Google Forms

Благодарим за заполнение формы Введение в СУБД. ДЗ-10!

Полученные ответы

Изменить ответ

Введение в СУБД. ДЗ-10

Транзакции

Электронная почта *

mirzo.zaynidinov@gmail.com

Студент *

Зайнидинов Мирзофирдавс Шавкатович

1. Уровни изоляции

Для всех заданий этого пункта в единственной строке кода должно быть приведено объявление начала транзакции. Предшествующие строки должны содержать комментарии (--), обосновывающие выбор уровня изоляции транзакции.

1.1. Администрирование

1.1.1 RegisterUser *

-- Избежание грязного чтения: Уровень Read Committed предотвращает грязные чтения, что означает, что транзакция не может видеть изменения, сделанные другими незавершенными транзакциями. Это важно для регистрации пользователей, чтобы избежать ситуации, когда два пользователя могут одновременно попытаться зарегистрироваться с одинаковым UserId. -- Производительность: Read Committed обеспечивает хорошую производительность по сравнению с более строгими уровнями изоляции, такими как Repeatable Read или Serializable. Это особенно важно в системах с высокой нагрузкой, где большое количество пользователей может одновременно пытаться зарегистрироваться. -- Согласованность данных: При использовании Read Committed гарантируется, что данные будут согласованы на момент выполнения запроса. Это позволяет избежать ситуации, когда один пользователь видит изменения другого пользователя до завершения его транзакции. START TRANSACTION READ WRITE ISOLATION LEVEL READ COMMITTED;

1.1.2. ManageFlight *

-- Избежание грязных чтений: Уровень изоляции Read Committed предотвращает грязные чтения, что означает, что транзакция не может видеть изменения, сделанные другими незавершенными транзакциями. Это важно для функции ManageFlight, чтобы гарантировать, что пользователь видит актуальные данные о рейсе и его настройках. -- Производительность: Read Committed обеспечивает хорошую производительность, так как он не требует блокировки строк на длительное время, как более строгие уровни изоляции (например, Serializable). Это особенно важно в системах с высокой нагрузкой, где пользователи могут одновременно пытаться изменять настройки рейсов. -- Согласованность данных: Этот уровень изоляции гарантирует, что изменения, сделанные другими транзакциями, не будут видны до тех пор, пока они не будут зафиксированы. Это важно для предотвращения конфликтов при одновременном обновлении настроек рейса. START TRANSACTION READ WRITE ISOLATION LEVEL READ COMMITTED;

1.2. Покупка и бронирование

1.2.1. FreeSeats *

-- Избежание грязных чтений: Уровень изоляции Read Committed предотвращает грязные чтения, что означает, что транзакция не может видеть незавершенные изменения других транзакций. Это критично для функции FreeSeats, так как она зависит от актуальных данных о рейсах и заказах. Если бы уровень изоляции был более низким, например Read Uncommitted, это могло бы привести к некорректным результатам при выборе доступных мест. --

Согласованность данных: Использование Read Committed гарантирует, что данные, которые функция читает, являются завершенными и согласованными на момент выполнения запроса. Это особенно важно при проверке условий для бронирования и покупки мест. -- Производительность: Уровень Read Committed обеспечивает хорошую производительность, так как он не требует блокировки строк на длительное время, в отличие от более строгих уровней изоляции, таких как Serializable. Это позволяет функции FreeSeats работать эффективно даже при высокой нагрузке. -- Неповторяемое чтение: Не может произойти из-за чтения каждой таблицы 1 раз -- Чтение фантомов: Не может произойти из-за чтения каждой таблицы 1 раз START TRANSACTION READ ONLY ISOLATION LEVEL READ COMMITTED;

1.2.2. Reserve *

-- Режим: Read Write так как происходит чтение и запись -- Фантомная запись: Может произайти если параллельно запустить Reserve два раза, обе транзакции проверят что место и свободно и попытаются его взять -- Косая запись: Не страшна так как обновляем одну запись => уровень изоляции snapshot START TRANSACTION READ WRITE ISOLATION LEVEL SNAPSHOT;

