Module 07 – Normalization

Module Objectives

Identify the three levels used in normalization

Normalize a data model to the third normal level

Identify the three levels used in normalization

Six levels of normalization exist, however only three levels are actively used.

Level	Rule
First normal form (1NF)	An entity type is in 1NF when it contains no repeating groups of data.
Second normal form (2NF)	An entity type is in 2NF when it is in 1NF and when all of its non-key attributes are fully dependent on its primary key.
Third normal form (3NF)	An entity type is in 3NF when it is in 2NF and when all of its attributes are directly dependent on the primary key.

An Entity should describe a specific person, place, or thing that you want to record data about.

The First Normal Form – 1NF

Eliminate repeating groups of data and guarantee atomicity, another words the data is self-contained and independent.

The Second Normal Form – 2NF

Further, reduce the incidence of repeated data, not necessarily groups.

The table must meet the rules of the first form

Each column must depend on the whole key.

The Third Normal Form – 3NF

Deals with the issue of having all the columns in our table not just dependent on something, but dependent on the right something.

The table must be in 2NF.

No column can have any dependency on any other non-key column.

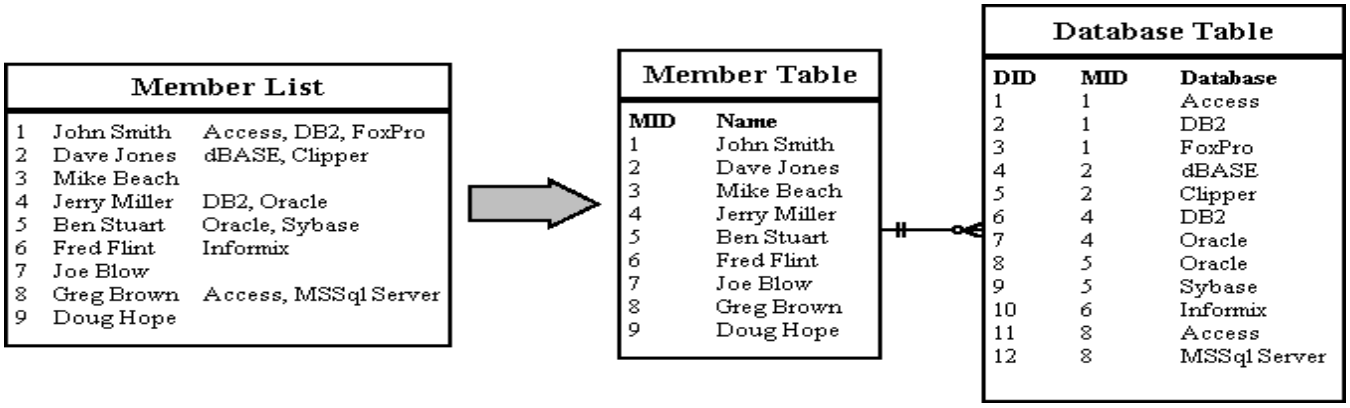
You cannot have derived data. (no computed columns)

Normalize a data model to the third normal level

1. Eliminate Repeating Groups

In the original member list, each member name is followed by any databases that the member has experience with. Some might know many, and others might not know any. To answer the question, "Who knows DB2?" we need to perform an awkward scan of the list looking for references to DB2. This is inefficient and an extremely untidy way to store information.

Moving the known databases into a separate table helps a lot. Separating the repeating groups of databases from the member information results in first normal form. The MemberID in the database table matches the primary key in the member table, providing a foreign key for relating the two tables with a join operation. Now we can answer the question by looking in the database table for "DB2" and getting the list of members.



Member List

1	John Smith	Access, DB2, FoxPro
2	Dave Jones	dBASE, Clipper
3	Mike Beach	
4	Jerry Miller	DB2, Oracle
5	Ben Stuart	Oracle, Sybase
6	Fred Flint	Informix
7	Joe Blow	
8	Greg Brown	Access, MSSql Server
9	Doug Hope	

Member Table

MID	Name
1	John Smith
2	Dave Jones
3	Mike Beach
4	Jerry Miller
5	Ben Stuart
6	Fred Flint
7	Joe Blow
8	Greg Brown
9	Doug Hope

Database Table

DID	MID	Database
1	1	Access
2	1	DB2
3	1	FoxPro
4	2	dBASE
5	2	Clipper
6	4	DB2
7	4	Oracle
8	5	Oracle
9	5	Sybase
10	6	Informix
11	8	Access
12	8	MSSql Server

