⇒ Database Design
- → Informal design guidelines
- → Functional dependencies
- → Normal form based on Primary keys
- → General definition of 2nd & 3rd NF.

→ Informal design guidelines for Relational Schema
- Semantics of the attributes
- Reducing the redundant values in tuples
- Reducing Null values in tuples
- Disallowing Spurious tuples

1. Semantics of the attributes

Guideline 1: Design a relational schema so that it is easy to explain its meaning. Do not combine attributes from multiple entity types and relationship entity types into a single entity.

If a relational schema corresponds to one entity type or one relationship type the meaning tends to be clear. otherwise it tends to be a mixture of multiple entities and relationship and hence semantically not correct.

2. Reducing the redundant values in tuples.

One of the goal of the Schema design is to minimize the storage space that the base relation occupies.

Serious problem with redundancy is the problem of update Anamolies.

This is classified in to
→ Insertion Anamolies
→ Deletion Anamolies
→ Modification Anamolies.

for ex: Suppose a manufacturing Company stores the employee information in a table as bellows:

emp-id	emp-name	emp-address	emp-Department
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	maggie	Agra.	D890
166	glenn	Chennai	D900
166	glenn	Chennai	D004

- Update Anamoly : In the above table we have 2 rows for employee Rick as he belongs to two department of Company. If we want to update the address of Rick we have to update the same in two rows otherwise the data will become inconsistent.

Some how the correct address gets updated in one departoneat but not in other as per database Rick could be having two diffberent address which is not correct which leads to data inconsistency.

b) Insert Anamoly: Suppose a new employee joins the Company who is under the training and ~~not~~ currently not assigned to any departoneat, then we could not be able to insert data into the table if employee departoneat field does not alloco null values.

c) Delete Anamoly: Suppose if at a point of time the company closes the departoneat D890 then deleting rows that are having employee departoneat D890 could also delete the infermation of employee maggie. Since she is assigned only to this departoneat.

Guideline 2 :- Design the Relational Schema so that no insertion, deletion, modification anamolies occour in the relation.

If any anamolies are present note them clearly so that the Program that update database will operate correctly.

3. Reducing the null values in tuples

In some schema designs we may group many attributes together in to a bat relation.
Null values can have multiple inter-pretation. Such as
a) Attribute does not applied to this tuples
b) Attribute value for this is unknown.
c) The value is known but absent it has not been recorded yet.

Guideline 3: As far as possible avoid placing attributes in base relation whose values may be null. If nulls are unavoidable make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation.

4. Disallowing Spurious tuples:
Supurious tuples may occur after join operation. Good example vs Cartesian product.

Guideline No: 4. Design relational schema so that they can be joined with equality condition on attributes, that are either on primary keys and foreign keys

Date: / /

Functional dependencies :-

Definition :

. A functional dependencies is defined between 2 sets of attributes from database.

• 'X' functionally determines 'Y' in a relation schema, if and only if whenever 2 tuples R(X, Y) do agree on their X values, they must necessarily agree on 'Y' value.

Given a table T containing atleast 2 attributes designated by X and Y, we say that $X \rightarrow Y$

X functionally determines Y

OR

Y is functionally dependent on 'X'

eg:

Employee

SSN	fname

SSN ← fname

works-on

SSN	PNO	Hrs

{ SSN, PNO } ← hours

Date: / /

Inference rules for functional dependencies:

→ Inference rules are also known as Armstrong's Axiom. are published by Armstrong.

→ Armstrong Properties are

1. Reflexivity Property.

   $x \to y$ if y is subset of x

2. Augmentation property.

   if $x \to y$ is true, then

   $xz \to yz$ is also true.

3. Transitivity property.

   if $x \to y$ and $y \to z$ then

   $x \to z$ is true.

4. Union property.

   if $x \to y$ and $x \to z$ are true

   then $x \to yz$ is also true

this property indicates that right hand side of F.D contains many attributes then F.D exists for each of them

5. Decomposition Property

If $x \rightarrow y$ is implied and z is subset of y then $x \rightarrow z$ is implied.
This property is reverse of Union Property

6. Pseudo transitivity Property

$x \rightarrow y$ and $wy \rightarrow z$ are given
then $xw \rightarrow z$ is true.

