TASK 3

Real time applications of joints

Joins in SQL are commands which are used to combine rows from two or more tables, based on a related column between those tables. There are predominantly used when a user is trying to extract data from tables which have one-to-many or many-to-many relationships between them. Joins in databases are used in real-time applications across various industries and domains to combine data from multiple tables and extract meaningful insights.One such example is ‘Online Booking Systems’.

Joins are used to fetch data from multiple tables to display detailed information about services offered. For instance, in a hotel booking system, joins combine data from separate tables, such as hotel details, room types, amenities, and prices, to present a complete overview of available options to users.

When a user selects a date and time for a service, joins help verify if the requested slot is available. By combining user input with existing booking records, the system can determine if the service can be scheduled for the specified date and time. It also allows the system to manage reservations, display booking history, and send booking confirmations and reminders to users.

Joins link booking records with payment information, enabling the system to verify successful payments, update payment status, and provide users with booking confirmations upon completing the payment process.

Normalization types with examples

1NF

A relation is in 1NF if it contains an atomic value.

Employer ID	Name	Phone No
121	Thomas	9876543219
121	Thomas	8765654329
122	Roshan	7865439009

2NF

A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.

Employer ID	Department ID	Location
121	MD1	Pune
121	MD2	Chennai
122	MD3	Delhi

In this case, **Office Location** only depends on **Department ID**, which is only part of the primary key. Therefore we need to break the table into two parts.

Table 1

Employer ID	Department ID
121	MD1
121	MD2
122	MD3

Table 2

Department ID	Location
MD1	Pune
MD2	Chennai
MD3	Delhi

3NF

A relation will be in 3NF if it is in 2NF and no transition dependency exists.

Student ID	Name	Subject ID	Subject
121	Thomas	67	C
122	Roshan	56	Python
123	Rachel	43	DBMS

In the above table, Student ID determines Subject ID, and Subject ID determines Subject. Therefore, Student ID determines Subject via Subject ID**.** This implies that we have a transitive functional dependency, and this structure does not satisfy the third normal form. Now in order to achieve third normal form, we need to divide the table as shown below:

Table 1

Student ID	Name	Subject ID
121	Thomas	67
122	Roshan	56
123	Rachel	43

Table 2

Subject ID	Subject
67	C
56	Python
43	DBMS

Boyce Codd Normal Form (BCNF)

This is also known as 3.5 NF. It is the higher version 3NF.

Student ID	Subject	Professor
121	C	Prof.Charles
122	Python	Prof.Alex
123	DBMS	Prof.Raj
124	Python	Prof.Alex

As you can see Student ID**,** and Subject form the primary key, which means the Subject column is a prime attribute. But, there is one more dependency, Professor → Subject**.** And while Subject is a prime attribute, Professor is a non-prime attribute, which is not allowed by BCNF.

Now in order to satisfy the BCNF, we will be dividing the table into two parts.

Table 1

Student ID	Professor ID
121	CS1
122	CS2
123	CS3
124	CS2

Table 2

Professor ID	Professor	Subject
CS1	Prof.Charles	C
CS2	Prof.Alex	Python
CS3	Prof.Raj	DBMS

4NF

A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.

Student ID	Subject	Hobby
121	C	Reading
121	Python	Dancing
123	DBMS	Reading

In the given table, subject and hobby are two independent entity. Hence, there is no relationship between subject and hobby. the student relation, a student with Student ID, 121 contains two courses, C and Python and two hobbies, Reading and Dancing. So there is a Multi-valued dependency on Student ID, which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into two tables:

Table 1

Student ID	Subject
121	C
121	Python
123	DBMS

Table 2

Student ID	Hobby
121	Reading
121	Dancing
123	Reading

5NF

A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

Subject	Professor	Semester
C	Charles	1
C	Raj	1
DBMS	Raj	1
DBMS	Edward	2
Python	Alex	1

In the above table, Raj takes both C and DBMS class for Semester 1 but he doesn't take DBMS class for Semester 2. In this case, combination of all these fields required to identify a valid data. Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Professor and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three tables

Table 1

Semester	Subject
1	C
1	DBMS
1	Python
2	DBMS

Table 2

Subject	Professor
C	Charles
C	Raj
DBMS	Raj
DBMS	Edward
Python	Alex

Table 3

Semester	Professor
1	Charles
1	Raj
1	Raj
2	Edward
1	Alex