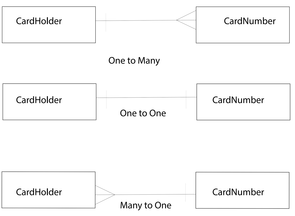
DataBase Design Strutures are:

1) EER model (Enhance entity relationship)
2) ERD model (Entity relationship diagram)
3) ER model (entity relatinship)

These are actually the structres that we follow for designing the database.
In database design we just make the entity and its attributes not going to add values.
ONLINE SOFTWARES AVAILABLE

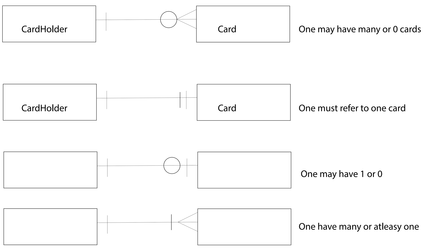
Cardinality: is a relationship between rows of one table with the rows of other table.
Possibilites of relationship in cardinality:
1) One to One
2) One to Many or Many to One

We can have the example of card and the cardHolder



Modality: is something that the child is required or relationship between the rows of the parent and child is required. This is actually the case of the not null characteristics. For Example: One card must have a cardHolder so this is here concept of not null. But there can be case that card are disable and dont have owner so now we have to remove the not null characterisitic.
We can do this by symbols:

1) ○ (not using the not null) 2) | (using the not null)



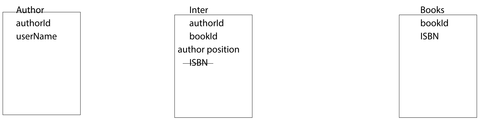
Normalization: is a process where we go through our database which may have some integrity problems and we reduce to efficent database after the normalization process.
FORMS:
1) 1 NF
2) 2 NF
3) 3 NF
Things should know:
Everything should be atomic
1) Data depending on other data means changing id must have diffrent user. here id is depending on new user.
2) You move from the 1 NF to 2 NF to 3 NF. Stepwise.

1 NF: Follow the atomic rules.
Problems:
1) Can't store one address which have multiple values in one column. so solution is to divide the address into multiple columns like state, provine.
2) Having 2 emails in one cloumn. we can make diffrent table for emails and can store multiple emails in new rows and can use FK.
3) Just change email and all the user value remain same in one table. Then it must repeat the PK which cant. So, you can make new email table and can repeat the user FK for differnt emails.

2 NF: You must be in 1 NF then we go for 2 NF.
So 2 NF deals with the parital dependency. But before it, we are going to understand dependency.
Dependency is something that depend on the priamry keys, like personId, name, address so name and address depend on the person id.
Vice versa we have car color so it do not depend on the person id.
For understanding the partial dependency, we take example of many to many where we have intermediary table.
Example of Author and Book (Many to Many)

First understand "author postion" is somehting that who has contribution in writing a book. like one author has 1 position and other maybe 2.
ISBN: is number of the book

So author postion is depend on the author as well to the book so we place it in intermediary table. But if we place the ISBN in the intermediary table then it relate to FK of bookId but not to the FK of authorId. So we should place the ISBN in the book table because it dependent on it not on both PK.



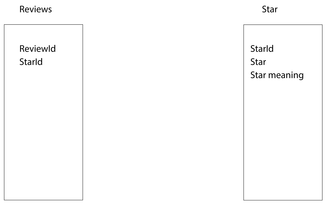
In 2 NF we simply remove the partial dependency to its table like here ISBN from inter to BOOK table

3NF: It must be in 2NF. It deals with the transitive dependency. Transitive dependency is that one column depend on other column and that column depend on the PK. This can be large like column to colmun to colmun and to PK maybe.

3NF is to remove the transitive dependency. Lets see example:
Reviews in buying something

NOTE: Here transitive depenecy is that start meaning depend on the start and star depend on the reviewId (PK).

