Support

Database Normalization

This article/tutorial will explain what database normalization is, why you need it and how to normalize a data model. We will use a simple example to explain the normalization rules.

What is Database Normalization

Database normalization is a technique of organizing the data in the database. Normalization is a formal approach that applies a set of rules to associate attributes with entities. Normalization is used when designing a database.

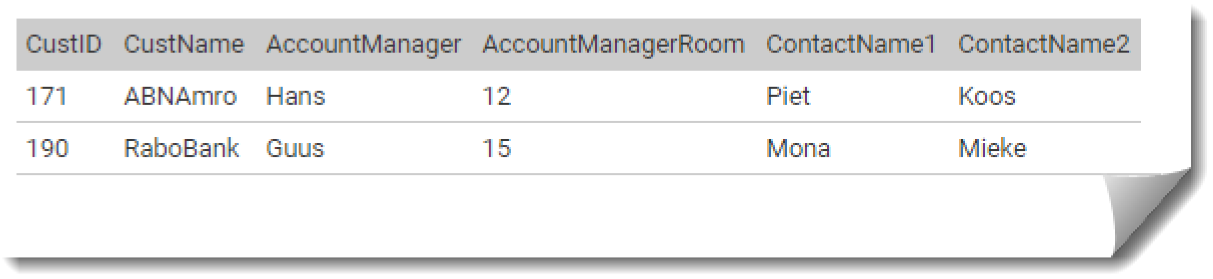
Database normalization is mainly used to:

- Eliminate reduntant data.
- Ensure data is logically stored (results in a more flexible data model).

Normalization of a data model consists of several steps. These steps are called normalization rules. Each rule is referred to as a normal form (1NF, 2NF, 3NF). The first three forms are the most important ones. There are more than 3 normal forms but those forms are rarely used and can be ignored without resulting in a non flexible data model. Each normal form constrains the data more than the previous normal form. This means that you must first achieve the first normal form (1NF) in order to be able to achieve the second normal form (2NF). You must achieve the second normal form before you can achieve the third normal form (3NF).

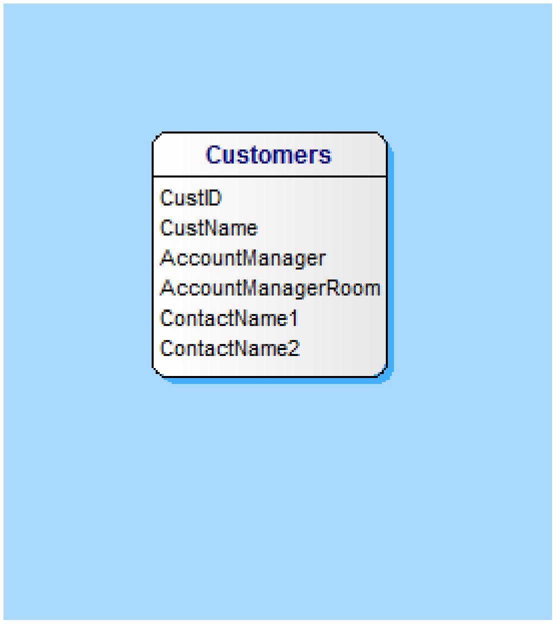
0NF: Not Normalized

The data in the table below is not normalized because it contains repeating attributes (contact1, contact2,...).



CustID	CustName	AccountManager	AccountManagerRoom	ContactName1	ContactName2
171	ABNAmro	Hans	12	Piet	Koos
190	RaboBank	Guus	15	Mona	Mieke

Not normalized customer data.

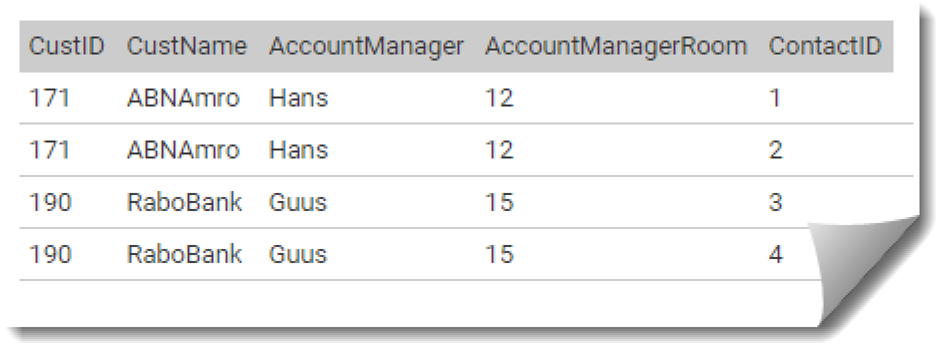


Not normalized (0NF) table/entity in a data model.