2. Eliminate Redundant Data

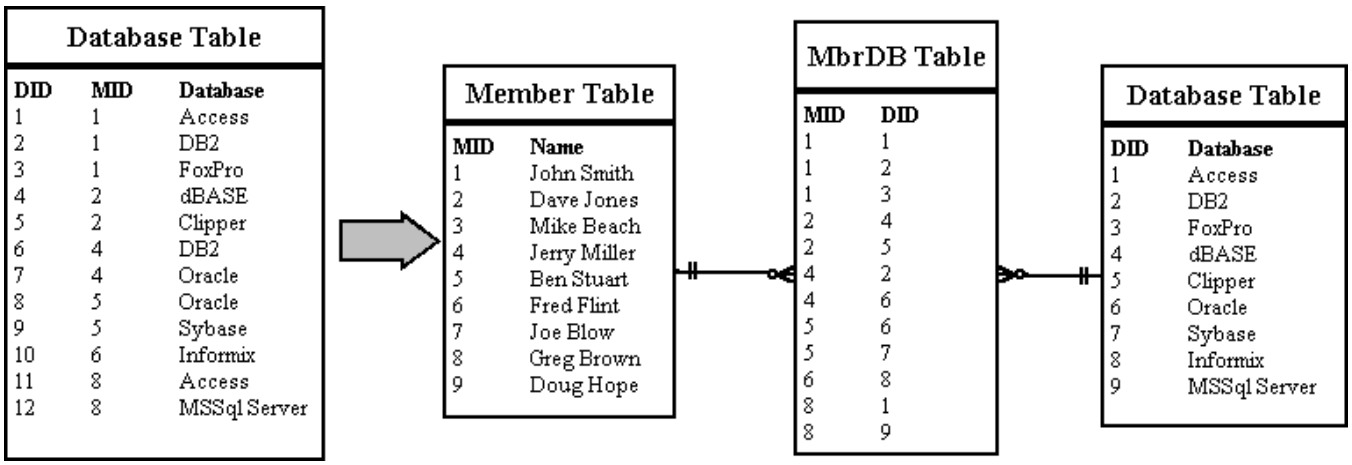
In the Database Table, the primary key is made up of the MemberID and the DatabaseID. This makes sense for the "Where Learned" and "Skill Level" attributes, since they will be different for every member/database combination. However, the database name depends only on the DatabaseID. The same database name will appear redundantly every time its associated ID appears in the Database Table.

Suppose you want to reclassify a database - give it a different DatabaseID. The change has to be made for every member that lists that database! If you miss some, you will have several members with the same database under different IDs. This is an update anomaly.

On the other hand, suppose the last member listing a particular database leaves the group. His records will be removed from the system, and the database will not be stored anywhere! This is a delete anomaly. To avoid these problems, we need second normal form.

To achieve this, separate the attributes depending on both parts of the key from those depending only on the DatabaseID. This results in two tables: "Database" which gives the name for each DatabaseID, and "MemberDatabase" which lists the databases for each member.

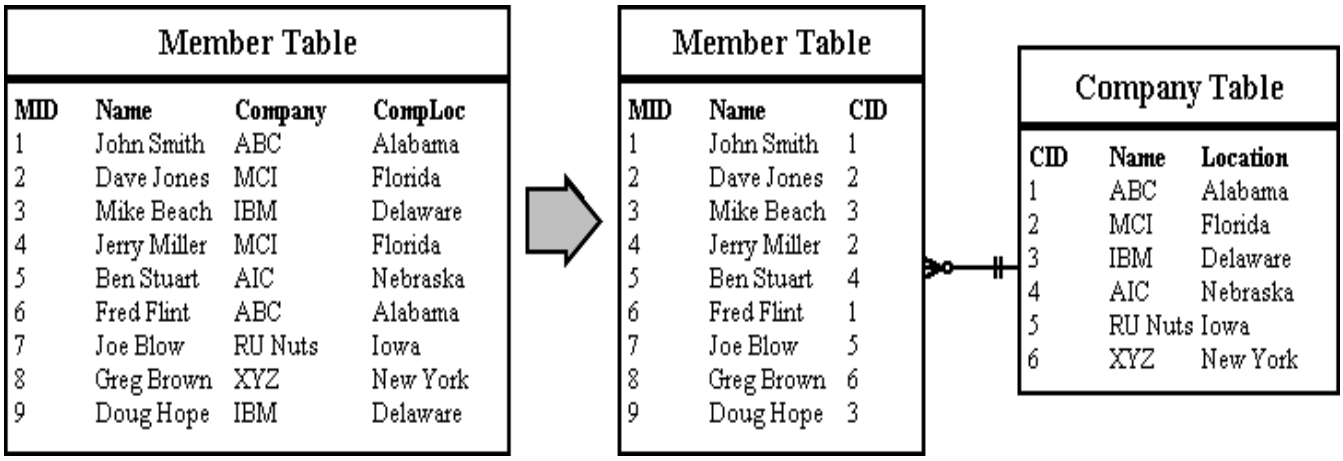
Now we can reclassify a database in a single operation: look up the DatabaseID in the "Database" table and change its name. The result will instantly be available throughout the application.



3. Eliminate Columns Not Dependent On Key

The Member table satisfies first normal form - it contains no repeating groups. It satisfies second normal form - since it does not have a multivalued key. However, the key is MemberID, and the company name and location describe only a company, not a member. To achieve third normal form, they must be moved into a separate table. Since they describe a company, CompanyCode becomes the key of the new "Company" table.

The motivation for this is the same for second normal form: we want to avoid update and delete anomalies. For example, suppose no members from the IBM were currently stored in the database. With the previous design, there would be no record of its existence, even though 20 past members were from IBM!



Member Table

MID	Name	Company	CompLoc
1	John Smith	ABC	Alabama
2	Dave Jones	MCI	Florida
3	Mike Beach	IBM	Delaware
4	Jerry Miller	MCI	Florida
5	Ben Stuart	AIC	Nebraska
6	Fred Flint	ABC	Alabama
7	Joe Blow	RU Nuts	Iowa
8	Greg Brown	XYZ	New York
9	Doug Hope	IBM	Delaware

Member Table

MID	Name	CID
1	John Smith	1
2	Dave Jones	2
3	Mike Beach	3
4	Jerry Miller	2
5	Ben Stuart	4
6	Fred Flint	1
7	Joe Blow	5
8	Greg Brown	6
9	Doug Hope	3

Company Table

CID	Name	Location
1	ABC	Alabama
2	MCI	Florida
3	IBM	Delaware
4	AIC	Nebraska
5	RU Nuts	Iowa
6	XYZ	New York