THE RELATIONAL MODEL & NORMALISATION (Chapter -8)

1) Who developed relational model Rules?

- **E.F.Codd** developed relation model rules.

2) Write all the Codd's rules ?

- **Rule 1 – The Information Rule –** All data should be presented to the user in a tabular form.
- **Rule 2 – Guaranteed Access Rule –** All data should be accessible without ambiguity. This can be accomplished through a combination of the table name, primary key & column name.
- **Rule 3 – Systematic Treatment of Null Values –** A filed shoul be allowed to remain empty. This cannot apply to primary keys.
- **Rule 4 – Dynamic On-Line Catalogue Based on the Relational Model –** A relational DB must provide access to its structure through the same tools that are used to access the data. This is usually accomplished by storing the structure definition within special system tables.
- **Rule 5 – Comprehensive Data Sublanguage Rule –** The DB must support at least one clearly defined language that includes functionality for data definition, data manipulation, data integrity & DB transaction control. SQL (Structure Query Language) is that comprehensive language.
- **Rule 6 – View Updating Rule –** Data can be presented to the users in diff. logical combinations, called views. Each view should support the same full range of data manipulation that directly access to a table has available.
- **Rule 7 – High level insert, update & delete –** Data can be retrieved from a relational DB in sets of data from multiple rows &/or multiple tables. This rule states that insert, update & delete operations should be supported for any row/column or table rather than single row in a single table.
- **Rule 8 – Physical Data Independence –** The user is isolated from the method of storing & retrieving information from database. Changes made to the underlying architecture without affecting how user accesses.
- **Rule 9 – Logical Data Independence –** How user views data should not change when the logical structure of the database changes.
- **Rule 10 – Integrity Independence –** The SQL(Database language) should support constraits on user input that maintain database integrity.
- **Rule 11 – Distribution Independence –** A user should be totally unaware of whether or not the DB is distributed (i.e. whether a part of the DB exist in multiple location).
- **Rule 12 – Non-subversion Rule –** There should be no way to modify the DB structure other than through the multiple row DB language like SQL.

3) Short Notes : -

i) **Relation –** It is a 2-D table consisting of row& columns, in which data can be stored in a tabular format. Each row in a table stores set of related values & each column stores property of that row.

ii) **Tuple –** Each row of the table in the relational model is called a tuple. A single tuple contains the data for a object. If we want a proper relation, duplicate tuples are not allowed i.e no rows can be identical in all respect in a relation.

iii) **Attribute –** Each column in the table is called an attribute. An attribute has a unique name indicating a particular characteristic of the object.

iv) **Cardinality –** The no. of tuples in a particular relation/table is called cardinality of the relation/table.

v) **Degree -** The no. of attributes in a particular relation/table is called degree of the relation/table.

vi) **Domain –** Each attribute in a relation has value which it can take from a set of permitted values for that attribute.

4) What are the diff. types of Key in DBMS ?

- In DBMS, Key is used to store data uniquely within a attribute or column, so data can be identify within a tuple or row.
  - ✓ There are many types of Key is there in DBMS i) Primary Key ii) Foreign Key iii) Super Key iv) Candidate Key v) Alternate Key vi) Record Key.
  - ✓ **Primary Key –** Within a table/relation tuple/row are identified & can be used to relationship between more than one table the key which is used in known as Primary Key.

Stu_Id	Stu_Name	Class	Sec	Ph_No,
Sc-01	Srijan Pal	XII	A	9652421875
Com-01	Priti Nag	XII	A	9785412457
Art-01	Satabdi Chakraborty	XII	A	9875487895
SC-02	Pritam Roy	XII	A	7848575142

Here STUDENT is a table/relation where **if we want to create primary key Stu_Id is a attribute i.e Stu_Id has to be unique** another **feather is that if we create a column as primary key that particular columns can not be null.**

- ✓ **Candidate Key –** Within a relational DBMS, within a table to identify each tuple or record Candidate key is used. **Within candidate key all the primary key's properties can be there.** Through candidate key we can easily identify a single selected tuple/record. Within a table multiple attribute can be used as a candidate key. Here within those candidate key one attribute has to be Primary Key.

Reg_No.	Certificate_No.	Exam_Passed	Year	Marks_Percentage
4300310-0010	MP-0052	M.P.	2002	82
60104060102	HS-92008	H.S.	2004	75
4300310-0100	MP-0059	M.P.	2008	81
60104060119	HS-91009	H.S.	2008	72

In this above mentioned Exam_Register table Reg_No. & Certificate_No. those attribute or columns are candidate key because these 2 attributes are unique.

- ✓ **Alternate Key –** Within a table there may be many candidate key but one key has to be used as a primary key , to omit that primary key rest of the candidate keys are known as alternate key.