For Removing it we make other table for the columns which are creating problems



Summary: 1 NF is to make atomic values, 2 NF is to remove partial dependency, 3 NF is to remove transitive dependency

---

Indexes: is something to help you find something faster.
3 Types:
1) Non Clusterd
2) Clusterd
3) Composite index

Non Clustered: A book has index page in the end. This index page points to actual data that the page number is that. So this is example of non clustered. In databases, if we make non cluster index then search where name = "JOHN" then datbase has index where it know that where it has JOHN in database and go there and check so its make things faster. Here if we are searching for name this column should be index column so that it may faster

Drawback: If wwe update anythignng in the database then we have to update the index.

Clustered: In which we organized the data in some way. PhoneBook is example of the clustered type where numbers are alphabetically ordered. So in databases we can make cluster and if we search where id = "Seventy" then it will go to J and search for me. Which make it faster. So, primary key usaully has the clustered index and we are using PK with where clasue

NOTE: The index help for join clause to do faster as it also go for ID and fetch data of that. If we use index then it will be faster. And the column which you are going to search for much be index. Primary key usaully clustered index.

Composite Index: index on two or more columns are called composite index. We can do as where firstName is this and lastName is that, so here applying index on two. There are exceptions to this.

In mysql order them in ceratin way like lastName and firstName and having index on both of them. So we can use where clause for finding with both conditions. Here we can set up like leftmost so it will check with last name and we cannot do with first name now because its not on left. So differnet database might have set up differently where you cant search individually or can.

when you make a composite index you should expect to always use that index by itself unless you have it set up to where you make a pretty awesome index different queries that you need to do like only on the last name or like leftmost

Data Types:
Main data types are:

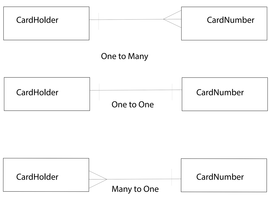
DataBase Design Strutures are:

1) EER model (Enhance entity relationship)
2) ERD model (Entity relationship diagram)
3) ER model (entity relatinship)

These are actually the structres that we follow for designing the database.
In database design we just make the entity and its attributes not going to add values.
ONLINE SOFTWARES AVAILABLE

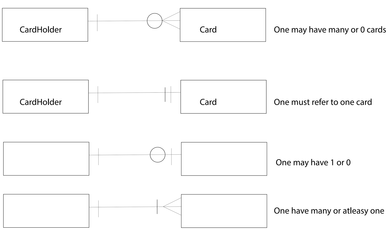
Cardinality: is a relationship between rows of one table with the rows of other table.
Possibilites of relationship in cardinality:
1) One to One
2) One to Many or Many to One

We can have the example of card and the cardHolder



Modality: is something that the child is required or relationship between the rows of the parent and child is required. This is actually the case of the not null characteristics. For Example: One card must have a cardHolder so this is here concept of not null. But there can be case that card are disable and dont have owner so now we have to remove the not null characterisitic.
We can do this by symbols:

1) ○ (not using the not null) 2) | (using the not null)



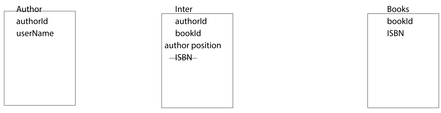
Normalization: is a process where we go through our database which may have some integrity problems and we reduce to efficent database after the normalization process.
FORMS:
1) 1 NF
2) 2 NF
3) 3 NF
Things should know:
Everything should be atomic
1) Data depending on other data means changing id must have diffrent user. here id is depending on new user.
2) You move from the 1 NF to 2 NF to 3 NF. Stepwise.

1 NF: Follow the atomic rules.
Problems:
1) Can't store one address which have multiple values in one column. so solution is to divide the address into multiple columns like state, provine.
2) Having 2 emails in one cloumn. we can make diffrent table for emails and can store multiple emails in new rows and can use FK.
3) Just change email and all the user value remain same in one table. Then it must repeat the PK which cant. So, you can make new email table and can repeat the user FK for differnt emails.