1.2.3. ExtendReservation *

-- Режим: Read Write так как происходит чтение и запись -- Фантомная запись: не может произайти так как работаем с одной существующей записью и repeatable read гарантирует что она не поменяется -- Косая запись: Не странша так как обновляем только одну запись. -- Неповторяемое чтение: Может произайти при одновременном бронировании или покупки места START TRANSACTION READ WRITE ISOLATION LEVEL REPEATABLE READ;

1.2.4. BuyFree *

-- Режим: Read Write так как происходит чтение и запись -- Фантомная запись: Может произайти если параллельно запустить BuyFree два раза, обе транзакции проверят что место и свободно и попытаются его взять -- Касая запись: Не страшна так как обноволяем одну запись => уровень изоляции snapshot START TRANSACTION READ WRITE ISOLATION LEVEL SNAPSHOT;

1.2.5. BuyReserved *

-- Режим: Read Write так как происходит чтение и запись -- Неповторяемое чтение: Может быть при одновременном продлении бронирования и покупки места -- Фантомная запись: не может

произайти так как работаем с одной существующей записью и repeatable read гарантирует что она не поменяется -- Косая запись: Не странша так как обновляем только одну запись. START TRANSACTION READ WRITE ISOLATION LEVEL REPEATABLE READ;

1.3. Статистика

1.3.1. FlightsStatistics *

-- Режим: Read Only так как происходит только чтение -- Грязное чтение: Может быть так как требуются корректные данные при проверки авторизации => уровень изоляции read commited так как предотвращает грязное чтение. -- Все остальные аномалии: Не проблема так как читаем только один раз START TRANSACTION READ ONLY ISOLATION LEVEL READ COMMITTED;

1.3.2. FlightStat *

-- Режим: Read Only так как происходит только чтение -- Грязное чтение: Может быть так как требуются корректные данные при проверки авторизации => уровень изоляции read commited так как предотвращает грязное чтение. -- Все остальные аномалии: Не проблема так как читаем только один раз START TRANSACTION READ ONLY ISOLATION LEVEL READ COMMITTED;

1.4. CompressSeats *

-- Режим: Read Write так как происходит чтение и запись -- Фантомная запись: проблема так как помешает корректно пересортировать места => уровень изоляции snapshot -- Косая запись: не может быть так как все происходит ппо очереди START TRANSACTION READ WRITE ISOLATION LEVEL SNAPSHOT;

2. Сценарий

2.0. Общий план *

Изложите общий план реализации, в том числе на каком этапе какие данные будут получены и как использованы

Сервис обращается к бд с запросом о свободных местах. Пользователь из предоставленных мест выбирает то место которое он хочет Сервис обрабатывает запрос от пользователя и выполняет соответствующую функцию с необходимым уровнем изоляции

2.1. Запрос списка свободных мест *

Приведите соответствующие SQL-запросы

START TRANSACTION READ ONLY ISOLATION LEVEL READ COMMITTED; SELECT SeatNo FROM FreeSeats(:FlightId); COMMIT;

2.2. Взаимодействие с пользователем *

Укажите как будет релизовано взаимодействие с пользователем

Список рейсов будет представлен пользователю в виде выпадающего меню, также будет доступна функция поиска. После того как пользователь выберет рейс из списка, по его идентификатору будет выполнен запрос к сервису, и отобразится список свободных мест. При выборе места пользователю будет предложено либо купить его (без необходимости авторизации), либо забронировать (для этого потребуется авторизация).

2.3. Действия с местом *

Приведите соответствующие SQL-запросы

START TRANSACTION READ WRITE ISOLATION LEVEL SNAPSHOT; SELECT Reserve(:UserId, :Pass, :FlightId, :SeatNo); COMMIT; START TRANSACTION READ WRITE ISOLATION LEVEL SNAPSHOT; SELECT BuyFree(:FlightId, :SeatNo); COMMIT;