PRANAB SAHA

- ✓ **Super Key –** Super Key is a collection of many columns/attribute within a table, in which values of that particular columns can be used to identify a tuple/row.

Roll	Sec.	Name	Address
1	Sc	Srijan Pal	Rajarhat
2	Sc	Sayan Roy	Kestopur
3	Com	Joyee Pal	Saltlake
4	Arts	Satabdi Chakraborty	Saltlake

In the above table we can make the probable super key as follows –
[Roll, Sec.]
[Roll, Sec., Name]
[Roll, Sec., Name, Address]
[Roll, Name]
[Name, Address]
[Name, Sec] etc.

- ✓ **Record Key –** In a DBMS table to differentiate a particular record an unique characteristic or attribute can be selected this is known as Record Key. eg. – In bank cheque no, or in student table roll no. is a record key.
- ✓ **Foreign Key –** in a Database a tables primary key can be present in another table's common attribute, then that particular attribute/column is called foreign key of the 2nd table.

Student Table

Reg_No.	**Name**	**Course**	**Status**
R-000125	Soumya Ghosh	Sc.	Passed
R-001264	Koel Mallick	Sc.	Passed
R-001234	Riya Das	Com	Passed

Batch Table

Roll_No.	**Address**	**Year_of_Passing**	**Reg_No.**
001	Saltlake	2018	R-000125
002	Kestopur	2018	R-001264
003	Saltlake	2019	R-001234

Here, in Student table Reg_No. is primary key. But Reg_No. is a attribute/field in Batch table which is here used a foreign key.

5) Diff. betwn Primary Key & Candidate Key ?

Primary Key	**Candidate Key**
1. Within a table there must be a primary key.	1. Within a table there must be many candidate key.
2. To identify, search & establish relationship primary key is used.	2. To search record candidate key is used.
3. DBMS can individually create primary key.	3. We can not individually create Candidate key.
4. A primary key can be used as a Candidate key.	4. Within all candidate key only one key can be used as a primary key.

6) Diff. betwn Primary Key & Foreign Key ?

Primary Key	**Foreign Key**
1. A primary key of one table can be used as foreign key of another table.	1. A foreign key of one table must be primary key of another table.
2. To identify unique data Primary key is used.	2. To avoid inconsistency foreign key is used.

7) Diff. betwn Super Key & Candidate Key ?

Super Key	**Candidate Key**
1. Set of Attribute or group of attribute/column is known as Super Key,	1. Within Super key lowest or minimal key attribute is known as candidate key.
2. Subset is made up of each Super Key.	2. Subset is not made up of Candidate Key.
3. In super key there is not feathers of Primary Key.	3. In Candidate key there is feathers of Primary Key.
4. All Super Key are not Candidate key.	4. All Candidate Key's are Super Key.
5. Here, all the records are differentiately identified.	5. Here, all the records are uniquely identified.

8) What is Constraints in Relational Model ?

- ➢ Within a DBMS to entry valid data range there are few rules. These rules to enter valid data in a DBMS is known as Constraints.
  Through this constraints we can enter valid data within the DBMS as well as we can avoid invalid data to enter into the DBMS.
  There are few Constraints are there in DBMS –
  i) Key Constraints - In this case within a table/relation tuple or record must be unique or distinct , in each relation/table minimum one super key has to be there & 2 tuples value can't be same here.

PRANAB SAHA

ii) Data Dependencies – In relational database Data dependencies is a situation where in a table one attribute/column's value is dependent on the earlier attribute's value.

9) What is Intigrity Constraints in DBMS ?

➢ Intigrity means right value or consistent value & Intigrity related rules are known as Intigrity Constraints. There are 4 types of Intigrity Constraints in DBMS –

i) Entity Integrity Constraints ii) Domains Integirty Constraints iii) Referential Integrity Constraints iv) Foreign Key Integirty Constraints.

✓ **Entity Integirty Constraints –** Here, within a table a tuple/record where primary key lies value can't be null. Because if there a null value for the record then the record can not be identified either. Thus, in a relational table except primary key all the other field can be null value i.e 0 or we can use space as a value.

Table Student

Regn_No	**Name**	**Year**
43003012	Sonali	2018
43003013	Bapan	2018
43003014	Pritam	2018

Table Status

Regn_No	**Status**
43003012	Passed
43003013	Passed
43003014	

Here, Student & Status tables are connected 1:1 relationship we can see that the student whose name is Pritam , his status is null so we can't get all the information of that student, But through Entity Integrity Constraints we can easily identify all the other row/record.

