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.

1 задание

```
1  USE JaweleryStore;
2
3  --проверка исходных данных
4  PRINT ' ИСХОДНЫЕ ДАННЫЕ до транзакции';
5  SELECT ID, data, status_, price, client_ID
6  FROM Contract
7  WHERE client_ID = 1
8  ORDER BY ID;
9
```

90 % Проблемы не найдены.

Результаты Сообщения

	ID	data	status_	price	client_ID
1	1	2025-01-01	Выполнен	45000,00	1
2	11	2025-11-14	Выполнен	89200,00	1

```sql
PRINT 'первая транзакция для отката';
BEGIN TRANSACTION Trans1;
GO

PRINT 'Проверяем максимальный ID перед добавлением:';
SELECT MAX(ID) as MaxID FROM Contract;
GO

PRINT 'Добавляем новый контракт в транзакции:';

INSERT INTO Contract (data, status_, price, client_ID, jewelery_ID, discount_ID)
SELECT
    '2025-12-01',
    'Ожидание',
    50000,
    1,
    1,
    1
WHERE EXISTS (SELECT 1 FROM Client WHERE ID = 1)
   AND EXISTS (SELECT 1 FROM Jewelery WHERE ID = 1)
   AND EXISTS (SELECT 1 FROM Discount WHERE ID = 1);
GO

PRINT 'Данные после добавления (в транзакции):';
SELECT ID, data, status_, price, client_ID
FROM Contract
WHERE client_ID = 1
ORDER BY ID;
GO
```

90 % Проблемы не найдены. Стр: 24 Симв: 8

Результаты Сообщения

	MaxID
1	13

	ID	data	status_	price	client_ID
1	1	2025-01-01	Выполнен	45000,00	1
2	11	2025-11-14	Выполнен	89200,00	1
3	17	2025-12-01	Ожидание	50000,00	1

```sql
PRINT 'выполняем откат транзакции';
ROLLBACK TRANSACTION Trans1;

PRINT 'Данные после отката:';
SELECT ID, data, status_, price, client_ID
FROM Contract
WHERE client_ID = 1
ORDER BY ID;
```

90 % Проблемы не найдены.

Результаты Сообщения

	ID	data	status_	price	client_ID
1	1	2025-01-01	Выполнен	45000,00	1
2	11	2025-11-14	Выполнен	89200,00	1

```
PRINT 'начинаем вторую транзакцию для фиксации';
BEGIN TRANSACTION Trans2;
GO

PRINT 'Проверяем данные перед изменениями:';
SELECT ID, data, status_, price, client_ID
FROM Contract
WHERE client_ID = 1
ORDER BY ID;
```

90 % Проблемы не найдены.

Результаты Сообщения

	ID	data	status_	price	client_ID
1	1	2025-01-01	Выполнен	45000,00	1
2	11	2025-11-14	Выполнен	89200,00	1

```
PRINT 'снова добавляем контракт (теперь для фиксации):';
INSERT INTO Contract (data, status_, price, client_ID, jewelery_ID, discount_ID)
VALUES ('2025-12-01', 'Ожидание', 50000, 1, 1, 1);
GO

PRINT 'изменяем существующий контракт:';
UPDATE Contract
SET
    status_ = 'В обработке',
    price = price + 5000  -- Простое увеличение вместо умножения
WHERE ID = 1 AND client_ID = 1;
GO

PRINT 'данные перед фиксацией:';
SELECT ID, data, status_, price, client_ID
FROM Contract
WHERE client_ID = 1
ORDER BY ID;
```

90 % Проблемы не найдены. Стр: 15 Симв: 44

Результаты Сообщения

	ID	data	status_	price	client_ID
1	1	2025-01-01	В обработке	50000,00	1
2	11	2025-11-14	Выполнен	89200,00	1
3	18	2025-12-01	Ожидание	50000,00	1

```
1  PRINT 'фиксируем транзакцию';
2  COMMIT TRANSACTION Trans2;
3  GO
4
5  PRINT 'данные после фиксации:';
6  SELECT ID, data, status_, price, client_ID
7  FROM Contract
8  WHERE client_ID = 1
9  ORDER BY ID;
```

90 % Проблемы не найдены.

