Database Management System

Unit - 3

Q1. CODD's Rules.

→ A database must obey these rules in order to be ~~Re~~ regarded as a true Relational-Database.

1 1 Rule 0: Relational Database Management.

- Any truly relational database must be managable entirely through its own relational capabilities.

2. Rule 1: The information rule:

- All information stored in a relational database is represented only by data item values, which are stored in the tables that make up the database.

- In simple terms, this means that if an item of data doesn't reside in a table in the database then it doesn't exist.

3. Rule 2: The Rule of Guaranteed Access:

- Each and energy data in r-db is guaranteed to be logically accessible by restoring to a combination of table name, primary key value & column value

4. Rule 3: Representing of null values:

- The DBMS has a consistent method for representing null values.
- For e.g., null values for numeric data must be distinct from zero or ony other numeric value and for character data it must be different from a string of blancks or any other character value.

5. Rule 4: The db description rule:
- Rule 4 states that there must be data dictionary within the RDBMS that is constructed of tables and/or views that can be examined ~~using~~ using SQL.

6. Rule 5: The comprehensive sub language rule:
- There must be at least one language ~~whose~~ which clearly define and can be expressed as character strings conforming to some well defined syntax, that is comprehensive in supporting of the following:

- Data Definition
- Integrity constraint.
- View definition
- Authorization
- Data Manipulation
- Transaction boundries.

- This means that the RDBMS must be completely managable through its own dialect of SQL.

7. Rule 6: The view updating rule:
- All views that can be updated in theory can be updated by the system.

8. Rule 7: The insert and update rule:
- The capability of handling a base relation or infact a derived relation as a single operand must hold good for all retrieve, update, delete and insert activity.

Rule 8: Physical data independance rule.
- Changes made to physical storage representations or access methods do not require changes to be made to application programs.

Rule 9: The Logical Data Independance Rule:
- Application programs and terminal activities must remain logically unimpaired whenever information preserving changes of any kind that are theoretically permitted are made to the base table.

Rule 10: Integrity Independance Rule:
- All integrity constraints defined for a database a null be definable in the language stored in the database as data in tables.

- The following integrity rules should apply to every R-DBMS.

- Entity Integrity: No component of a primary key can be have missing values.

- Referential Integrity: For each distinct foreign key value there must exist a matching primary key value in the same domain.

Rule 11: Distribution independance.
- The distribution of database at various locations should not be visible to end user.
- Users should always get the impression that the data is located at one site only.

- That means even though the data is distributed, it should not affect the speed of access and performance of data compared to centralized database.

13. Rule 12: No subversion Rule:
- If an RDBMS supports a lower level language that permits for example row-at-a-time-processing, then language must not be able to bypass any integrity rules or constraints defined in the higher level, set-at-a-time, relational language.

Q2 What is Normalization?
What are the anomalies?
What is the impact of insert, update and delete anomaly on overall design of database?
How normalization is used to remove these anomalies?
Explain 1NF, 2NF, 3NF, BCNF & 4NF.
Enlist the differences.
What is need of Normalized database?

→ Normalization of database :-

- Database Normalization is a technique of organizing the data in such a manner that it should reduce redundancy and dependency of data.

- Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like insertion, update and deletion anamolies.

- It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.

- Normalization is used for mainly two purposes:-

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

- Problems without Normalization i.e. Anamolies :-

- If a table is not properly normalized and have data redundancy then it will not only eat up extra memory space but will also make it difficult to handle and update the database, without facing data loss.

- Insertion, updation and deletion anomalies are very frequent if database is not normalized. ₹

- To understand these anomalies let us take an example of a Student table -

rollno	name	branch	hod	office_tel
301	abc	comp	Mr. Z	88889
302	pqr	comp	Mr. Z	88889
303	lmn	comp	Mr. Z	88889
304	xyz	comp	Mr. Z	88889

- In the above table, we have 4 students of computer branch. As we see, data for the fields branch, hod & office-tel is repeated for the students who are in the same branch in the college, this is Data Redundancy.

1. Insertion Anomaly:

- Suppose for a new admission, until and unless a student opted for a branch, data of the student connot be inserted or else we will have to set the branch information as NULL.

- Also, if we have to insert data of 100 students of some branch, then the branch information will be repeated for all those 100 students.

- These scenarios are nothing but insertion anomalies.

2. Updation Aonamaly:

- What if Mr. Z leaves the college? or is no longer the HOD of computer department? In that case all the student records will have to be updated, and if by ~~ma~~ mistake we miss any record, it will lead to data inconsistency. This is updation anomaly.

3. Deletion Anomaly:

- In our student table, two different informations are kept together, Student information and Branch informat

- Hence, at the end of the academic year, if student records are deleted, we will also lose the branch information. This is deletion anomaly.

