Normalization:-

--Normalization is a process of organizing the data in db to avoid the redundency(duplication).

--because of data redundency threre are several problems in the DB.

Anomalies in DB:
==========

--An anomaly is something is diff from what is normal or usual(insconsistency) or abnormalities:

--lets try to have a single table to manage emp details:

EID	NAME	SALARY	DID	DNAME	LOCATION
1	Ram	5000	1001	HR	DELHI
2	Shyam	6000	1001	HR	DELHI
3	MANOJ	6500	1001	HR	DELHI
4	Suresh	7000	1002	SALES	MUMBAI
5	RAVI	7200	1002	SALES	MUMBAI
6	Ramesh	7500	1003	Accounts	Chennai

Here entire dept related data is repeated for each emp.(data redundency)

data redundency: when same data is stored multiple time unnecessarly in DB.

Redundency occurs when we try to keep all the data in a single table.

problems with data redundency:
----
1. insertion, updation, deletion anomalies
2.inconsistnecy of DB
3.increase the DB table size and slow the performance while fetching the data.

Insertion anomalies:- when certain data can not be inserted into the table without the presence of other data.

Updation anomalies: --when we want to update a single piece of data, but is must be updated at all its copies.

deletion anomalies:- if we delete some data, it cause deletion of some other data.

--to solve the above problem we need to normalize(decompose) the table into multiple related tables.

Note: the main purpose of normalization is to avoid the data redundency and maximize the efficiency of the DB.

In Normalization we should split a table in multiple tables so that each table should contains a single idea/concept.

--with the normalization we refine a big table into multiple related tables.

so the above table we should split into 2 tables:

1. dept
2.emp

--to normaize a table we have diff types of normal forms:

1. INF
2. 2NF
3. 3NF-----> upto 3NF data redundency is almost minimized or removed
4. BCNF
5. 4NF
6. 5NF

--each normal form provides a diff level of refinement of a DB.

First Normal Form:
==========

--table should not contains any multivalue attributes(comma/space seperated values)
--each cell should contains only atomic value
--a table should not have the repeating columns.

EMPID	NAME	DEPT_NAME
1	RAM	HR, SALES
2	RAVI	Marketing
3	AMIT	HR, Accounts

so the above schema is in 0NF or unnormalized.

lets convert it into the 1st NF.

solution1: valid soultion

EMPID	NAME	DEPT_NAME
1	RAM	HR
1	RAM	SALES
2	RAVI	Marketing
3	AMIT	HR
3	AMIT	Accounts

here empid can not be PK, here we need to take PK as composit PK (empid,dept_name).

solution2: invalid solution

EMPID	NAME	DEPT_NAME1	DEPT_NAME2
1	RAM	HR	SALES
2	RAVI	Marketing	Null
3	AMIT	HR	Accounts

--the above solution is violates the 1NF because of repreating column.

Solution3: valid solution

--we can devide the table into 2 tables:

1. as a base table
2. as reference table

1. emp table:

EMPID	NAME
1	RAM
2	RAVI
3	AMIT

2. emp_dept table

EMPID	DEPT_NAME	// here EMPID will be the FK
1	HR	
1	SALES	
2	MARKETING	
3	HR	
3	ACCOUNTS	

2NF :
=====

To understand the 2NF or to normalize a table in 2nd NF we need to understand following concepts:

1. functional dependency
2. super key
3.candidate key
4.prime attribute
5.non-prime attribute

Key:
====
--generally we keep data in a table ,so that later we can retrieve it easy manner.

--in a table all col has a unique name but for a row we don't have a unique name. so inorder to find a row
uniquly we required a key.

--so a key is an attribute or set of multiple attributes that uniquely identify a row/record in a table.

school:----students.

ram, father_name, address, age ----->

super key:

--all the valid combination of attributes by using we can find a row uniquely in a table.

ex:

student(roll, name marks, address, dob, email)

1.roll *
2.roll name
3.roll name address
4.email
4. name email
5.name address
6.name dob
7.address email
8. dob email

Candidate key:

--it is a minimal set of super key.

