

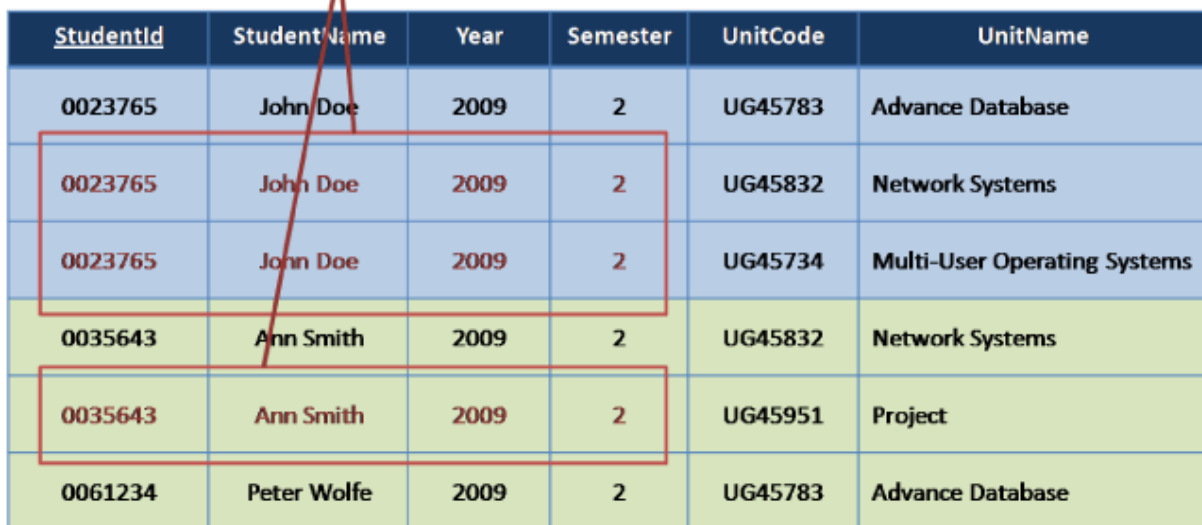
NORMALIZATION

First-Normal Form (1NF)

With our un-normalized relation now complete we are ready to start the normalization process. First Normal form is probably the most important step in the normalization process as it facilitates the breaking up of our data into its related data groups, with the following normalized forms fine tuning the relationships between and within the grouped data.

With First Normal Form we are looking to remove repeating groups. A repeating group is a domain or set of domains, directly relating to the key, that repeat data across tuples in order to cater for other domains where the data is different for each tuple.

Repeating Groups of Data



<u>StudentId</u>	StudentName	Year	Semester	UnitCode	UnitName
0023765	John Doe	2009	2	UG45783	Advance Database
0023765	John Doe	2009	2	UG45832	Network Systems
0023765	John Doe	2009	2	UG45734	Multi-User Operating Systems
0035643	Ann Smith	2009	2	UG45832	Network Systems
0035643	Ann Smith	2009	2	UG45951	Project
0061234	Peter Wolfe	2009	2	UG45783	Advance Database

In this example with Student ID as the primary key we see the three domains, StudentName, Year and Semester repeat themselves across the tuples for each of the different UnitCode and UnitName entries. Though workable it means our relation could potentially be huge with loads of repeating data taking up valuable space and costing valuable time to search through.

The rules of First Normal Form break this relation into two and relate them to each other so the information needed can be found without storing unneeded data. So from our example we would have one table with the student information and another with

the Unit Information with the two relations linked by a domain common to both, in this case, the StudentId.

So the steps from UNF to 1NF are:

1. Identify repeating groups of data. Make sure your model data is of good quality to help identify the repeating groups and don't be afraid to move the domains around to help with the process.
2. Remove the domains of the repeating groups to a new relation leaving a copy of the primary key with the relation that is left.
3. The original primary key will not now be unique so assign a new primary key to the relation using the original primary key as part of a compound or composite key.
4. Underline the domains that make up the key to distinguish them from the other domains.

Taking our original example once we have followed these simple steps we have relations that looks like this:

<u>StudentId</u>	StudentName	Year	Semester
0023765	John Doe	2009	2
0035643	Ann Smith	2009	2
0061234	Pete Smith	2009	2

<u>StudentId</u>	<u>UnitCode</u>	UnitName
0023765	UG45783	Advance Database
0023765	UG45832	Network Systems
0023765	UG45734	Multi-User Operating Systems
0035643	UG45832	Network Systems
0035643	UG45951	Project
0061234	UG45783	Advance Database

Composite Key

Tables in First Normal Form

Second-Normal Form (2NF)

Now our data is grouped into sets of related data we still need to check we are not keeping more data than we need to in our relation. We know we don't have any repeating groups as we removed these with First Normal Form. But if we look at our example we can see for every UnitCode we are also storing the UnitName.

<u>StudentId</u>	<u>UnitCode</u>	UnitName
0023765	UG45783	Advance Database
0023765	UG45832	Network Systems
0023765	UG45734	Multi-User Operating Systems
0035643	UG45832	Network Systems
0035643	UG45951	Project
0061234	UG45783	Advance Database

Would it not seem more sensible to have a different relation we could use to look up UG45783 and find the unit name 'Advanced Database'? This way we wouldn't have to store lots of additional duplicate information in our Student/Unit relation.