1NF: No Repeating Groups

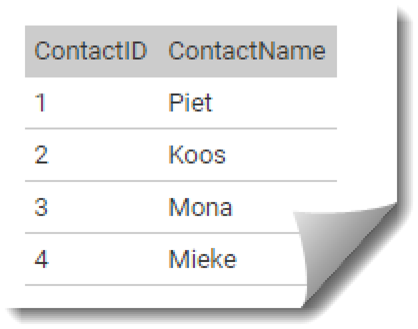
In the first normal form, an entity contains no repeating groups and all attributes must have a unique name.

Entities may only consist of two dimensions. A Customer has multiple Contacts (contact1, contact2,...). This is why we must create a new entity for the contacts. A data model containing attributes with names like contact1, contact2 etc, is a bad database design. Repeating attributes make your data less flexible and make it difficult to search for data.



CustID	CustName	AccountManager	AccountManagerRoom	ContactID
171	ABNAmro	Hans	12	1
171	ABNAmro	Hans	12	2
190	RaboBank	Guus	15	3
190	RaboBank	Guus	15	4

Customers.



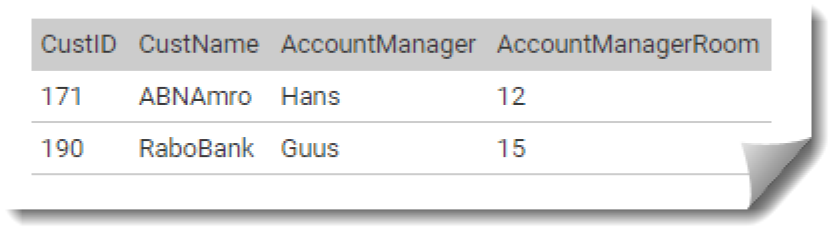
ContactID	ContactName
1	Piet
2	Koos
3	Mona
4	Mieke

Contacts.

2NF: Eliminate Redundant Data

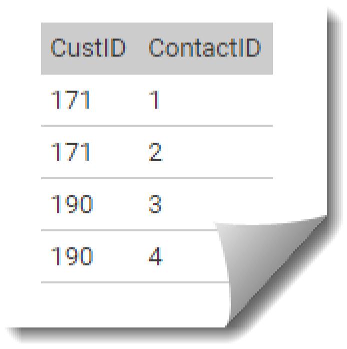
An entity is in the second normal form if it has achieved the first normal form and if all of its attributes depend on the complete (primary) key. An entity with a primary key consisting of only one attribute is in the second normal form (2NF).

In our example, the key is CustID + ContactID. If we remove the ContactID attribute we ensure that all attributes in the Customer entity depend on the new primary key CustID. We create a new entity CustomerContact for the relationship between Customer and Contact.



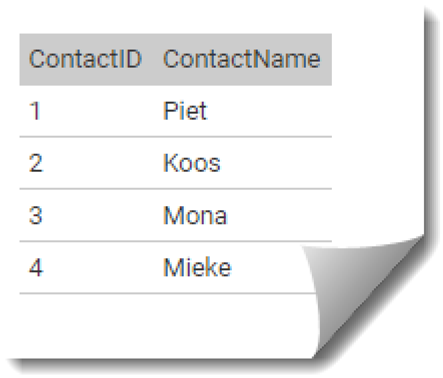
CustID	CustName	AccountManager	AccountManagerRoom
171	ABNAmro	Hans	12
190	RaboBank	Guus	15

Customers.



CustID	ContactID
171	1
171	2
190	3
190	4

CustomerContacts.