Note: a candidate key should not have a subset as another super key.

1. roll

2.email

3.name address

6.name dob

Primary key:
----

here from the cadidate key DBA will choose a PK, generally it will be minimum number of attribute declared as PK.

some time we can make other than minimum value attribute also as a PK (composit key)

Prime attribute:
==========

--those attribute that r part of any candidate key is called prime attribute
--and those attribute which are not part of any candidate key is known as non-prime attribute.

in the above example : marks will be a non-prime attribute.

1stNF does not eliminate redundency,

--2nF applies in a table which is having a composit key, i.e a table with a PK compound with two or more attribute.

--Note: a table with a single column PK is automatically in 2NF.

--to be in 2nF, a table must be in 1stNF and the table must not contains any partial dependency.

partial dependency:

Note: if the proper subset of a candidate key determines non-prime attribute ,then it is called PD.

--the normalization of 1NF table to the 2nF involves the removal of PD .

--if any PD exist, we remove the partial dependecy attribute from the table by placing them in a new table.

ex:

ROLL	CID	FEE
1	c1	1000
2	c2	1500
1	c3	2000
3	c4	1000
3	c1	1000
2	c5	2000

--here there are many courses having same fee.

--here there will be only one Candidate key(roll and cid) that can uniquley indentify the record.

prime attribute:(roll and cid)
non-prime attribute (fee)

here fee is dependent on the CID, which is the example of PD. becoz CID is a proper subset of candidate key
--the above table is not in 2nF.

lets convert it into the 2NF

table 1:

(roll, cid)

table 2:
(cid, fee)

3NF:
======

Although 2ndNF relations have less redundencies than those in 1stNF still have a chance of data redundency because of transitive dependency.

--in order to make a table in 3rdNF we need to remove TD from the table.

A relation will be in 3NF if only:

1.it should be in 2NF.
2.there should not be any TD.

TD: if a non-prime attribute is transitivly dependent on primary key.
ex:

if A ------> B and B-------->C then A---->C is called TD.

ex:

Roll Name Age Country State

1	RAM	25	INDIA	MP
2	RAM	35	INDIA	UP
3	RAVI	28	INDIA	MAHARASTRA

functional dependencies:

roll----> name
roll----> Age
roll----> State
State ----> Country

here the candidate key will be (roll)

Roll--->State and State ----> Country ,,, so Country is transitivly dependent on Roll , and it will violates 3NF,

--to convert this relation in 3NF we need to decompose this relation in multiple related tables:

Student(roll, name, age, state)

State_Country(state, country)

Transaction management:
==========

Transaction:

--it is a set of related operations used to perform a logical unit of work.

ex:

withdrawing money from the ATM
transfering amount from one account to another.

--tx access data using read and write operations.

--if a tx fails it should not be resumed, instead it should be restarted.

---->
-- 1000
-- 2000
-- ----------
-- 3000
--
--
--
--
--->

500
2500

3000

in sql to manage the tx we need to use

TCL(Transaction controll language)
rollback
commit
savepoint

In order to maintain consistancy in DB , before and after the tx certain properties should be followd
these are called ACID properties:

A: (Atomicity) : (All or noting rule) the entire tx takes place at once or does not happens at all.

C: Consistancy: the DB must be consistant before and after tx

I: Isoloation: multiple tx should occurs independently at a time without any interference.

D: Durability: The changes of a sucessfull tx should be permanent even if system failure occurs.

--if we perform any DML operation on a table inside a tx area, that operation will be partically commited(it will be stored in the local copy)

--and any time we can rollback these operation

--but once we do commit, then only it will be stored permanently.

Note: In Mysql it is bydefault autocommit is enabled.

Note: in the DB all the DDL statements are by default commited.

--in mysql to disable the autocommit mode:

>set aut0commit = 0;

to start the tx area in mysql:

>start transaction

>delete from x1 where id= 102;

>rollback;

>delete from x1 where id= 102;

>commit;

savepoint p1;