✓ **Domain Integrity Constraints –** Through relational table each attribute or column's value must be valid. So suppose we can not insert text data into a number type datatype attribute.
In a domain constraints the following topics must be followed :-
i) Data type or type of data like – number, text,data/time etc.
ii) Length of attribute like- single, double,float).
iii) Size of string data can't be fixed.
iv) Null value can be allowed.
Also we can set a default value of an attribute, or we can specify range of an attribute etc.
So, We can insert data in a right & proper manner in a Database through domain integrity constraints.

✓ **Referential Integrity Constraints –** To keep consistency between 2 table/relation we use referential integrity constraints. Here, a relation/table's one tuple which will relation with another table also has to be present in the 2nd table.

Table – Customer

Cust_Id	**Cust_Name**
001	Priti
002	Joyee
003	Sayani

Table – Order

Order_Id	**Cust_Id**	**Order_Date**
A-01	001	19/04/2017
B-01	002	20/04/2017
D-01	004	21/04/2017

Here, Cust_Id of Order table is referenced with Cust_Id of Customer table. So, tuple(Cust_Id) of Order table is related with tuple(Cust_Id) of Customer Table.
Here, Cust_Id of Customer table is primary key & Cust_Id of Order table must be Foreign Key.

✓ **Advantage of Referential Integrity Constraints –**
i) If there is connection between 1st table & 2nd table then any record between these 2 table can't be deleted.
ii) Primary key's value of 1st table can't be change , if there is a relationship between 1st & 2nd table.
iii) In 2nd table where foreign key is connected we can't enter any data, if this data exist in the 1st table as primary key.
iv) In this case, foreign key's data can be null.

✓ **Foreign Key integrity Constraints –** To affect the referential integrity constraints foreign key integrity is used, There are 2 types of Foreign key integrity constraints –
i) Cascade Update related field.
ii) Cascade Delete related rows.
i) Cascade Update related field – In this case, 1st table's any row's primary key's value can be change, so automatically 2nd table's matching row's foreign key's value will also be change.

Model_Id	**Model_Name**	**Rate**
1	Maruti Omni	2,13,615
2	Maruti Alto	2,54,354
3	Maruti Wagon	3,71,625
4	Tata Sumo	5,75,246

Table- Car

Regn_No	**Model_Id**	**Year**
R001	1	2012
M002	1	2010
M003	2	2009
T004	3	2008

Table - CarRegn

Here, 2 table(Car & CarRegn) , if we can establish a cascade update constraints then we can change the data of Car table's primary key Model_Id automatically Model_Id of CarRegn table will change.
ii) Cascade Delete related rows – In this case, we can delete a row from the 1st table when there is a relationship between 2 table & automatically 2nd tables matching row will be deleted.
Here, 2 table(Car & CarRegn) , if we can establish a cascade delete constraints then if we delete a row suppose Maruti Alto's data of Car table's primary key automatically related row will be deleted from CarRegn table.

PRANAB SAHA

Normalization of Database

Database Normalisation is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anamolies. It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables.

Normalization is used for mainly two purpose,

- ✓ Eliminating reduntant(useless) data.
- ✓ Ensuring data dependencies make sense i.e data is logically stored.

- ✓ Anomalies in DBMS.
- ✓ There are three types of anomalies that occur when the database is not normalized. These are – Insertion, update and deletion anomaly. Let's take an example to understand this.
- ✓ Example: Suppose a manufacturing company stores the employee details in a table named employee that has four attributes: emp_id for storing employee's id, emp_name for storing employee's name, emp_address for storing employee's address and emp_dept for storing the department details in which the employee works. At some point of time the table looks like this -

EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_DEPT
101	RICK	DELHI	D001
101	RICK	DELHI	D002
123	MARIE	AGRA	D003
166	PHILIP	CHENNAI	D900
166	PHILIP	CHENNAI	D004

- The above table is not normalized. We will see the problems that we face when a table is not normalized.
- **Update anomaly:** In the above table we have two rows for employee Rick as he belongs to two departments of the company. If we want to update the address of Rick then we have to update the same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Rick would be having two different addresses, which is not correct and would lead to inconsistent data.
- **Insert anomaly:** Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.
- **Delete anomaly:** Suppose, if at a point of time the company closes the department D900 then deleting the rows that are having emp_dept as D900 would also delete the information of employee PHILIPS since she is assigned only to this department.
- To overcome these anomalies we need to normalize the data. In the next section we will discuss about normalization.

Problem Without Normalization

Without Normalization, it becomes difficult to handle and update the database, without facing data loss. Insertion, Updation and Deletion Anamolies are very frequent if Database is not Normalized. To understand these anomalies let us take an example of Student table.