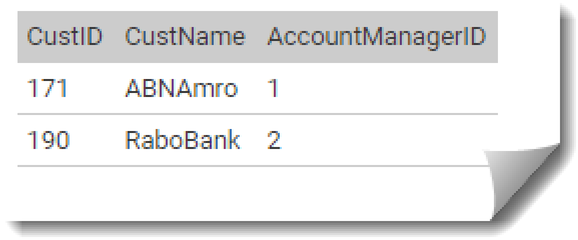
ContactID	ContactName
1	Piet
2	Koos
3	Mona
4	Mieke

Contacts.

3NF: Eliminate Transitive Dependency

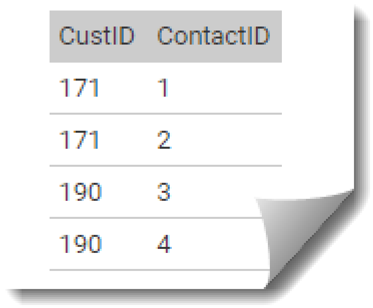
An entity is in the third normal form (3NF) if it is in the second normal form and all of its attributes are not transitively dependent on the complete primary key. Transitive dependency exists when a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key. In other words: The third normal form means that no attribute within an entity is dependent on an non-prime attribute that, in turn, depends on the primary key.

In our example, attribute AccountManagerRoom is transitive dependent on attribute AccountManager. A new entity AccountManager must be created so that we can remove attribute AccountManagerRoom from the Customer entity. The Customer entity is now in the third normal form (3NF).



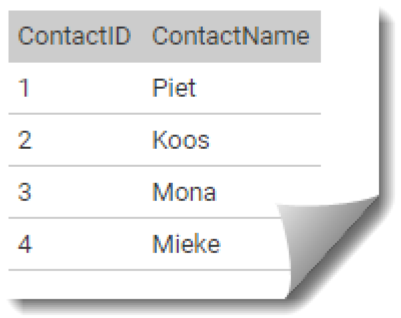
CustID	CustName	AccountManagerID
171	ABNAmro	1
190	RaboBank	2

Customers.



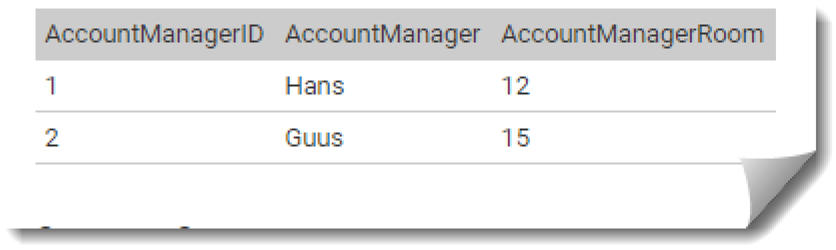
CustID	ContactID
171	1
171	2
190	3
190	4

CustomerContacts.



ContactID	ContactName
1	Piet
2	Koos
3	Mona
4	Mieke

Contacts.