- To remove above anomalies from the database we required normalization techniques.

Normalization Rules:-

1. 1NF - First Normal Form
2. 2NF - Second Normal Form
3. 3NF - Third Normal Form
4. BCNF - Boyce - Codd Normal Form
5. 4NF - Fourth Normal Form

1 1NF - First Normal Form :-

- For a table to be in the First Normal Form, it should follow the following 4 rules:

1. It should only have single (atomic) valued attributes/columns.
2. Values stored in a column should be of the same domain.
3. All the columns in a table should have unique names.
4. And the order in which data is stored, does not matter.

- e.g. Create a table to store student data which will have student's roll no, their name & the name of subjects they have opted for.

roll_no	name	subject
1	abc	OS, CN
2	lmn	Java
3	pqr	C, C++

- Our table already satisfies 3 rules out of the 4 rules, as all our column names are unique, we have stored data in the order we wanted to and we have not inter-mixed different type of data in columns

- BW out of the 3 different students in our table, 2 have opted for more than 1 subject, and we have stored the subject names in a single column. But as per INF each column must contain automic value

How to solve this problem?

- We have to do is break the values into automic values

- So the updated table be:

roll_no	name	subject
1	abc	OS
1	abc	CN
2	lmn	Java
3	pqr	C
3	pqr	C++

- By doing so, although a few values are getting repeated but values for the subject column are now atomic for each record/row.

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

2 2NF - Second Normal Form:

- For a table to be in the second normal form, it must satisfy following 2 rules:

1. The table should be in the first Normal ~~Fo~~ Form
2. There should be no Partial Dependency.

- e.g. create a table student, a single column like student_id can uniquely identify all the records in a table.

- But this is not true all the time. So let's extend the above example to see if more than 1 column together can act as a primary key

- So, let's create another table for subject, which will have subject_id and subject_name field and subject_id will be the primary key

subject_id	subject_name
1	Java
2	C++
3	Python

- Now we have a student table with student information and another table subject for storing subject information.

- So, create another table named as score, to store the marks obtained by students in the respective subjects. We will also be giving name of subject teacher who teaches that subject along with marks.

score_id	student_id	subject_id	marks	teacher
1	10	1	80	Java teacher
2	10	2	85	C++ teacher
3	11	1	90	Java teacher.

- In the above table, we are saving student_id to know which student's marks are these and subject_id to know for which subject the marks are for.

- Together, student_id + subject_id forms a candidate key for this table, which can be the Primary key

- We are using student_id + subject_id as a primary key to uniquely identify any row.

- Now if you look at the score table, we have a column names teacher which is only dependent on the subject, for java it's java teacher and for C++ it's C++ teacher & so on.

- The primary key for this table is a composition of two of columns which is student_id & subject_id but the teacher's name only depends on subject, hence the subject_id and has nothing to do with student_id.

- This is Partial Dependency, where an attribute in a table depends on only a part of the primary key and not on the whole key.

How to remove Partial Dependency?

- There can be many different solutions for this, but our objective is to remove teacher's name from score table

- The simplest solution is to remove columns teacher from score table and add it to the subject table. Hence, the subject table will become:

subject_id	subject_name	teacher
1	Java	Java teacher
2	C++	C++ teacher
3	Python	Python teacher

- Now, our score table is now in the 2NF, with no partial dependency.

score_id	student_id	subject_id	marks.
1	10	1	80
2	10	2	85
3	11	1	90

3. 3NF - Third Normal Form:

- For a table to be in third normal form it must satisfy following 2 rules:

1. The table should be in Second Normal Form.
2. The table should not have transitive dependency.

e.g. Student table:

student_id	name	reg.no	branch	address
10	abc	07-KSOE	Comp	Pune
11	lmn	08-KSOE	Entc	Mumbai
12	par	09-KSOE	Mech	Rajastan

Subject table

subject-id	subject_name	teacher
1	Java	Java teacher
2	C++	C++ teacher
3	Python	Python teacher

Score table:

score_id	student_id	subject_id	marks
1	10	1	70
2	10	2	25
3	11	1	80

- In the above score table, we need to store some more information, which is the exam_name & total_marks so let's add 2 more columns to the score table.

score_id	student_id	subject_id	marks	exam_name	total_marks
1	10	1	70	Mains	70
2	10	2	25	Practicals	30
3	11	1	80	Workshop	200

- With adding 2 new columns to score table, it saves more data now. Primary key for our score table is a composite key, student_id + subject_id.

- New column exam_name depends on both student & subject. e.g. a mechanical student will have examp but computer student won't. And for some subjects you have Practical exams and for some you won't. so can say that exam_name is dependent on student_id & subject_id.

