Data Modelling is designing the structure of the database.

Normalization is a set of rules which we need to follow when storing data in database to make the data more relevant, clean and easy to change.

Normalization makes data

* Data consistency
* Easy to retrieve and change
* Increase quality of data

**1NF**

1. Only one of each and each column

Don't mix up many information in one column

Eg. 1 person-> many interests

1. Need unique identification of data

|  |  |  |
| --- | --- | --- |
| Name | Class | ContactInfo |
| John | Mat100 | N 1st street, San Jose, 95031 |
| Mary | Mat100 | Saratoga Road, Fremont, 94071 |
| Vijay | Sci221 | Arques Ave, Sunnyvale, 94086 |
| Muks | Sci221 | Vicente dr, Sunnyvale, 94086 |

Consider student data like above

1. We need to split contactInfo column into address, city, zip code so there will be only 1 attribute for each column
2. We need to have StudentID as primary key for unique identification of each student.

Table after 1NF will be:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| StudentID | Name | Class | Address | City | Zipcode |
| 100 | John | Mat100 | N 1st street | San Jose | 95031 |
| 101 | Mary | Mat100 | Saratoga Road | Fremont | 94071 |
| 102 | Vijay | Sci221 | Arques Ave | Sunnyvale | 94086 |
| 103 | Muks | Sci221 | Vicente dr | Sunnyvale | 94086 |

**2NF**

Remove columns having redundant data

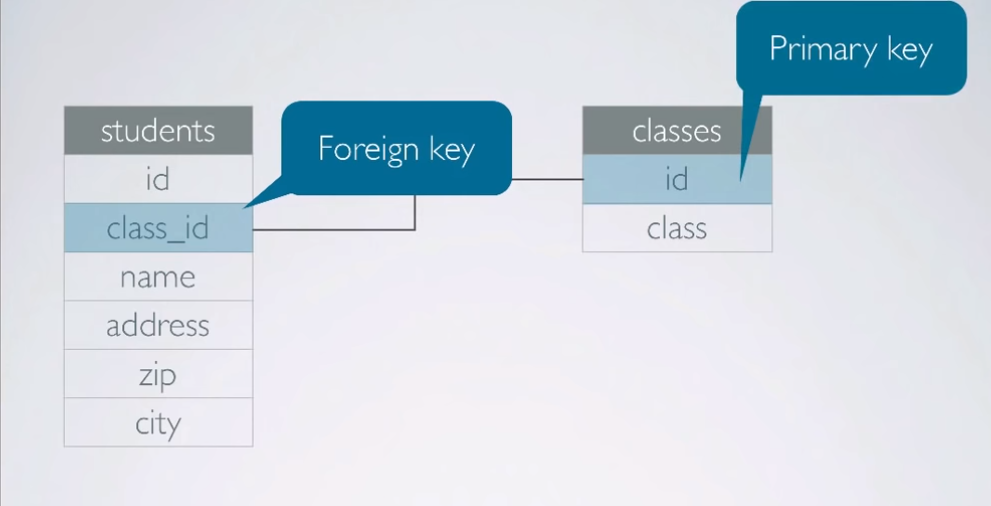
Eg. in above table we can see Class being repeated for students i.e. many students can belong to the same class. Hence as per 2NF we separate the class information into a table. Results look like below.

