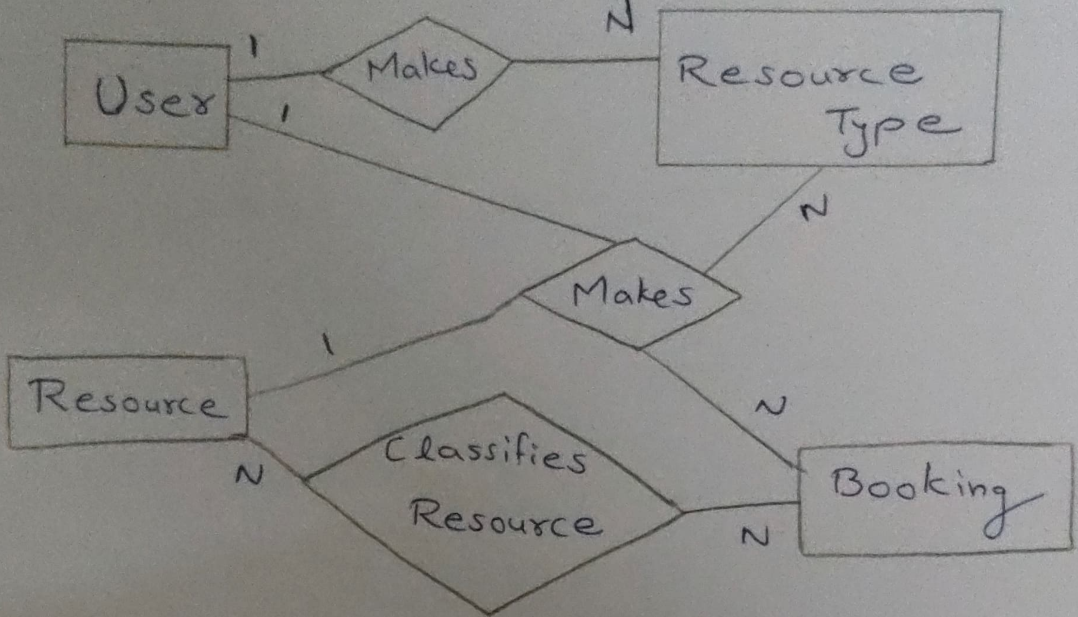
① Smart campus resource booking system design and E-R diagram for resource like classrooms, labs and conference halls.

A:



② Convert the E-R diagram to relational schema and apply normalization

A: Convert the E-R diagram into normalized relational tables

- User : User_Id (pk), name, email
- Resource : resource_id (pk), type, name, location, capacity
- Booking : Booking_Id (PK), User_Id (FK), resource_Id (FK), Start_time, end_time, status.
- Payment : payment_Id (PK), booking_Id (FK),

amount, date, status

Normalization (Up to 3NF):

- All non-key attributes are fully dependent on the primary key.
- Remove partial and transitive dependencies

③ SQL queries for room availability and manage bookings

A SQL query to check Room availability

```
SELECT   B.Booking_ID
FROM     Booking B
WHERE    B.Resource_ID=101
   AND   B.Booking_Date= '2025-11-20'
   AND    B.status = 'confirmed'
   AND('10:00:00' < B.End_Time AND '12:00:00' >
          B.start_Time);
```

B) SQL query to create a new booking

```
INSERT INTO Booking( Booking-Date, Start_Time,
End_Time, status, User-ID, Resource_ID)
VALUES (' 2025-11-20', '10:00:00', '12:00:00',
'confirmed', 50, 101);
```

C) SQL query to cancel a Booking

```
UPDATE Booking
SET status= 'cancelled'
WHERE Booking_ID= 500
      AND User_ID= 50,
```

Cancel (or an admin)

4 Describe how transaction recovery answers data integrity when multiple users book simultaneously.

A • Transaction recovery user mechanisms like write A head Logging (WAL), checkpointing and undo/redo operations.

• Transaction recovery is crucial for maintaining atomicity, consistency, isolation and durability (ACID) properties, especially during simultaneous bookings.

1 Atomicity: The booking process must be treated as a single, indivisible unit. If any part fails the entire transaction is rolled back, and the system state remains as it was before the attempt.

2 Isolation: This is key for simultaneous

bookings. The database uses locking protocols or multi version concurrency control (MVCC) to prevent transactions from interfering with each other.

(3) Durability (Persistence): Once a transaction is successfully \text{COMMITTED}$ (the booking is confirmed), the changes are permanently stored on non-volatile storage

(4) Transaction Recovery:

Transaction Recovery ensures that after a system crash, the database is restored to a consistent state.

(5) Implement CRUD operations in a NO SQL database for event management

A
- Create: db.event.insert one({name: "Tech_Fest", date: "2025-11-20", location: "Auditorium"})
- Read: db.events.find({location: "Auditorium"})
- Update: db.events.update one({name: "Tech_Fest"}, {set: {location: "Auditorium"}})
- Delete: db.events.delete one({name: "Tech_Fest"})