

Normalization - I

Gagan Deep

Founder & Director

Rozy Computech Services

Kurukshetra , rozygag@yahoo.com

www.rozyph.com

Contents of Discussion

Today's our topics of Discussion are

- **What Normalization is?**
- **Functional Dependencies**

BASICS – You must know

- **My topics Normalization is of Relational Database. Before Discussing them I am just briefing the following topics**
- **Relation?**
- **Entity ?**
- **Keys?**

What a Relation is?

A Table of **Employee** Database

EmpIDNo.	Name	DOB	Department	Salary
101	Mukesh	10-06-72	Mathematics	5000
102	Rajesh	02-08-67	Physics	5600
103	Mohan	01-12-70	Chemistry	6200
104	Rajesh	12-01-71	Mathematics	4200

Relation(Table), **Entity(Employee)**, **Attributes(First Row)**, Attribute Values(Remaining Table Values), Record (Each Row in a Table except attributes), Tuple(Any Record or segment of Record), **Domain(Like Yellow)**, Primary Key(EmpIDNo.), Secondary Key(Any other attribute)

We Discussed in Last Table

- Relation
- Entity
- Attribute
- Attribute Values
- Record
- Tuple
- Domain
- Primary Key
- Secondary Key

Keys

Super Key : A *super key* is a primary key for a relation is a set of attributes that uniquely identifies a tuple.

Candidate Key : A *candidate key* is a super key such that no proper subset is a super key.

EmpIDNo.	Name	DOB	Department	Salary
101	Mukesh	10-06-72	Mathematics	5000
102	Rajesh	02-08-67	Physics	5600

Foreign Key

- A foreign key is an attribute of a relation that is the primary key of another relations. We will call the relation in which the attribute is the primary key is home relation.

(a) Game is Foreign Key here


Rollno	Name	Game
101	Devesh	Tennis
102	Rajesh	Cricket
103	Ravinder	0
104	Tilak	Swim
105	Suman	Football
106	Saminder	Tennis

b) Game is Primary Key

Game	Fees
Tennis	100
Cricket	150
Swim	200
Football	175
Handball	225
Basketball	265

Our Concern

- In this lecture we discuss some of the theory that has been developed to attempt to choose **GOOD** relations or relational models or relational schemas - that is to measure formally why one set of grouping of attributes into relations is better than another.
- There are two levels at which we can discuss the **goodness** of relations.

- 
- The first is the **logical level**, which refers to how the users interpret the relations and meaning of other attributes. Having good relations at this level helps the users to understand clearly the meaning of the data tuples in the relations, and hence to formulate their queries correctly.

- The second is the **manipulation (or storage) level**, which refers to how the tuple in a base relation are stored and updated.
- This level applies only to base relations – which will physically stored as files- whereas at the logical level we are interested in both base relations and views.

OBJECTIVES OF NORMALIZATION

- The basic objective of logical modeling is to develop a good description of the data, its relationships, and its constraints. For the relational model, **this means that we must identify a suitable set or relations.** However, the task of choosing the relations is a difficult one, because there are many options for the designer to consider. The techniques presented here are based on a large body of research in to the logical design process called *Normalization*.

- The purpose of *normalization* is to produce a stable set of relations that is a faithful model of the operations of the enterprise. By following the *principles of normalization*, we can achieve a design that is highly flexible, allowing the model to be extended. We will begin our study of normalization with a discussion of modification anomalies. Then we will examine relationship among attributes. These relationships help to determine which attributes (or data items) belong together in the same relation. Following that, we will discuss each of the *normal forms*.

Modification Anomalies

- **A design that satisfies user needs is better than one that does not, but that does not mean that any relation is a well structured one.**
- **Changing the data in some relations can have undesirable consequences called modification anomalies.**
- **Anomalies can be eliminated by changing the structure of the relation.**

Explanation by Example

- Consider the GAME relation in Table. The attributes are **ROLLNO** , **GAME**, and **FEE**. The meaning of a row that a student engages in the named **GAME** for the specified **FEE**.