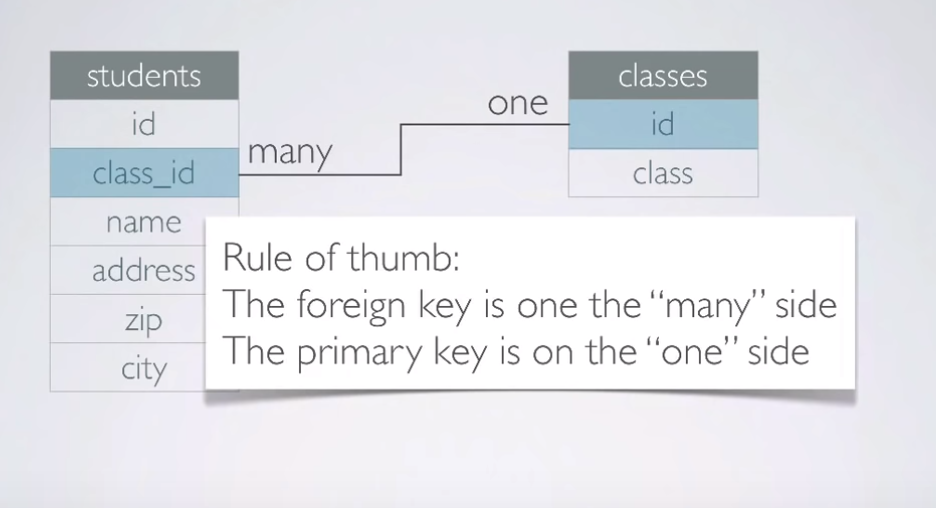
Student

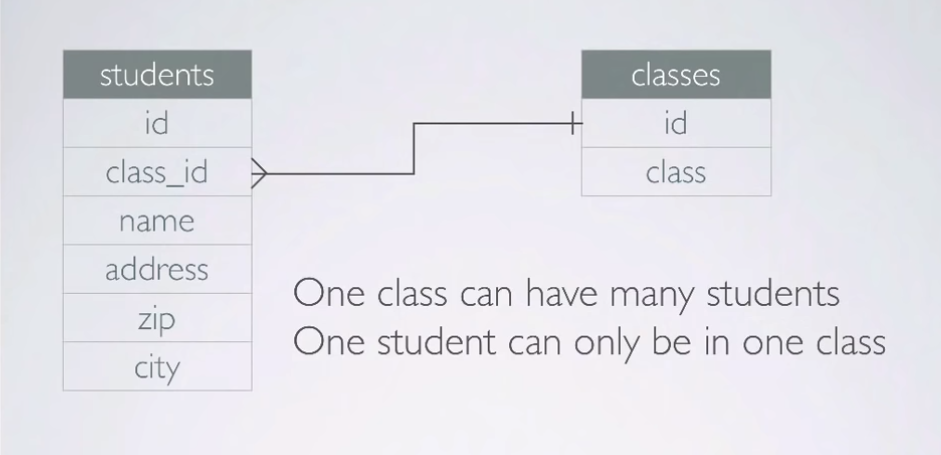
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| StudentID | Name | ClassID | Address | City | Zipcode |
| 100 | John | 10 | N 1st street | San Jose | 95031 |
| 101 | Mary | 10 | Saratoga Road | Fremont | 94071 |
| 102 | Vijay | 11 | Arques Av | Sunnyvale | 94086 |
| 103 | Muks | 11 | Vicente dr | Sunnyvale | 94086 |