2 NF: You must be in 1 NF then we go for 2 NF.
So 2 NF deals with the parital dependency. But before it, we are going to understand dependency.
Dependency is something that depend on the priamry keys, like personId, name, address so name and address depend on the person id.
Vice versa we have car color so it do not depend on the person id.
For understanding the partial dependency, we take example of many to many where we have intermediary table.
Example of Author and Book (Many to Many)

First understand "author postion" is somehting that who has contribution in writing a book. like one author has 1 position and other maybe 2.
ISBN: is number of the book

So author postion is depend on the author as well to the book so we place it in intermediary table. But if we place the ISBN in the intermediary table then it relate to FK of bookId but not to the FK of authorId. So we should place the ISBN in the book table because it dependent on it not on both PK.



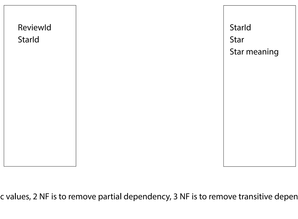
In 2 NF we simply remove the partial dependency to its table like here ISBN from inter to BOOK table

3NF: It must be in 2NF. It deals with the transitive dependency. Transitive dependency is that one column depend on other column and that column depend on the PK. This can be large like column to colmun to colmun and to PK maybe.

3NF is to remove the transitive dependency. Lets see example:
Reviews in buying something

NOTE: Here transitive depenecy is that start meaning depend on the start and star depend on the reviewId (PK).

For Removing it we make other table for the columns which are creating problems



Summary: 1 NF is to make atomic values, 2 NF is to remove partial dependency, 3 NF is to remove transitive dependency

---

Indexes: is something to help you find something faster.
3 Types:
1) Non Clusterd
2) Clusterd
3) Composite index

Non Clustered: A book has index page in the end. This index page points to actual data that the page number is that. So this is example of non clustered. In databases, if we make non cluster index then search where name = "JOHN" then datbase has index where it know that where it has JOHN in database and go there and check so its make things faster. Here if we are searching for name this column should be index column so that it may faster

Drawback: If wwe update anythignng in the database then we have to update the index.

Clustered: In which we organized the data in some way. PhoneBook is example of the clustered type where numbers are alphabetically ordered. So in databases we can make cluster and if we search where id = "Seventy" then it will go toJ and search for me. Which make it faster. So, primary key usaully has the clustered index and we are using PK with where clasue

NOTE: The index help for join clause to do faster as it also go for ID and fetch data of that. If we use index then it will be faster. And the column which you are going to search for much be index. Primary key usaully clustered index.

Composite Index: index on two or more columns are called composite index. We can do as where firstName is this and lastName is that, so here applying index on two. There are exceptions to this.

In mysql order them in ceratin way like lastName and firstName and having index on both of them. So we can use where clause for finding with both conditions. Here we can set up like leftmost so it will check with last name and we cannot do with first name now because its not on left. So differnet database might have set up differently where you cant search individually or can.

when you make a composite index you should expect to always use that index by itself unless you have it set up to where you make a pretty awesome index different queries that you need to do like only on the last name or like leftmost

Data Types:
Main data types are:
1) Strings
2) Numeric
3) Date

Strings have subtypes:
1) Char if we fix it size to 8 then store value of 6 then remaining 2 will also fill by default
2) Varchar uses the amount necessary to store the actual text.

Numeric are numbers:
1) Decimal of base 10
2) Floating point (store in binary, base 2, 0 and 1)
3) Integer stores only whole numbers
Which should use:
Decimal has accurate to mathematics while binary has problems in mathematics.
Numbers also have unsigned(positive) and signed(both positive and negitive)

DATES:
1) DataTime => Combination of date and time like 12:22:2000 then hours minutes etc.
2) TimeStamps => Exact moment time when some row is inserted or some action takes place. Like row inserted then its col may have timestap which will store exact moment.