Name – Harshali deore

Branch – MCA

Email – harshalideore2002@gmail.com

Concurrency Handling

Scenario: You are developing a Node.js application where users can book events. The event booking system should prevent double bookings for the same event.

Requirements:

- Describe how you would handle concurrent booking requests to ensure that an event cannot be booked more than once at the same time.
- Suggest a strategy for implementing this in both the application code and the database schema.

---

1) Pessimistic Locking (Database Level) - Obtain a lock on the event record before making any updates, according to the pessimistic locking mechanism (database level). No other transaction may alter the record while it is locked.
   Execution:
   Schema for Databases: To the event database, add a boolean field called is_booked.
   Query: To obtain a lock on the event, use a transaction containing a SELECT... FOR UPDATE statement. Update the is_booked column to true and commit the transaction if it is currently false. If not, terminate the transaction.

   Optimistic Locking (Application Level) - The mechanism of Optimistic Locking (Application Level) involves storing a timestamp or version number alongside the event record. Verify that the version number in the database matches the one being used when changing the record. Proceed with the update if they correspond. If not, carry out the procedure again.
   Execution:
   Schema for Databases: Incorporate an integer version column into the event table.
   Application Code: Get the version number of the event record.
   Both the version number and the is_booked column should be updated.
   Make an effort to update the database record.
   The reservation is verified if the update is successful. Try the procedure again if a version mismatch is the reason it fails.

2) **. Database Schema Strategy:**

- **Unique Constraints**: To ensure that no two bookings for the same user and event occur at the same time, enforce a **unique constraint** on the userId and eventId columns in the Bookings table.

```
CREATE TABLE Bookings (
    booking_id SERIAL PRIMARY KEY,
    user_id INT NOT NULL,
    event_id INT NOT NULL,
    booking_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(user_id, event_id),
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (event_id) REFERENCES Events(event_id)
);
```

This will ensure that each user can book an event only once, and if there's a concurrent booking attempt, the database will raise a unique constraint violation.