Table : Game Relation , Key : ROLLNO

ROLLNO	GAME	FEE
101	CRICKET	500
110	FOOTBALL	150
135	SWIMMING	100
180	TENNIS	300

Deletion Anomaly

- It is supposed that each GAME has a fixed FEE and that is same for all students.
- If we delete the **tuple for the student 101**, we lose not only the fact that **student 101 is CRICKETER**, but also the fact that **cricket costs Rs.500**.
- This is called a deletion anomaly. In this deleting the facts about one entity (**that student 101 is a CRICKETER**), we inadvertently **deleted** facts about another entity (**that cricket costs Rs.500**).

Explanation by Example

- We loose facts about two entities with one deletion.
- This is known as **Deletion Anomaly**

ROLLNO	GAME	FEE
	CRICKET	500
110	FOOTBALL	150
135	SWIMMING	100
180	TENNIS	300

Insertion Anomaly

- If we want to store the fact that WRESTLING costs Rs.125.
- We cannot enter this data into the GAME relation until a student takes up wrestling as a game.
- This relation is called having an *insertion anomaly* because we cannot insert a fact about one entity until we have an additional fact about another entity.

Explanation by Example

Table : Game Relation , Key : ROLLNO

ROLLNO	GAME	FEE
101	CRICKET	500
110	FOOTBALL	150
135	SWIMMING	100
180	TENNIS	300
	WRESTLING	125

How we can eliminate Anomalies ?

- We can eliminate these anomalies by dividing the GAME relation into two relations, each dealing with a different theme.
- For example, we can put the ROLLNO and GAME attributes into one relation (**S-GAME**), the Game and Fee attributes into a relation called **G-FEE**.

ROLLNO	GAME
101	CRICKET
110	FOOTBALL
135	SWIMMING
180	TENNIS

GAME	FEES
Cricket	500
FOOTBALL	150
SWIMMING	100
TENNIS	300

Functional Dependencies

- The concept of functional dependency is the fundamental basis of normalization.
- A functional dependency is a relationship between two attributes of the same relational database table. One is called the **Determinant** and the other is called **determined**.

Name	Address	DOB	Age

- We can say that given the value of one attribute, we can obtain the value of another attribute.
- For example, if we know the value of **Acc_No**, we can obtain the value of **Acc_Balance**. We say that *Acc_Balance is functionally dependent on Acc_No.*
- In general terms If **A** is a determinant and **B** is determined then we may say that A functionally

determines **B** and represented as **A** \rightarrow **B** “ in the following table there is unique value of **B** for every **A** so we can say that **B** is functionally dependent on **A**

A	B
2	4
3	9
4	16

Features of FD

1. If $X \rightarrow Y$ then it not necessary that $Y \rightarrow X$ i.e. relation is non transitive.
2. If an attribute acts as a Primary Key the all other attributes will functionally depend on it.