This is exactly what we aim to achieve with Second Normal Form and its purpose is to remove partial dependencies.

We can consider a relation to be in Second Normal Form when: The relation is in First Normal Form and all partial key dependencies are removed so that all non key domains are functionally dependant on all of the domains that make up the primary key.

Before we start with the steps, if we have a table with only a single simple key this can't have any partial dependencies as there is only one domain that is a key therfroe these relations can be moved directly to 2nd normal form.

For the rest the steps from 2NF to 3NF are:

1. Take each non-key domain in turn and check if it is only dependant on part of the key?
2. If yes
 - a. Remove the non-key domain along with a copy of the part of the key it is dependent upon to a new relation.
 - b. Underline the copied key as the primary key of the new relation.
3. Move down the relation to each of the domains repeating steps 1 and 2 till you have covered the whole relation.
4. Once completed with all partial dependencies removed, the table is in 2nd normal form.

In our example above, UnitName is only dependant on unitCode and has no dependency on studentId. Applying the steps above we move the unitName to a new relation with a copy of the part of the key it is dependent upon. Our table in second normal form would subsequently look like this:

<u>StudentId</u>	<u>UnitCode</u>
0023765	UG45783
0023765	UG45832
0023765	UG45734
0035643	UG45832
0035643	UG45951
0061234	UG45783

Tables in Second Normal Form

<u>UnitCode</u>	UnitName
UG45783	Advance Database
UG45832	Network Systems
UG45734	Multi-User Operating Systems
UG45951	Project

Third-Normal Form (3NF)

Third Normal Form deals with something called ‘transitive’ dependencies. This means if we have a primary key A and a non-key domain B and C where C is more dependent on B than A and B is directly dependent on A, then C can be considered transitively dependant on A.

Another way to look at it is a bit like a stepping stone across a river. If we consider the primary key A to be the far bank of the river and our non-key domain C to be our current location, in order to get to A, our primary key, we need to step on a stepping stone B, another non-key domain, to help us get there. Of course we could jump directly from C to A, but it is easier, and we are less likely to fall in, if we use our stepping stone B. Therefore current location C is transitively dependent on A through our stepping stone B.

<u>UnitCode</u>	UnitName	CourseCode	CourseName
UG45783	Advance Database	COMP2009	Computing
UG45832	Network Systems	COMP2009	Computing
UG45734	Multi-User Operating Systems	COMP2009	Computing
UG45951	Project	BUS22009	Business & Computing

Before we start with the steps, if we have any relations with zero or only one non-key domain we can't have a transitive dependency so these move straight to 3rd Normal Form

For the rest the steps from 2NF to 3NF are:

- 1.Take each non-key domain in turn and check it is more dependent on another non-key domain than the primary key.
- 2.If yes

- a. Move the dependent domain, together with a copy of the non-key attribute upon which it is dependent, to a new relation.
 - b. Make the non-key domain, upon which it is dependent, the key in the new relation.
 - c. Underline the key in this new relation as the primary key.
 - d. Leave the non-key domain, upon which it was dependent, in the original relation and mark it a foreign key (*).
3. Move down the relation to each of the domains repeating steps 1 and 2 till you have covered the whole relation.
 4. Once completed with all transitive dependencies removed, the table is in 3rd normal form.

In our example above, we have unitCode as our primary key, we also have a courseName that is dependent on courseCode and courseCode, dependent on unitCode. Though courseName could be dependent on unitCode it more dependent on courseCode, therefore it is transitively dependent on unitCode.

So following the steps, remove courseName with a copy of course code to another relation and make courseCode the primary key of the new relation. In the original table mark courseCode as our foreign key.

<u>UnitCode</u>	UnitName	CourseCode*
UG45783	Advance Database	COMP2009
UG45832	Network Systems	COMP2009
UG45734	Multi-User Operating Systems	COMP2009
UG45951	Project	BUS22009

<u>CourseCode</u>	CourseName
COMP2009	Computing
BUS22009	Business & Computing

Before we finish with Normalization there are a couple of little extras we need to consider.

Multiple and Nested Repeating Groups

As we mentioned earlier, First Normal Form is probably the most important step in the normalization process as mistakes here will have a ripple effect on the rest of the normalization process. Where some people get caught out is in applying 1st Normal Form steps and then moving on quickly to 2nd Normal Form and this is where many people are caught out.

One of the most vital steps for 1st Normal Form is to make sure you revisit each relation after you have normalized it to 1st Normal Form and make sure the rules are still not applicable to the new relation. In other words hidden within the first run through is another un-normalized relation.

Multiple Repeating Groups

A multiple repeating group is where there are two or more repeating groups within the relation to be normalized. The data within each repeating group is unconnected with any other data within a repeating group.

The normalization process is exactly the same as described previously, except it needs to be carried out for each repeating group. Each group of repeating domains will result in a separate table.

Nested Repeating Groups

A nested repeating group is where there are two or more repeating groups within the relation to be normalized. The data within one repeating group is nested within another repeating group. That is, the data is related.

The normalization process is exactly the same as described previously, except it needs to be carried out repeatedly till the relation is in 1st Normal form.

The upshot of both of these anomalies is you should never move on to 2nd Normal Form until you are absolutely sure the relation you are dealing with is defiantly in 1st Normal Form.