Лабораторная работа. Транзакции.

Цель работы

Используя данные базы данных, подготовленной в предыдущей лабораторной работе, подготовить и реализовать серию запросов, связанных с выборкой информации и модификацией данных таблиц.

Задание 1

Используя базу, полученную в лабораторной 2, создать транзакцию, произвести ее откат и фиксацию. Показать, что данные существовали до отката, удалились после отката, снова были добавлены, и затем были успешно зафиксированы. При необходимости используйте точки сохранения и вложенные транзакции.

```
-- записей нет
SELECT * FROM Deal WHERE ClientID = 10 AND DaysCount = 7;

-- создание
BEGIN TRANSACTION;

INSERT INTO Deal (
    [Date],
    DaysCount,
    TotalPrice,
    ClientID,
    DiscountID
)
VALUES (GETDATE(), 7, 13650.00, 10, 2);

DECLARE @DealID INT = SCOPE_IDENTITY();

SELECT * FROM Deal WHERE ID = @DealID;

-- назначение автомобиля
INSERT INTO Deal_Car (DealID, CarID, ReturnID)
VALUES (@DealID, 3, NULL);

SELECT * FROM Deal_Car WHERE DealID = @DealID;

-- создание возврата
INSERT INTO [Return] (Fine, [Date])
VALUES (0, DATEADD(day, 7, GETDATE()));

DECLARE @ReturnID INT = SCOPE_IDENTITY();

-- обновление связи с возвратом
UPDATE Deal_Car
SET ReturnID = @ReturnID
WHERE DealID = @DealID AND CarID = 3;

SELECT * FROM Deal_Car WHERE DealID = @DealID;

-- сейвпоинт и добавление скидки
SAVE TRANSACTION BeforeDiscount;
UPDATE Deal
SET DiscountID = 3,
    TotalPrice = TotalPrice * 0.9
WHERE ID = @DealID;

SELECT * FROM Deal WHERE ID = @DealID;

-- частничный откат
ROLLBACK TRANSACTION BeforeDiscount;

-- проверка после частичного отката
SELECT * FROM Deal WHERE ID = @DealID;
SELECT * FROM Deal_Car WHERE DealID = @DealID;
SELECT * FROM [Return] WHERE ID = @ReturnID;

-- полный откат
ROLLBACK TRANSACTION;

-- проверка после полного отката
SELECT * FROM Deal WHERE ClientID = 10 AND DaysCount = 7;

-- повторное выполнение и фиксация
```

```sql
BEGIN TRANSACTION;

INSERT INTO Deal (
    [Date],
    DaysCount,
    TotalPrice,
    ClientID,
    DiscountID
)
VALUES (GETDATE(), 7, 13650.00, 10, 2);

DECLARE @DealID2 INT = SCOPE_IDENTITY();

INSERT INTO Deal_Car (DealID, CarID, ReturnID)
VALUES (@DealID2, 3, NULL);

INSERT INTO [Return] (Fine, [Date])
VALUES (0, DATEADD(day, 7, GETDATE()));

DECLARE @ReturnID2 INT = SCOPE_IDENTITY();

UPDATE Deal_Car
SET ReturnID = @ReturnID2
WHERE DealID = @DealID2 AND CarID = 3;

COMMIT TRANSACTION;

-- проверка после коммита
SELECT * FROM Deal WHERE ID = @DealID2;
SELECT * FROM Deal_Car WHERE DealID = @DealID2;
SELECT * FROM [Return] WHERE ID = @ReturnID2;
```