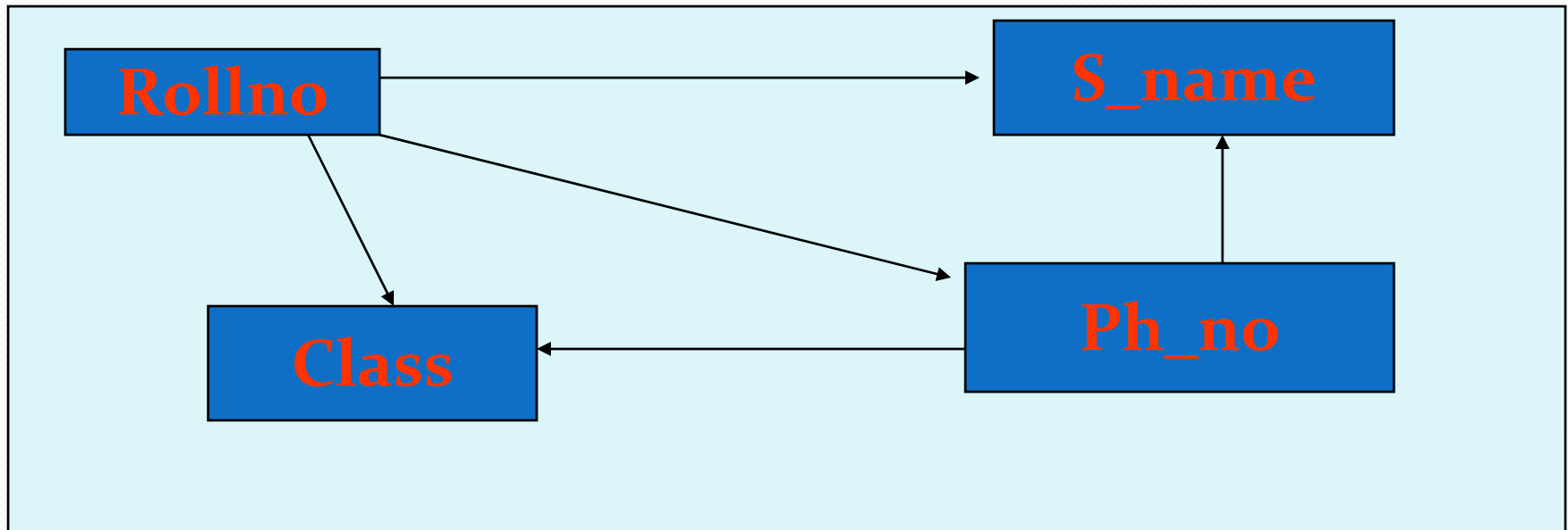
- So we can say that Every Primary Key is determinant for all other attributes.

Example

- Student(Rollno,S_Name,Class,Ph_no)
- In this table **Rollno** is **PK** in this table **S_Name, Class** and **Ph_no** functionally depend on **Rollno** because it unique for all student.

Rollno	S_Name	Class	Ph_no
2000	Suresh	MCA	225070
2001	Parkash	M.Com	215070
2002	Kuldeep	B.Tech	225070

- Some attributes may not be functionally dependent upon others. e.g. **Ph_no** is not **FD** on **class** and **S_name** etc. These functional dependencies can also be represented using the dependency diagram



Fully Functional Dependency

- Fully functional dependency is applicable to those tables where primary key is composite of more than two attributes
- Let us consider an example Library order with attribute order_no, B_name, qty and cost. Here PK is composed of two attribute (**order_no, B_name**). Here qty and cost do not depend on a single attribute. These will be shown as

(order_no , B_name) → Qty

(order_no , B_name , Qty) → cost

- Full functional dependency indicates that if A and B are attributes of a relation, B is fully functionally dependent on A if B is functionally dependent on A, but not on any proper subset of A.
- A functional dependency $A \rightarrow B$ is **partially dependent** if there is some attributes that can be removed from A and the dependency still holds.

Trivial and Non Trivial Dependencies

1. Trivial FD

If R.H.S. is a subset(not necessarily a proper subset) of L.H.S. then it is called trivial dependency.

e.g. $(A,B) \rightarrow A$ Here A is a subset of (A,B) so it is a trivial dependency.

2. Non Trivial FD

If a FD is not trivial it is called non trivial FD

We are generally more interested in **nontrivial dependencies** because they represent integrity constraints for the relation

Armstrong's Axioms of FD

1. Reflexive FD

if **B** is a subset of **A** then **B** functionally depend upon **A** i.e. $A \rightarrow B$

2. Augmentation

if $A \rightarrow B$ then $AC \rightarrow BC$

3. Transitivity

if $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$

4. *Decomposition Rule*

If $A \rightarrow BC$ then $A \rightarrow B$ & $A \rightarrow C$

5. *Union Rule*

If $A \rightarrow B$ & $A \rightarrow C$ then $A \rightarrow BC$

6. *Pseudo transitive Rule*

If $A \rightarrow B$ and $DB \rightarrow C$ then $DA \rightarrow C$

Thanks !

If you have any query please mail me at

rozygag@yahoo.com

www.rozyph.com