2NF and 3NF regardless of what stage of the normalisation process you are (normalisation is recursive), **especially for tables with a composite key (PK is made up of more than one column).**
- The combination of Flat, FlatRoomNo as the PK in this functional dependency (FlatNo, FlatRoomNo → FlatAddress) contains the partial dependency already identified above FlatNo → FlatAddress **which is already a new table.**
- According to 2NF to 3NF, you remove any partial dependencies and put them in their own table leaving only the PK of the new table (FlatNo) as a FK.
(FlatNo, FlatRoomNo → FlatAddress)
- Thus this table is going to be left with only the composite PK and no other fields. It is actually an example of a table created to split up many-to-many relationships.
 - 1 Flat may have 1...* rooms
 - 1 (the same) Flat Room Number (e.g. Flat 1, Flat 2, etc.) may exist in 0...* flats

FlatNo (PK)	FlatRoomNo (PK)
...	...

Lease Table

For the original table (Delete all the columns on the right-side of the other functional dependencies, leaving only the PKs from the new tables as Foreign Keys (FKs).

StudentAccommodationLease (LeaseNo, StudentNo, FlatRoomNo, FirstName, Surname, StartDate, FinishDate, FlatNo, FlatAddress)

LeaseNo (PK)	StudentNo (FK)	FlatNo (FK)	FlatRoomNo (FK)	StartDate	FinishDate
...	...	...	...	...	...

¹ Adapted from Database Systems: A Practical Approach to Design, Implementation and Management by Thomas Connolly & Carolyn Begg