	ID	Date	DaysCount	TotalPrice	ClientID	DiscountID
1	16	2026-01-13	7	13650,00	10	2

	DealID	CarID	ReturnID
1	16	3	NULL

	DealID	CarID	ReturnID
1	16	3	16

	ID	Date	DaysCount	TotalPrice	ClientID	DiscountID
1	16	2026-01-13	7	12285,00	10	3

	ID	Date	DaysCount	TotalPrice	ClientID	DiscountID
1	16	2026-01-13	7	13650,00	10	2

	DealID	CarID	ReturnID
1	16	3	16

	ID	Fine	Date
1	16	0,00	2026-01-20

	ID	Date	DaysCount	TotalPrice	ClientID	DiscountID
1	17	2026-01-13	7	13650,00	10	2

	DealID	CarID	ReturnID
1	17	3	17

	ID	Fine	Date
1	17	0,00	2026-01-20

Задание 2

Подготовить SQL-скрипты для выполнения проверок изолированности транзакций. Ваши скрипты должны работать с одной из таблиц, созданных в лабораторной работе №2.

Выполнение работы

1.Запустить клиента и соединиться с базой данных. Открыть второе окно для ввода текста запросов (Ctrl+N в первом окне).

2. Установить в обоих сеансах уровень изоляции READ UNCOMMITTED. Выполнить сценарии проверки:

-потерянных изменений

Окно 1:

```sql
-- изоляция
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
BEGIN TRANSACTION;

SELECT * FROM Deal WHERE ID = 1;

-- Изменение данных
UPDATE Deal
SET TotalPrice = TotalPrice * 1.1
WHERE ID = 1;

-- Пауза для выполнения сессии 2
WAITFOR DELAY '00:00:10';

-- Завершение транзакции
COMMIT;

-- Проверка результата
SELECT * FROM Deal WHERE ID = 1;
```

	ID	Date	DaysCount	TotalPrice	ClientID	DiscountID
1	1	2025-02-01	5	12350,00	1	1

	ID	Date	DaysCount	TotalPrice	ClientID	DiscountID
1	1	2025-02-01	5	13585,00	1	1

Окно 2:

```sql
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

-- Пауза для начала выполнения сессии 1
```

```
WAITFOR DELAY '00:00:05';

BEGIN TRANSACTION;
SELECT * FROM Deal WHERE ID = 1;

-- Попытка изменения тех же данных
UPDATE Deal
SET DaysCount = DaysCount + 1
WHERE ID = 1;

-- Завершение транзакции
COMMIT;

-- Проверка результата
SELECT * FROM Deal WHERE ID = 1;
```

Результаты | Сообщения

	ID	Date	DaysCount	TotalPrice	ClientID	DiscountID
1	1	2025-02-01	5	13585,00	1	1

	ID	Date	DaysCount	TotalPrice	ClientID	DiscountID
1	1	2025-02-01	6	13585,00	1	1

-грязного чтения

Окно 1:

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

BEGIN TRANSACTION;
-- Изменение данных без коммита
UPDATE Deal
SET TotalPrice = 15000.00
WHERE ID = 2;

WAITFOR DELAY '00:00:10';
-- Откат изменений
ROLLBACK;
-- Проверка, что данные вернулись к исходным
SELECT * FROM Deal WHERE ID = 2;
```

	ID	Date	DaysCount	TotalPrice	ClientID	DiscountID
1	2	2025-02-02	3	5265,00	1	2

Окно 2:

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
-- Пауза для начала выполнения сессии 1
WAITFOR DELAY '00:00:05';

BEGIN TRANSACTION;
-- грязное чтение
SELECT * FROM Deal WHERE ID = 2;
-- Дополнительная проверка
SELECT 'Прочитано грязное значение: ' + CAST(TotalPrice AS VARCHAR)
FROM Deal
WHERE ID = 2;
-- Пауза для отката в сессии 1
WAITFOR DELAY '00:00:10';
-- Проверка данных после отката
SELECT * FROM Deal WHERE ID = 2;

-- Завершение транзакции
COMMIT;
```

	ID	Date	DaysCount	TotalPrice	ClientID	DiscountID
1	2	2025-02-02	3	5265,00	1	2

	(Отсутствует имя столбца)
1	Прочитано грязное значение: 5265.00

	ID	Date	DaysCount	TotalPrice	ClientID	DiscountID
1	2	2025-02-02	3	5265,00	1	2

READ UNCOMMITTED позволяет транзакциям читать незафиксированные данные других транзакций "грязные данные". В эксперименте вторая транзакция увидела значения, которые первая потом отменила через ROLLBACK.