Date: / /

Normalization:

Normalization is the process of removing redundant data from the tables in order to improve storage efficiency, data integrity, and scalability.

There are two goals of the normalization process
- Eliminating redundant data.
- Ensuring data dependencies

Why do we need normalization:

- Normalization is the aim of well designed Relational database management system. It is step by step set of rules by which data is put in its simplest forms. We normalize the RDBMS because of the following reasons:

- Minimize data redundancy.
- To make DB structure flexible i.e it should be possible to add new data values and rows without reorganizing the database structure.
- Data should be consistent throughout the DB i.e it should not suffer from following anomalies
  - Insert Anamoly
  - Delete Anamoly
  - Update anamoly
- Complex ~~query~~ queries required by the user should be easy to handle

- an decomposition of a relation in to smaller relations with fewer attributes on normalization the resulting relations whenever joined must result in the same relation without any extra rows. This is known as lossless join decomposition.

Advantages of Normalization

- more efficient data structure
- Avoid redundant fields or columns
- more flexible data structure
- Better understanding of data
-

Different forms of Normalization:

- first normal form
- Second " "
- Third " "
- Boyce - codd Normal form
- fourth normal form.
- fifth " ".

First normal form :- Its purpose is to eliminate repeating groups of attributes in an entity.

1st NF disallow Composite attributes, multivalued attributes, nested relations.

Attributes whose values for an individual tuples are atomic.

Consider the following table

Department

a)

Dname	Dnumber	Dmgr-SSN	Dlocation

Relational schema not in 1NF

b)

Dname	Dnumber	Dmgr-SSN	Dlocation
Research	5	333	Bangalore, Mangalore, Calcutta.
Administrate	3	555	Delhi
Headquatras	6	777	Bombay

c)

Dname	Dnumber	Dmgr-SSN	Dlocation
Research	5	333	Bangalore
Research	5	333	mangalore
Research	5	333	Calcutta.
Administrate.	3	555	Delhi
Headquatras	6	777	Bombay.

Date: / / 11

Seamd normal form : 2NF uses the concept of functional dependencies and Primary key.

A table is said to be in 2NF if both the following conditions hold :

- Table is in 1NF
- All non key attributes are fully functional dependent on the Primary key.

Consider the following example :

Customer Id.	Store Id.	Purchase location
1	1	Los Angeles
1	3	San Francisco
2	1	Los Angles
3	2	New York.
4	3	Sanfrancisco.

This table has a composite Primary key, [Customer Id, Store Id]. The non-key attribute is [Purchase location]. In this case [Purchase location] is only depends on [Store Id], which is only a part of Primary key, Therefore the table is not in 2NF.

- We break the table into two tables and. Now we have the following

Table - Purchase

Customer id	Store id
1	1
1	3
2	1
3	2
4	3

Table - Store

Store id	Purchase location
1	Losangeles
2	New York
3	Saibramesco

- Know the table is in 2NF. The [purchase location] is fully dependent on the primary key of the table which is [Store id].

3rd Normal form:

A Relational schema is in third NF if it satisfies the following Conditions:

- It is in Second Normal form
- There is no transitive functional dependency,

Transitive functional dependency:

A is functionally dependent on B and. B is functionally dependent on C. In this case, 'C' is transitively dependent on 'A' via B.

3rd Example:

Consider the following example:

Book-id	Genre-Id	Genre-type	Price
1	1	Gardening	25.99
2	2	Sports	14.99
3	1	Gardening	10.00
4	3	Travel	12.99
5	2	Sports	17.99

In this table

[Book-id] determines [Genre id]

and [Genre ID] determines [Genre type].

Therefore [Book Id] determines [Genre type] via [Genre id].

We have transitive functional dependency and this structure does not satisfy third NF.

To bring this table to third NF, we split table into two as follows:

Table-book

Book id	Genre Id	price
1	1	25.99
1	2	14.99
3	1	10.00
4	3	12.99
5	2	17.99.

Table-Genre

Genre-id	Genre-type
1	Gradning
2	Sports
3	Travel.

Date: / /

14

. Now all non key attributes are fully functional dependent only on the primary key.

. In [Table - Book], both [Genre Id] and [Price] are only dependent on [Book ID].

. In [Table - Genre], [Genre type] is only dependent on [Genre Id].