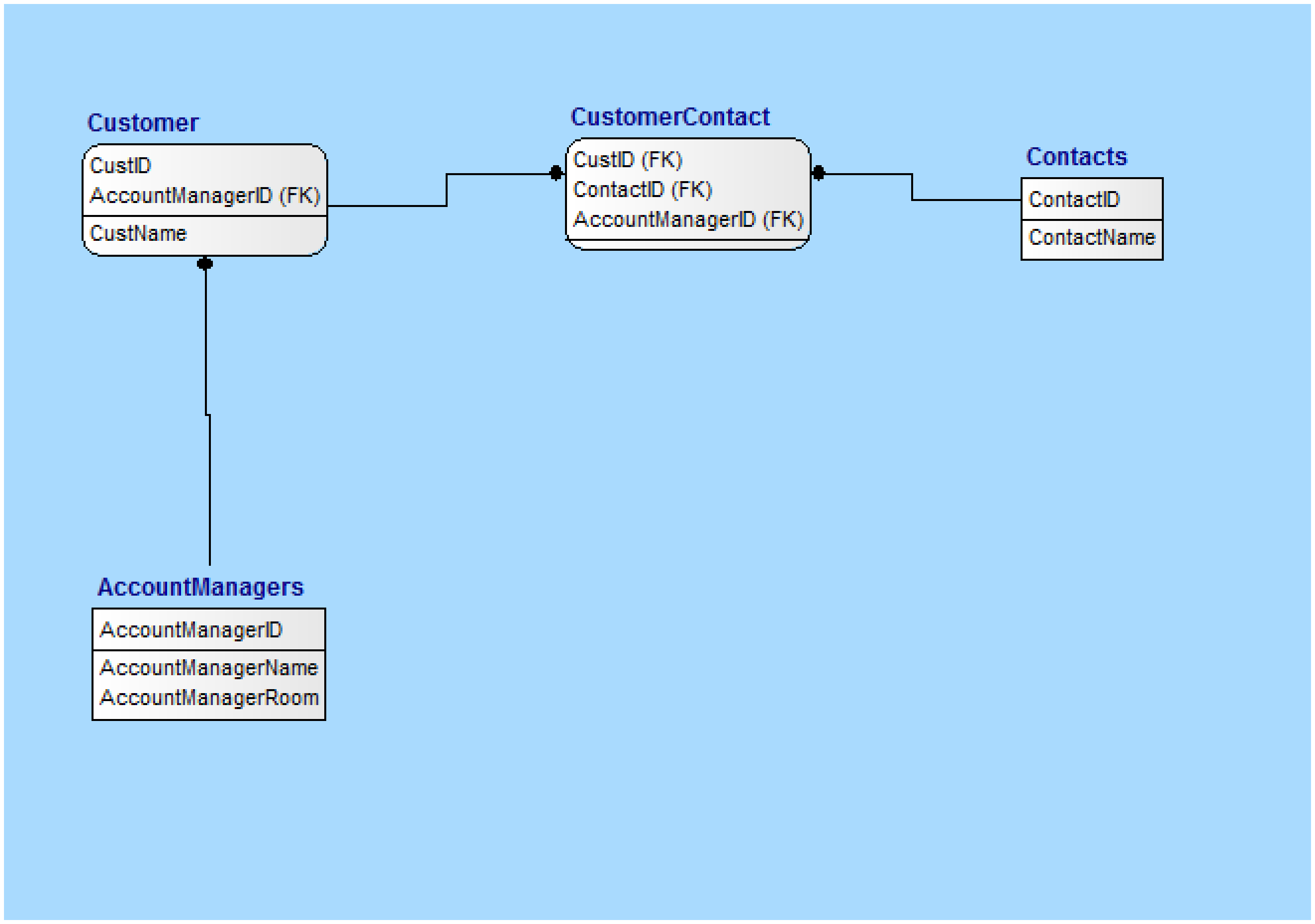


AccountManagerID	AccountManager	AccountManagerRoom
1	Hans	12
2	Guus	15

AccountManagers.

Data Model After Normalization

This is the data model after database normalization (3NF):



Data model after database normalization.

Build your next data model with DeZign for Databases

Download the free DeZign for Databases trial (../download/download-dezign.html)

Resources

Learn

- DeZign for Databases (../dezign/): Learn more about DeZign for Databases.
- Getting started with DeZign for Databases (../support/lt-dez001-getting-started-with-dezign.html): Start making a data model directly.
- Display data types in a diagram (../support/lt-dez002-display-data-types-in-a-diagram.html): Learn how to display data type and/or domain info in the entity boxes on your diagram.

Get products and technologies

- Build your next data model with DeZign for Databases trial software, available for download (../download/) directly from Datanamic's download section.
- Need (realistic) test data for your new database? Try out our data generator (../datagenerator/).

Products (../products/)

DeZign for Databases (../dezign/)

Datanamic DataDiff (../datadiff/)

Datanamic SchemaDiff (../schemadiff/)

Datanamic Data Generator (../datagenerator/)

Datanamic Repository (../repository/)

Datanamic MultiRun (../multirun/)

Purchase (../order/)

Online Store (../order/)

Support (../support/)

Support Overview (../support/)

Articles & Tutorials (../support/articles.html)

Videos (../support/videos.html)

Knowledge Base (../support/kb.html)

Company (../company/)

Contact Datanamic (../contact/)

About Us (../company/)

Customers (../customers/)

News (../company/news.html)

Datanamic Solutions

Hooigracht 15

2312 KM Leiden

The Netherlands