Итог: Это самый быстрый уровень изоляции, но он полностью жертвует согласованностью данных. Подходит только для операций, где не нужна точность — например, для аналитики или примерных отчётов.

3.Записать протокол выполнения сценариев.

4. Установить в обоих сеансах уровень изоляции READ COMMITTED. Выполнить сценарии проверки:

-грязного чтения

Окно 1:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
GO

BEGIN TRANSACTION;

-- Изменение данных без коммита
UPDATE Deal
SET TotalPrice = 16000.00
WHERE ID = 3;
-- Проверка своих изменений
SELECT 'Сессия 1: Мои изменения (в транзакции) - ' + CAST(TotalPrice AS VARCHAR)
FROM Deal
WHERE ID = 3;
-- Пауза для выполнения окна 2
PRINT 'Окно 1: Ожидание 15 секунд...';
WAITFOR DELAY '00:00:15';
-- Откат изменений
ROLLBACK;
-- Проверка после отката
SELECT 'Окно 1: После отката - ' + CAST(TotalPrice AS VARCHAR)
FROM Deal
WHERE ID = 3;
```

	(Отсутствует имя столбца)
1	Сессия 1: Мои изменения (в транзакции) - 16000.00

	(Отсутствует имя столбца)
1	Окно 1: После отката - 7400.00

Окно 2:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
GO
-- Пауза для начала выполнения окна 1
WAITFOR DELAY '00:00:05';

BEGIN TRANSACTION;
-- Попытка чтения данных
SELECT 'Сессия 2: Попытка чтения (READ COMMITTED) - ' + ISNULL(CAST(TotalPrice AS VARCHAR), 'NULL')
FROM Deal
WHERE ID = 3;

WAITFOR DELAY '00:00:15';
-- Проверка данных после завершения окна 1
SELECT 'Окно 2: После завершения сессии 1 - ' + CAST(TotalPrice AS VARCHAR)
FROM Deal
WHERE ID = 3;

COMMIT;
```

	(Отсутствует имя столбца)
1	Сессия 2: Попытка чтения (READ COMMITTED) - 7400...

	(Отсутствует имя столбца)
1	Окно 2: После завершения сессии 1 - 7400.00

-неповторяющегося чтения

Окно 1:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
GO
BEGIN TRANSACTION;
-- Первое чтение данных
SELECT 'Окно 1: Первое чтение - ID: ' + CAST(ID AS VARCHAR) +
       ', DaysCount: ' + CAST(DaysCount AS VARCHAR) +
       ', TotalPrice: ' + CAST(TotalPrice AS VARCHAR)
FROM Deal
WHERE ID = 4;
```

```
-- Пауза между чтениями
WAITFOR DELAY '00:00:20';
-- Второе чтение тех же данных
SELECT 'Окно 1: Второе чтение - ID: ' + CAST(ID AS VARCHAR) +
       ', DaysCount: ' + CAST(DaysCount AS VARCHAR) +
       ', TotalPrice: ' + CAST(TotalPrice AS VARCHAR)
FROM Deal
WHERE ID = 4;

COMMIT;
```

	(Отсутствует имя столбца)
1	Окно 1: Первое чтение - ID: 4, DaysCount: 7, TotalPrice: 18270.00

	(Отсутствует имя столбца)
1	Окно 1: Второе чтение - ID: 4, DaysCount: 7, TotalPrice: 18270.00

Окно 2:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
GO
-- Пауза для начала выполнения окна 1
WAITFOR DELAY '00:00:10';

BEGIN TRANSACTION;
-- Изменение данных между чтениями окна 1
UPDATE Deal
SET DaysCount = DaysCount + 2,
    TotalPrice = TotalPrice * 1.05
WHERE ID = 4;
-- Проверка своих изменений
SELECT 'Окно 2: После изменения - ID: ' + CAST(ID AS VARCHAR) +
       ', DaysCount: ' + CAST(DaysCount AS VARCHAR) +
       ', TotalPrice: ' + CAST(TotalPrice AS VARCHAR)
FROM Deal
WHERE ID = 4;

COMMIT;
```