Boyce-Codd Normal Form (BCNF) was proposed as a simpler form of 3NF, but it was found to be stricter than 3NF. That is, every relation in BCNF is also in 3NF, however a relation in 3NF is not necessarily in BCNF.

Defination: A relational schema R is in BCNF if whenever a nontrival function dependency $X \rightarrow A$ holds in R, then X is a superkey of R

The formal defination of BCNF slightly differs from the defination of 3NF. The only difference b/w the two is that in 3NF ~~a~~ A relational is schema R is in 3NF if every nonprime attribute of R is non transitively dependent on everykey of R, but this condition does not exists in BCNF

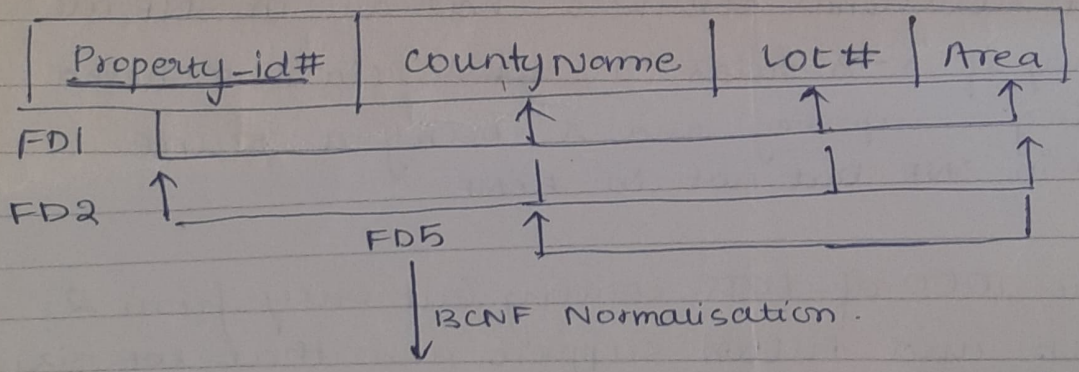
In practise, most relational schemas that are in 3NF are also in BCNF. Only if $X \rightarrow A$ holds in ~~redth~~ relationschema R with X not being a superkey and A being a prime attribute will be in 3NF but not in BCNF

Suppose we have 1000 of LOTS coming but only from 2 countries: Dekalb and Fulton. suppose also that lot sizes in Dekalb country are only 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 acres whereas lot size in Fulton country restricted to 1.1, 1.2 - - 20 acres. In such situation we could have the additional functional dependency FD5 : Area → country_Name. If we add this to the other dependencies, the relational schema. LOTSIA still is in 3NF because country_Name is a prime attribute.

The area of a lot that determines the county, a specified key FD5 can be represented by 16 tuples in seperate relation R (Area, County-name), since there are only 16 posible Area values. This representation reduces the reedundancy of repeating the same information in the thousand of LOTSIA tuples. BCNF is stronger normal from that would disallow LOTSIA and suggest need of decomposing it.

FD5 violates BCNF in LOTSIA because Area is Not the superkey of LOTSIA. Note that, FD5 satisfies 3NF in LOTSIA because county-name is a prime attribute (condition b), but this condition does not exist in defination of BCNF. We can decompose LOTSIA into 2 BCNF relation LOTSIAX and LOTSIAY. This decomposition loses the functional dependency FD2 because its attributes no longer coexist in the same relation after decomposition.

LOTSIA



LOTSIAX

Property-id#	Area	Lot#

LOTSIAY

Area	County-Name

Multivalued Dependencies and Fourth Normal Form :-

In database design there are certain cases where relation have constraints that cannot be specified as functional dependency.

Multivalued dependencies are a consequence of 1NF which disallows an attribute in a tuple to have a set of values.

If we have two or more multivalued independent attributes in ~~a single~~ the same relation schema, we get into a problem of having to repeat every value of one of the attributes with each value of the other attributes to keep the relation state consistent and to maintain the independence among the attributes involved. This constraint is specified by multivalued dependency.

EMP

Ename	Pname	Dname
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

Ename $\twoheadrightarrow$ Pname
Ename $\twoheadrightarrow$ Dname

* Here Ename works on Pname and has dependent Dname
* An employee may work on several projects and may have several dependents
* Emp's project and Emp's dependent are indepe of one another

To keep the relation state consistent we must have seperate tuples to represent every combination of employee's project and employee's dependent.