Результаты Сообщения

	ID	data	status_	price	client_ID
1	1	2025-01-01	В обработке	50000,00	1
2	11	2025-11-14	Выполнен	89200,00	1
3	18	2025-12-01	Ожидание	50000,00	1

```
PRINT 'демонстрация точки сохранения';
BEGIN TRANSACTION Trans3;
GO

PRINT 'исходные данные контракта ID=2:';
SELECT ID, data, status_, price, client_ID
FROM Contract
WHERE ID = 2;
GO

-- Создаем точку сохранения
SAVE TRANSACTION SavePoint1;

PRINT 'изменяем контракт ID=2:';
UPDATE Contract
SET
    status_ = 'Отменен',
    price = 70000
WHERE ID = 2;
GO

PRINT 'данные после изменения:';
SELECT ID, data, status_, price, client_ID
FROM Contract
WHERE ID = 2;
GO

PRINT 'откатываемся к точке сохранения:';
ROLLBACK TRANSACTION SavePoint1;
GO

PRINT 'данные после отката к точке сохранения:';
SELECT ID, data, status_, price, client_ID
FROM Contract
WHERE ID = 2;
GO

PRINT 'делаем другие изменения:';
UPDATE Contract
SET
    status_ = 'Пересмотр',
    price = price + 1000
WHERE ID = 2;
GO

PRINT 'данные перед фиксацией:';
```

```
SET
    status_ = 'Пересмотр',
    price = price + 1000
WHERE ID = 2;
GO

PRINT 'данные перед фиксацией:';
SELECT ID, data, status_, price, client_ID
FROM Contract
WHERE ID = 2;
GO

PRINT 'фиксируем транзакцию с точкой сохранения';
COMMIT TRANSACTION Trans3;
GO

PRINT 'финальные данные контракта ID=2:';
SELECT ID, data, status_, price, client_ID
FROM Contract
WHERE ID = 2;
GO
```

	ID	data	status_	price	client_ID
1	2	2025-02-04	В работе	78000,00	2

	ID	data	status_	price	client_ID
1	2	2025-02-04	Отменен	70000,00	2

	ID	data	status_	price	client_ID
1	2	2025-02-04	В работе	78000,00	2

	ID	data	status_	price	client_ID
1	2	2025-02-04	Пересмотр	79000,00	2

	ID	data	status_	price	client_ID
1	2	2025-02-04	Пересмотр	79000,00	2

```
1  PRINT 'итоговая проверка всех договоров'
2  SELECT ID, data, status_, price, client_ID
3  FROM Contract
4  ORDER BY ID;
5
```

90 % Проблемы не найдены.

Результаты Сообщения

	ID	data	status_	price	client_ID
1	1	2025-01-01	В обработке	50000,00	1
2	2	2025-02-04	Пересмотр	79000,00	2
3	3	2025-03-05	Выполнен	125000,00	3
4	4	2025-04-07	Отменен	32000,00	4
5	5	2025-05-08	Выполнен	95600,00	5
6	6	2025-06-09	В работе	45000,00	6
7	8	2025-08-11	Ожидание	189000,00	8
8	9	2025-09-12	Выполнен	54300,00	9
9	10	2025-10-13	В работе	36700,00	10
10	11	2025-11-14	Выполнен	89200,00	1
11	12	2025-12-12	Ожидание	67400,00	3
12	13	2025-12-12	Выполнен	50000,00	11
13	18	2025-12-01	Ожидание	50000,00	1