	(Отсутствует имя столбца)
1	Окно 2: После изменения - ID: 4, DaysCount: 9, TotalPrice: 19183.50

READ COMMITTED предотвращает грязное чтение — транзакции видят только зафиксированные данные. Однако неповторяющееся чтение остаётся: если одна транзакция дважды читает одни данные, другая может изменить их между чтениями.

Итог: Это баланс между скоростью и надёжностью. Подходит для большинства операций, но не для расчётов, требующих неизменности данных.

5.Записать протокол выполнения сценариев.

6.Установить в обоих сеансах уровень изоляции REPEATABLE READ. Выполнить сценарии проверки:

-неповторяющегося чтения:

Окно 1:

```
BEGIN TRANSACTION;
-- Первое чтение данных
SELECT 'Окно 1: Первое чтение - ID: ' + CAST(ID AS VARCHAR) +
       ', ClientID: ' + CAST(ClientID AS VARCHAR) +
       ', TotalPrice: ' + CAST(TotalPrice AS VARCHAR)
FROM Deal
WHERE ID = 5;

WAITFOR DELAY '00:00:20';
-- Второе чтение тех же данных
SELECT 'Окно 1: Второе чтение - ID: ' + CAST(ID AS VARCHAR) +
       ', ClientID: ' + CAST(ClientID AS VARCHAR) +
       ', TotalPrice: ' + CAST(TotalPrice AS VARCHAR)
FROM Deal
WHERE ID = 5;
COMMIT;
```

	(Отсутствует имя столбца)
1	Окно 1: Первое чтение - ID: 5, ClientID: 3, TotalPrice: 7500.00

	(Отсутствует имя столбца)
1	Окно 1: Второе чтение - ID: 5, ClientID: 3, TotalPrice: 7500.00

Окно 2:

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
GO
WAITFOR DELAY '00:00:10';
BEGIN TRANSACTION;
-- Попытка изменения данных, которые читает окно 1
UPDATE Deal
SET TotalPrice = TotalPrice * 1.1,
    DiscountID = 2
WHERE ID = 5;
-- Проверка своих изменений
SELECT 'Окно 2: После изменения - ID: ' + CAST(ID AS VARCHAR) +
       ', ClientID: ' + CAST(ClientID AS VARCHAR) +
       ', TotalPrice: ' + CAST(TotalPrice AS VARCHAR)
FROM Deal
WHERE ID = 5;

COMMIT;
```

	(Отсутствует имя столбца)
1	Окно 2: После изменения - ID: 5, ClientID: 3, TotalPrice: 8250.00

-фантома

Окно 1:

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
GO
BEGIN TRANSACTION;
-- Первое чтение диапазона
SELECT 'Окно 1: Сделки с TotalPrice от 10000 до 15000:'
SELECT ID, [Date], DaysCount, TotalPrice, ClientID
FROM Deal
WHERE TotalPrice BETWEEN 10000 AND 15000
ORDER BY TotalPrice;

WAITFOR DELAY '00:00:15';
-- Второе чтение того же диапазона
SELECT 'Окно 1: Повторное чтение сделок с TotalPrice от 10000 до 15000:'
SELECT ID, [Date], DaysCount, TotalPrice, ClientID
FROM Deal
WHERE TotalPrice BETWEEN 10000 AND 15000
ORDER BY TotalPrice;

COMMIT;
```

	(Отсутствует имя столбца)
1	Окно 1: Сделки с TotalPrice от 10000 до 15000:

	ID	Date	DaysCount	TotalPrice	ClientID
1	11	2025-03-11	4	11160,00	6
2	8	2025-03-08	5	12000,00	4
3	12	2025-03-12	2	12200,00	6
4	1	2025-02-01	6	13585,00	1
5	17	2026-01-13	7	13650,00	10

	(Отсутствует имя столбца)
1	Окно 1: Повторное чтение сделок с TotalPrice от ...

