Relational and E-R Model

- A model in database system basically defines the structure or organization of data and a set of operations on that data.
- **Relational model** is a simple model in which database is represented as a collection of relations where each relation is represented as a two-dimensional table.
  - Tuple – Each row in a table
  - Attribute – Name of each column.
  - Domain – A set of permissible values that can be given to an attribute.
  - A relation consists of:
    - Relational schema
    - Relational Instance
- **A relational schema** specifies the relation's name, its attributes and the domain of each attribute.

  If R is the name of the Relation and A1, A2, A3… An is the list of attributes representing R, then R(A1,A2,A3….An) is the relational schema.

  Each attribute in its relational schema takes a value from some specific domain (A_i).
- **Relational Instance** or **Relation state**, denoted by r is a collection of tuples for a given relational schema at a specific point of time.

  A relation state r of a relational schema R(A_1, A_2, A_3, …. A_n), also denoted by r(R) is a set of n-tuples

  $R = \{t_1, t_2, t_3, ..., t_n\}$

Example 1

Relational schema for **Student**:

Student(Roll_no# String, name: string, age: integer)

Student

	Roll_no	Name	Age
t_1	3467	Preeti Zinta	21
t_2	3468	Subhas Chandra	20

t_1 = (3467,Preeti Zinta,21)

Super Keys, Candidate Keys and Primary Keys for a Relation.

Super Key – A super key is an attribute or a set of attributes used to identify the records uniquely in a relation.

Relational Schema for PERSON

PERSON(person_id, name, age, address)

person_id	name	Age	address
1	Sanjay	35	Gariahat
2	Sharad	45	Behala
3	Dhruba	25	Garia

In this relation,

Person_id, is a super key, since it is unique for each person.

Similarly (Person_id, Age) and (Person_id, Name) are also super keys of the relation PERSON since their combination is also unique for each record

Candidate Key – Super keys of a relation can contain extra attributes. Candidate Keys are minimal super keys – i.e. with no extraneous attributes.

An attribute is called extraneous, if even after removing it from the key, the remaining attributes still has the properties of a key.

The following properties must be satisfied by a candidate key:

- A candidate key must be unique
- A candidate key's value cannot be NULL
- A candidate key is a minimal set of attributes
- The value of a candidate key must be stable.

Primary Key – A relation can have more than one candidate keys and one of them can be chosen as a primary key.

e.g. in PERSON, there can be two candidate keys Person_id and name, where Person_id can be chosen as primary key.

Relational Constraints

There are three types of constraints on relational database.

1. Domain constraint
2. Primary Key constraint
3. Integrity constraint
    a. Entity Integrity constraint
    b. Referential Integrity constraint

Domain Constraint

It specifies that each attribute in a relation must contain an atomic value only from the corresponding domain.

The data types associated with commercial RDBMS domain include

1. Standard numeric data type for integers.
   (short integer, integer, long integer)
2. Real numbers.
   (float, double)
3. Characters
4. Fixed length strings and variable length strings

Thus domain constraint specifies the condition that we want to put on each instance of the relation, so that the values that appear in each column must be drawn from the domain associated with that column.

e.g. Age in the STUDENT Relation always belongs to the integer domain within a specific range (if any) and not to strings or any other domain.

Key Constraint

This constraint states that the key attribute value in each tuple must be unique. This is because the value of the primary key is used to identify the tuple in the relation.

Integrity Constraint

There are two types of integrity constraints.

Entity Integrity Constraint

It states that no primary key value can be NULL. This constraint is specified on one individual relation.

Note that '#' identifies the primary key of a relation.

Referential Integrity Constraint

It states that the tuple in one relation that refers to another relation must refer to an existing tuple of that relation.

This constraint is specified on two relations. It uses the concept of foreign key.

R

A#	B	C^
A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C5

S

E	C#
E1	C1
E2	C3
E3	C5
E4	C2

In the above example, the value of C^ in every R tuple is matching with the value of C# in some S tuple.

If a tuple (A6, B2, C4) is added, then it is invalid since referential relation S does not include C4. Thus it will be a violation of referential integrity constraint.

There are three basic operations to be performed on a relation:

1. Insertion
2. Deletion
3. Updation

Insertion operation: it allows us to insert a new tuple in a relation. When we try to insert a new record, any of the four constraints can be violated:

- Domain constraint – if value given to an attribute lies outside the domain of the attribute
- Key constraint – if the value already exists in the relation
- Entity Integrity constraint – if the primary key entered is null.
- Referential Integrity constraint – If the value of the foreign key in t refers to a tuple that does not appear in the referred location.

PERSON(person_id#, name, age, address)

person_id#	name	age	address
1	Sanjay	35	Garia
2	Rahul	25	Gariahat
3	Tamal	26	Ballygunge

What are the violated constraints and what reasons

(i) INSERT<1, 'Bipul', 20 'salt lake'>

Violated constraint – Key constraint

Reason -primary key 1 already exists

(ii) INSERT< 'null' 'Anurag', 25, 'Garia'>

Violated constraint – Entity integrity constraint

Reason – primary key is null

(iii) INSERT< 'abc', 'Suman', 25, 'Shyambazar'>

Violated constraint – domain constraint

Reason – value of person_id is string.

The Deletion operation

Using this operation, some existing records can be deleted from a relation.

Only one type of constraint can be violated during deletion. It can occur when you want to delete a record in a table where it is referenced by the foreign key of another table.

The violated constraint is "Referential integrity constraint"

Dealings

1. Rejection of deletion.
2. DBMS can automatically delete all tuples that has a reference with the one which you want to delete. – Cascade deletion.