This constrained is termed as Multivalued dependency.

Informally, when two or more indepedent 1:N relationship A:B and A:C are mixed in same relation a MVD arises.

Formal Defination of MVD :-

A MVD $X \twoheadrightarrow Y$ specified on relation schema R where X and Y are both subset of R, specifies the following constraints on any relation state r of R : If two tuples t_1 and t_2 exists in r such that $t_1[X] = t_2[X]$ then two tuples t_3 and t_4 should also exist in r with the following properties —

$$t_3[X] = t_4[X] = t_1[X] = t_2[X]$$
$$t_3[Y] = t_1[Y] \text{ and } t_4[Y] = t_2[Y]$$
$$t_3[Z] = t_2[Z] \text{ and } t_4[Z] = t_1[Z]$$

where $Z = R - (X \cup Y)$.

* MVD $X \rightarrow\rightarrow Y$ is said to be a trivial MVD if it satisfies the below two conditions –

(a) Y is subset of X

(b) $X \cup Y = R$

* A MVD $X \rightarrow\rightarrow Y$ is said to be a non-trivial MVD if it does not satisfies the above conditions.

Fourth Normal Form

Defn

A relation schema is in 4NF with respect to a set of dependencies F (Functional and Multivalued) if for each non-trivial multivalued dependency $X \rightarrow\rightarrow$ in F^+, X is a superkey for R.

converting the relation EMP in 4NF

Since relation EMP is a all-key relation Hence it is by default in ~~4NF~~ BCNF

*EMP relation is not in 4NF because of

(i) Ename $\rightarrow\rightarrow$ Pname

(ii) Ename $\rightarrow\rightarrow$ Dname

(iii) Ename is not a superkey.

The relation EMP is decomposed into two relations in order to make it in 4NF based on the above MVDs (i) and (ii)

Emp_Project

Ename	Pname
Smith	X
Smith	Y

Emp_Dependent

Ename	Dname
Smith	John
Smith	Anna

This decomposition helps

(i) eliminate increased storage need

(ii) eliminate the update anomaly.

Joint dependency

The formal definition for Joint dependency is denoted by $JD(R_1, R_2 \cdots R_n)$ specified on relation schema R specifies a constraint on the states r of R.

Every legal state r of R should have a non-additive join decomposition into $R_1 R_2 \cdots R_n$.

These dependency is very peculiar semantic constraint i.e difficult to detect in pratice.

e.g- Consider a schema R having three attributes supp, parts + project.

supp supplies parts to a project.

R

Supp	parts	project
S_1	P_1	r_1
S_1	P_2	r_2
S_2	P_1	r_1
S_2	P_1	r_2

Now decompose R into three other schema such that $R \rightarrow (R_1, R_2, R_3)$

R_1

Supp	parts
S_1	P_1
S_1	P_2
S_2	P_1

R_2

supp	proj
S_1	r_1
S_1	r_2
S_2	r_1
S_2	r_2

R_3

parts	proj
P_1	r_1
P_2	r_2
P_1	r_2

Now let us merge the above decomposed relations R_1 R_2 & R_3 to form a schema R. Here we are merging for supplier S_1 only.

R_{new}

supp	parts	proj
S_1	P_1	r_1
S_1	P_1	r_2
S_1	P_2	r_2

After merging the decomposed schema $R_1 R_2 R_3$, we get the above schema R_{new} for supp S_1. In this schema we get a new tuple which was not there in original schema R. Such tuple is called additive Join.

If we don't get any new tuple after merging the decomposed schema then it is called non-additive loss less join.

A relation schema R has a join dependency if it is non-additive loss-less join.

5th Normal Form or Project join normal form (PJNF)

A relation schema R is said to be in 5th NF if it satisfy following conditions

1) R must be in 4NF

2) if Joint Dependency (JD) not exists
   R is in 5NF

   else if JD exists
   check for trival JD if yes
   R is in 5NF

   elseif (All R_i is superkey) // $R_i \in R_1, R_2, R_3 \ldots R_n$
   R is in 5NF

   else
   R is not in 5NF

So, if above two conditions are satisfied then we say the relation schema R is in 5NF.