- The column total_marks depends on exam_name as with exam type the total score changes.

- But, exam_name is just another column in the score table. It is not a primary key or even a part of the primary key, and total_marks depends on it.

- This is Transitive Dependency. When a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key.

How to remove Transitive Dependency?

- Take out the column exam_name and total_marks from score table and put them in an Exam table and use the exam_id wherever required, so that score table is in 3NF.

Exam table:

exam_id	exam_name	total_marks
1	Workshop	200
2	Mains	70
3	Practicals	30

Score table (updated):

score_id	student_id	subject_id	marks	exam_id
1	10	1	70	2
2	10	2	75	3
3	11	1	80	1

Advantages of removing Transitive Dependency:

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

4. Boyce - Codd Normal Form (BCNF):

- For a table to satisfy the Boyce - Codd Normal Form, it should satisfy the following two conditions:

1. It should be in the third normal form
2. And for any dependency, A → B, A should be a super key.

- The second point means that for a dependency A → B, A cannot be a non-prime attribute, if B is a prime attribute.

- e.g. College enrolment table

student_id	subject	professor
101	Java	P. java
101	C++	P. cpp
102	Java	P. java2
103	C#	p. Chash
104	Java	P. java

- As we see the above table satisfies all the conditions of 1NF, 2NF & 3NF.

- In the above table, student_id, subject together form the primary key, because using student_id and subject, we can find all the columns of the table

- There is one more important point to note here is, one professor teaches only one subject, but one subject may have two different professors.

- Hence, there is a dependency between subject and professor here, where subject depends on the professor name.

- In the above table student_id, subject form primary key, which means subject column is a prime attribute. But there is one more dependency, professor → subject. & while subject is a prime attribute, professor is a non-prime attribute, which is not allowed by BCNF.

How to satisfy BCNF ?

- To make this relation (table) satisfy BCNF, we will decomposes this table into two tables, student table and professor table

- So the updated tables are:-

Student table:

student_id	p_id
101	1
101	2

and so on..

Professor table:

p_id	professor	subject
1	P. java	Java
2	P. cpp	C++

5. 4NF - Fourth Normal Form:

- For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

1. It should be in the Boyce-Codd Normal Form
2. And, the table should not have any Multi-Valued Dependency.

- For a dependency A -> B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.

- e.g. let's create college enrolment table.

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
2	C#	Cricket
2	Python	Hockey

- As you see, in the above table, student with s_id, 1 has opted for two courses, Science and Maths. & has two hobbies, Cricket and Hockey.

- Well, the two records for student with s_id 1, will give rise to two more records as shown below, because for one student, two hobbies exists, hence along with both the courses, these hobbies should be specified

s_id	course	hobby
1	Science	Cricket
1	Science	Hockey
1	Maths	Cricket
1	Maths	Hockey

- And, in the above table, there is no relationship between the columns course and hobby. They are independent of each other.

- So there is multi-value dependency, which leads to un-necessary repetition of data and other anomalies as well.

How to satisfy 4th Normal Form?

- To make the above relation satisfy the 4th Normal Form, we can decompose the table into 2 tables.

Course opted table:

s_id	couse
1	Science
1	Maths
2	C#
2	Python

Hobbies table:

s_id	hobby
1	Cricket
1	Hockey
2	Cricket
2	Hockey

~~A table.~~

3NF	BCNF
1 It stands for third Normal Form	1 It stands for Boyce-Codd Normal Form.
2. A database design should be already in 2NF to convert in 3NF.	2 A database design should be already in 3NF to convert in BCNF
3. Rule: For any non-trivial functional dependency, $X \rightarrow A$ Either X is a superkey or A is prime attribute.	3. Rule: For any non-trivial functional dependency, $X \rightarrow A$, X must be a super-key.
4. 3NF is weaker than BCNF.	4. BCNF is stronger than 3NF.
5. 3NF connot catch all the anomalies.	5. BCNF was developed to capture those anomalies that could not captured by 3NF.
6 More redundancy	6. Less redundancy.
7. Computational time is more.	7. Computational time is less.

4NF	BCNF.
1. It stands for fourth normal form	1. It stands for Boyce-Codd Normal Form
2. A database design should be already in 3NF and BCNF to convert in 4NF	2. A database design should be already in 3NF to convert in BCNF.
3. No multi-valued dependancies exist in the tables.	3. Multi-valued dependencies exist in the tables.
4. The 4NF is more desirable than BCNF because it avoid repetition of data.	4. The BCNF is less desirable than BCNF because it does not avoid repetition of data
5. The decomposition in 4NF does not lead to loss of information when lossless join decomposition is used.	5. This is little bit difficult in BCNF.
6. Reedundancy is less	6. Redundancy is more.