	ID	Date	DaysCount	TotalPrice	ClientID
1	11	2025-03-11	4	11160,00	6
2	8	2025-03-08	5	12000,00	4
3	12	2025-03-12	2	12200,00	6
4	1	2025-02-01	6	13585,00	1
5	17	2026-01-13	7	13650,00	10

Окно 2:

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
GO
WAITFOR DELAY '00:00:08';

BEGIN TRANSACTION;
-- Вставка новой записи в диапазон
INSERT INTO Deal ([Date], DaysCount, TotalPrice, ClientID, DiscountID)
VALUES (GETDATE(), 5, 12000.00, 7, 1);
-- Проверка вставки
SELECT 'Окно 2: Проверка вставки - ID: ' + CAST(ID AS VARCHAR) +
       ', TotalPrice: ' + CAST(TotalPrice AS VARCHAR)
FROM Deal
WHERE TotalPrice = 12000.00 AND ClientID = 7;

COMMIT;
```

	(Отсутствует имя столбца)
1	Окно 2: Проверка вставки - ID: 18, TotalPrice: 12000.00

REPEATABLE READ защищает от неповторяющегося чтения — прочитанные строки блокируются до конца транзакции. Но фантомные записи возможны: другие транзакции могут добавлять новые строки в диапазон выборки.

Итог: Хорош для операций, где важна стабильность существующих данных, но не гарантирует полной изоляции.

7.Записать протокол выполнения сценариев.

8.Установить в обоих сеансах уровень изоляции SERIALIZABLE. Выполнить сценария проверки

-фантома

Окно 1:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN TRANSACTION;
-- Чтение диапазона
SELECT 'Окно 1: Первое чтение'
SELECT ID, TotalPrice, ClientID FROM Deal WHERE TotalPrice BETWEEN 5000 AND 10000;
WAITFOR DELAY '00:00:10';
-- Повторное чтение
SELECT 'Окно 1: Второе чтение'
SELECT ID, TotalPrice, ClientID FROM Deal WHERE TotalPrice BETWEEN 5000 AND 10000;
COMMIT;
```

	(Отсутствует имя столбца)
1	Окно 1: Первое чтение

	ID	TotalPrice	ClientID
1	2	5265,00	1
2	3	7400,00	2
3	5	8250,00	3
4	6	8400,00	3
5	10	5250,00	5

	(Отсутствует имя столбца)
1	Окно 1: Второе чтение

	ID	TotalPrice	ClientID
1	2	5265,00	1
2	3	7400,00	2
3	5	8250,00	3
4	6	8400,00	3
5	10	5250,00	5

Окно 2:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
WAITFOR DELAY '00:00:05';
BEGIN TRANSACTION;
INSERT INTO Deal ([Date], DaysCount, TotalPrice, ClientID, DiscountID)
VALUES (GETDATE(), 3, 7500.00, 8, NULL);
COMMIT;
```

```
(Затронута 1 строка)
```

SERIALIZABLE полностью устраняет все аномалии параллельного доступа: грязные, неповторяющиеся и фантомные чтения. При этом уровне транзакции выполняются так, будто работают последовательно, а не одновременно. Достигается это за счёт жёстких блокировок — например, если одна транзакция читает определённый диапазон значений, другая не может добавить в него новые записи до завершения первой.

Итог: SERIALIZABLE гарантирует максимальную целостность данных, но серьёзно снижает производительность и параллелизм. Применяется в критических операциях (финансовые расчёты, бронирование), где ошибки недопустимы.

9.Записать протокол выполнения сценария.

Содержание отчета:

-Сценарий и протокол его выполнения.

-Краткие выводы о навыках, приобретенных в ходе выполнения работы.