S_ID	S_NAME	S_ADDRESS	SUBJECT_OPTED
401	AKASH	NOIDA	BIOLOGY
402	ARKADEEP	LUCKNOW	MATHS
403	SAURAV	JAMMU	MATHS
404	AKASH	NOIDA	PHYSICS

PRA

- **Updation Anomaly :** To update address of a student who occurs twice or more than twice in a table, we will have to update S_Address column in all the rows, else data will become inconsistent.
- **Insertion Anomaly :** Suppose for a new admission, we have a Student id(S_id), name and address of a student but if student has not opted for any subjects yet then we have to insert NULLthere, leading to Insertion Anomaly.
- **Deletion Anomaly :** If (S_id) 401 has only one subject and temporarily he drops it, when we delete that row, entire student record will be deleted along with it.

Normalization Rule

Normalization rule are divided into following normal form.

1. **First Normal Form**
2. **Second Normal Form**
3. **Third Normal Form**
4. **BCNF**

First Normal Form (1NF)

As per First Normal Form, no two Rows of data must contain repeating group of information i.e each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that distinguishes it as unique.

The Primary key is usually a single column, but sometimes more than one column can be combined to create a single primary key. For example consider a table which is not in First normal form

Student Table :

STUDENT	AGE	SUBJECT
AKASH	15	Biology, Maths
ARKADEEP	14	Maths
SANDEEP	17	Maths

In First Normal Form, any row must not have a column in which more than one value is saved, like separated with commas. Rather than that, we must separate such data into multiple rows.

Student Table following 1NF will be :

STUDENT	AGE	SUBJECT
AKASH	15	BIOLOGY
AKASH	15	MATHS
ARKADEEP	14	MATHS
SANDEEP	17	MATHS

Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

Second Normal Form (2NF)

As per the Second Normal Form there must not be any partial dependency of any column on primary key. It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails **Second normal form.**

In example of First Normal Form there are two rows for Adam, to include multiple subjects that he has opted for. While this is searchable, and follows First normal form, it is an inefficient use of space. Also in the above Table in First Normal Form, while the candidate key is {**Student, Subject**}, **Age of Student only depends on Student column, which is incorrect as per Second Normal Form.** To achieve second normal form, it would be helpful to split out the subjects into an independent table, and match them up using the student names as foreign keys.

New Student Table following 2NF will be :

STUDENT	AGE
AKASH	15
ARKADEEP	14
SANDEEP	17

In Student Table the candidate key will be Student column, because all other column i.e Age is dependent on it.

New Subject Table introduced for 2NF will be :

STUDENT	SUBJECT
AKASH	BIOLOGY
AKASH	MATHS
ARKADEEP	MATHS
SANDEEP	MATHS

In Subject Table the candidate key will be {**Student, Subject**} column. Now, both the above tables qualifies for Second Normal Form and will never suffer from Update Anomalies. Although there are a few complex cases in which table in Second Normal Form suffers Update Anomalies, and to handle those scenarios Third Normal Form is there.

Third Normal Form (3NF)

Third Normal form applies that every non-prime attribute of table must be dependent on primary key, or we can say that, there should not be the case that a non-prime attribute is determined by another non-prime attribute.

So this transitive functional dependency should be removed from the table and also the table must be in Second Normal form. For example, consider a table with following fields.

Student_Detail Table :

STUDENT_ID	STUDENT_NAME	DOB	STREET	CITY	STATE	ZIP

In this table Student_id is Primary key, but street, city and state depends upon Zip. The dependency between zip and other fields is called transitive dependency. Hence to apply 3NF, we need to move the street, city and state to new table, with Zip as primary key.

New Student_Detail Table :

Student_id	Student_name	DOB	Zip

Address Table :

Zip	Street	City	state

The advantage of removing transtive dependency is,

- Amount of data duplication is reduced.
- Data integrity achieved.

PRANAB SAHA

Boyce and Codd Normal Form (BCNF)

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

- R must be in 3rd Normal Form
- and, for each functional dependency (X → Y), X should be a super Key.

Consider the following relationship : **R (A,B,C,D)**

and following dependencies :

A -> BCD
BC -> AD
D -> B

Above relationship is already in 3rd NF. Keys are **A** and **BC.**

Hence, in the functional dependency, **A -> BCD**, A is the super key.
in second relation, **BC -> AD**, BC is also a key.
but in, **D -> B**, D is not a key.

Hence we can break our relationship R into two relationships **R1** and **R2**.

$$\begin{tikzcd}
 & R(A, B, C, D) \arrow[dl, dash] \arrow[dr, dash] & \\
R1 (A, D, C) & & R2(D, B)
\end{tikzcd}$$

Breaking, table into two tables, one with A, D and C while the other with D and B.

PRANA