Normalization

Consider the following table: STUDENT

Roll	Name	Addr	cno	cname	Teacher	Office
341	Amit	Kol	011	Phy	NK	102
341	Amit	Kol	012	Chem	AS	105
341	Amit	Kol	013	Maths	PA	103
345	Rahul	Delhi	013	Maths	PA	103

Lot of information has been repeated and therefore the table has Data Redundancy

There are three anomalies in this database system:

(i) Update Anomaly: Redundant information make updation difficult because an update in one particular data needs all the related redundant data to be updated. Otherwise data inconsistency or loss of integrity will result.

e.g. changing the name of the teacher needs all tuples with the same teacher to be changed.

(ii) Insertion Anomaly: If the primary key here is (Roll, cno), a new tuple insertion means the primary key should have a value (as per entity integrity constraint). But here if we want to insert a new course and a teacher, it cannot be inserted until a new student enrolls for the course.

e.g. The course "Comp Science" whose teacher is "RP", cannot be inserted unless a student enrolls in Chemistry.

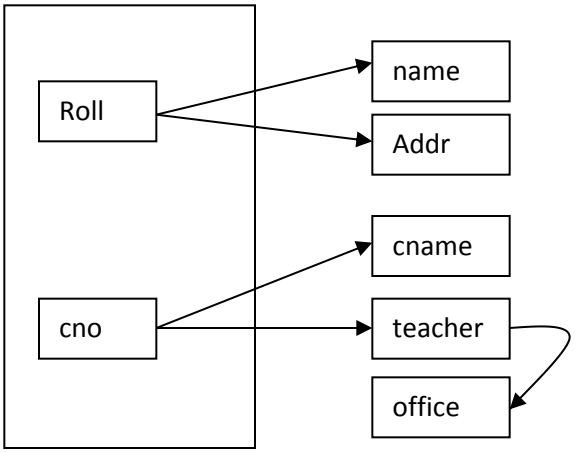
(iii) Deletion Anomaly: Many important information may be deleted, if a tuple needs to be deleted.

e.g. if 2nd tuple is deleted, the information that the teacher AS takes Chemistry, gets deleted.

These anomalies arise because the relation STUDENT has information about students as well as subjects.

One solution to this problem is to decompose the relation into two or more smaller relations.

First let us see, how the data is related in the relation:



Normalization involves decomposition of a relation into smaller relations based on the concept of functional dependence to overcome undesirable anomalies.

Functional Dependency

Consider that a relation R has two attributes A and B. The attribute B is functionally dependent on A, if attribute A, uniquely determines the value of B.

B is functionally dependent of A: B = f(A)

Where the dependent variable is B and independent variable is A.
This can also be read as A determines B.

Or, for each value of A, no more than one value of B is associated.

In the above example of relation STUDENT, the dependencies can be written as:

Roll → Name
Roll → Addr
Cno → cname
Cno→ teacher
Teacher → office

Single Valued Normalization

Codd in the year 1972, presented three Normal Forms (1NF, 2NF, 3NF). These were based on functional dependencies among the attributes of a relation.

Later Boyce and Codd proposed another BNormal Form called Boyce-Codd Normal Form (BCNF).

The fourth and fifth normal forms are based on multi-value and join dependencies.

Normalization results in decomposition of the original relation based on principles such as functional dependence that ensures that the original relation may be re-composed from the decomposed relation, if and when required.

The First Normal Form (1NF)

A relation is in 1NF if

(i) There are no duplicate tuples in the relation.
(ii) Each data value stored is single valued
(iii) Entries in an attribute are of same type.
(iv) The relation must have a key.

The relation

STUDENT(Roll, name, addr, cno, cname, teacher, office) is in 1NF. The primary key in the relation is (Roll, cno).

The second Normal Form (2NF)

A relation is in 2NF if it is in 1 NF and every non-key attribute is fully dependent on each candidate key of the relation.

In the above relation

STUDENT(Roll, name, addr, cno, cname, teacher, office)

The FDs can be written as:

Roll → Name, addr ……..(1)
Cno → cname, Teacher ……..(2)
Teacher → office ……..(3)

The key attribute of the relation are (Roll, cno). Rest attributes are non-key attributes.

We see that FD(1) and FD(2) relate to partial dependence on the key (Roll + cno). Hence they are not in 2NF and hence will suffer from all the anomalies.

To convert it to 2NF, let us use the FDs.

As per FD(1), the Roll uniquely determines Name and Addr.

So one relation should be

STUDENT1(Roll#, Name, addr)

Now in FD(2)

Cno→ Teacher …….(2)
Teacher → office …….(3)

➔Cno → office (Transitive dependency)

Therefore FD(2), can be written as:

Cno → cname, teacher, office …… (2a)

This FD now give the 2nd decomposed relation

COU_TEACHER(cno, cname, teacher, office)

We cannot rejoin these two relations into the original, unless we create a joining relation.

Therefore the relation STUDENT in 2NF form would be:

STUDENT1(roll, name, addr)…….2NF(a)

COU_TEACHER(cno, cname, teacher, office)……. 2NF(b)

COU_STUD(roll, cno) …….2 NF(c)

Third Normal Form (3NF)

A relation is in 3rd Normal Form, if it is in 2 NF and every non-key attribute of the relation is non-transitively dependent on each candidate key of the relation.

Thus the relation STUDENT in 3rd NF would be:

STUDENT1(roll, name, addr)
COURSE(cno, cname, teacher)
TEACHER(teacher, office)
COU_STUD(roll, cno)