Class

|  |  |
| --- | --- |
| ClassID | Class |
| 10 | Mat100 |
| 11 | Sci221 |



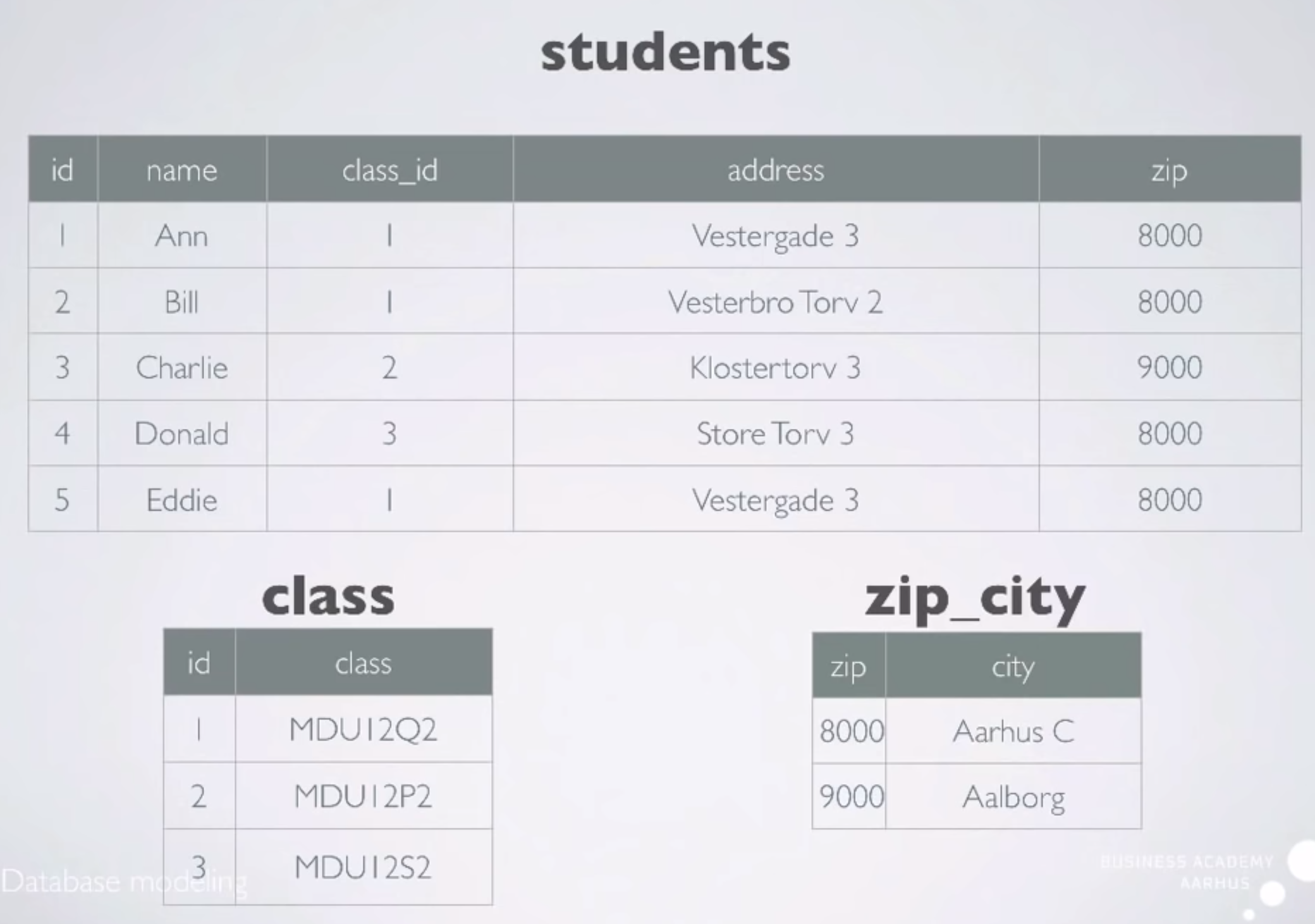




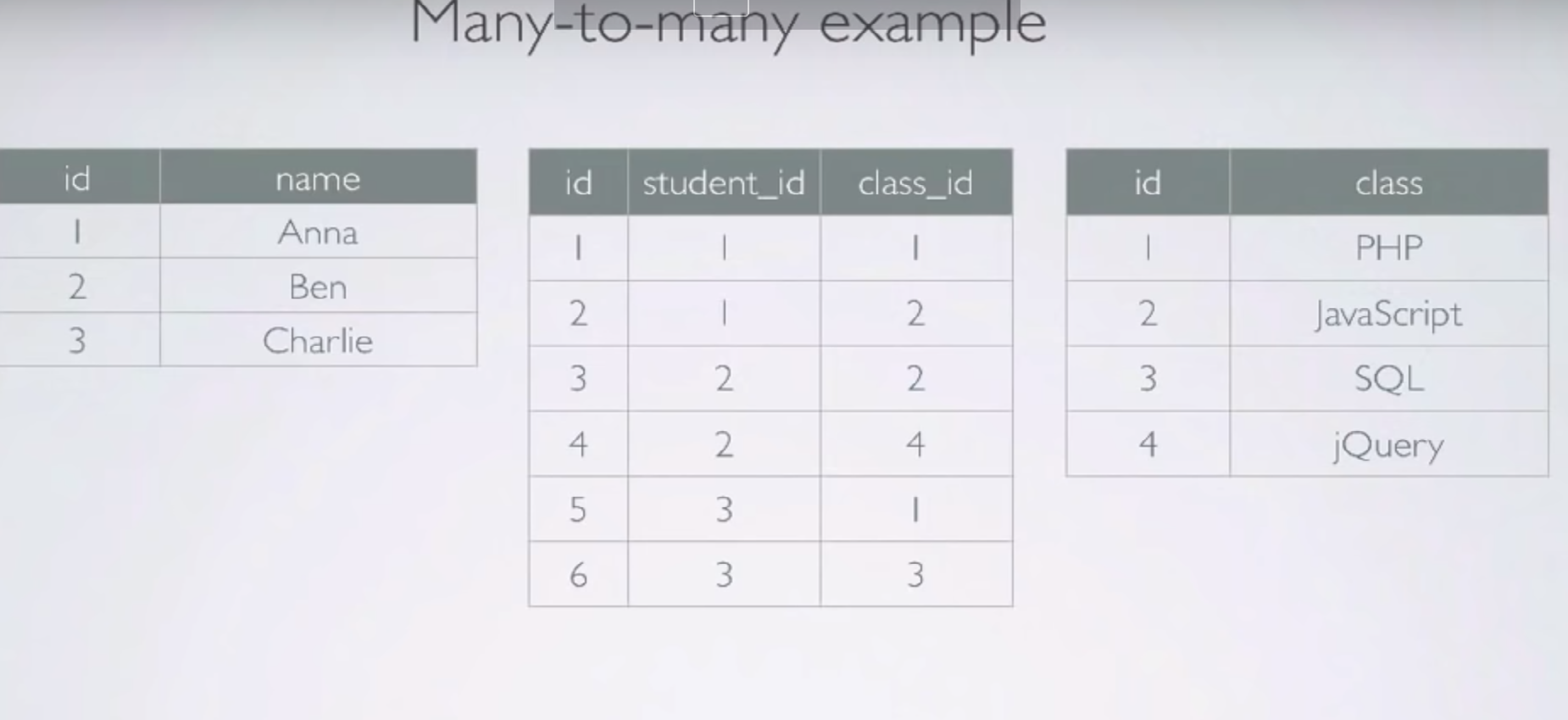
**3NF**

Only columns with direct dependency with the primary key should be in entity (table)

So, we split the zipcode and city from student as the column city is not directly dependent on StudentID but it’s dependent on zipcode.



Now, If there’s a many to many relationship between studentid and classid i.e. a student can enroll in multiple classes and a class can have multiple students then our model will look like this



However many to many relationships are not practically possible to be created in physical model. They are valid till logical model.

One to one relationships are very rarely practically used.

Uses can be.

Security- if you want to hide few attributes from user then create another 1:1 table for them

Avoid nulls. If data is like this

Title | videoname | bookno

Abc | null | 123

Cde | netflix123 | null

Create seperate 1:1 tables

Title video Book

Abc Cde | netflix123 Abc | 123

cde

**Conceptual, Logical and Physical Data Model**

* Conceptual Data Model:
  + It contains only the entities and relationships between the entities.
  + It’s easily understandable to the end user. Relationships are abstract. Can be written on paper or white board.
* Logical Data Model:
  + Entities and attributes. Attributes are divided in to key and nonkey attributes. Primary key- foreign key relationships are clearly defined. User friendly attribute names. Not specific to database.We can use tools like ERWin or powerdesigner to design logical DM.
* Physical Data Model:

We call entities-> Tables attributes-> Columns

Column names: database compatible not user friendly

Data types are defined for columns(attributes)

Non technical users- hard to understand

Also defines indexes, constraints, triggers etc at this level

**ER diagram: Entity relationship diagram**

ER notations

1:1 -|- -|-

-||- -0|- means 1st table has to contain value related to 2nd table and 2nd table may or may not contain related value

1:many -|--------<-

-||-------|-<- Similar as above an extra | means that column in relationship need to have related value 0 means it can have other unrelated values

Many:many

->-----<- | and 0 can be applied same as above