2 задание

Уровень изоляции: READ UNCOMMITTED

```sql
 --устанавливаем уровень изоляции
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
GO

--начинаем транзакцию
BEGIN TRANSACTION DirtyRead_T1;

-- изменяем данные (но не фиксируем)
PRINT 'Меняем статус контракта ID=1 на "Грязный статус"';
UPDATE Contract
SET status_ = 'Грязный статус'
WHERE ID = 1;

-- показываем незафиксированные данные
PRINT 'Показываем измененные данные (еще не зафиксированы):';
SELECT ID, status_, price FROM Contract WHERE ID = 1;

--ждем 10 секунд, чтобы окно 2 успело прочитать
PRINT 'Ждем 10 секунд...';
WAITFOR DELAY '00:00:10';

--откатываем изменения
PRINT 'Откатываем изменения (ROLLBACK)';
ROLLBACK TRANSACTION;

--показываем финальное состояние
PRINT 'Финальное состояние после отката:';
SELECT ID, status_, price FROM Contract WHERE ID = 1;
GO
```

90 % Проблемы не найдены. Стр: 29 Симв: 3 Та

Результаты Сообщения

	ID	status_	price
1	1	Грязный статус	50000,00

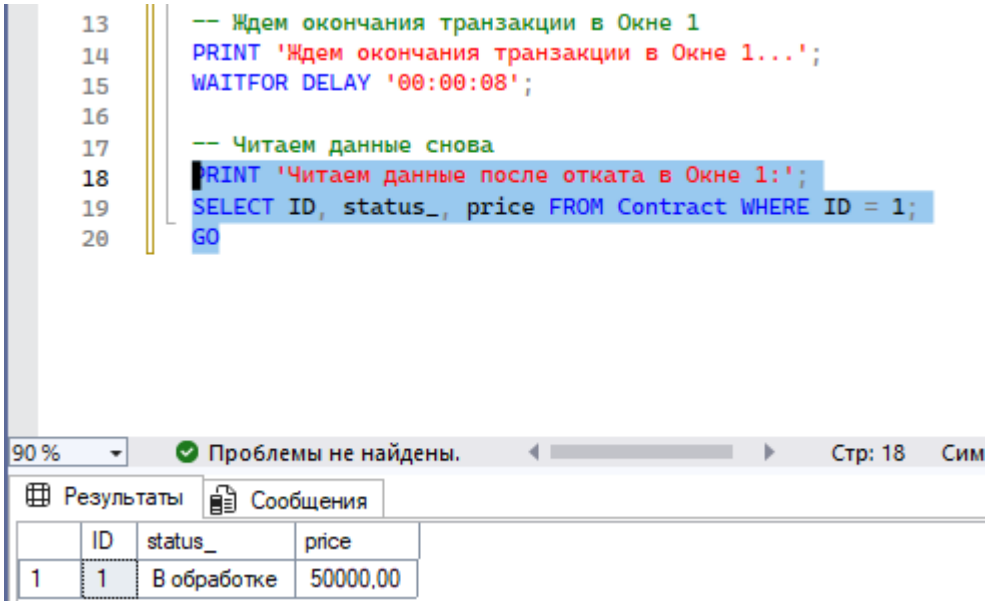
```sql
-- Устанавливаем уровень изоляции
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
GO

-- Ждем 3 секунды
PRINT 'Ждем 3 секунды, чтобы Окно 1 начало транзакцию...';
WAITFOR DELAY '00:00:03';

-- Читаем данные (грязное чтение!)
PRINT 'Читаем данные (ГРЯЗНОЕ ЧТЕНИЕ):';
SELECT ID, status_, price FROM Contract WHERE ID = 1;

-- Ждем окончания транзакции в Окне 1
PRINT 'Ждем окончания транзакции в Окне 1...';
WAITFOR DELAY '00:00:08';

-- Читаем данные снова
PRINT 'Читаем данные после отката в Окне 1:';
SELECT ID, status_, price FROM Contract WHERE ID = 1;
GO
```

90 % Проблемы не найдены. Стр: 1 Симв

Результаты Сообщения

	ID	status_	price
1	1	Грязный статус	50000,00

	ID	status_	price
1	1	Грязный статус	50000,00



Что выполнялось:

Окно 1: status_ = 'Грязный статус' (незафиксировано)

Окно 2: status_ = 'Грязный статус' ← грязное чтение

Окно 1: ROLLBACK

Окно 2: status_ = 'Выполнен' (исходное значение)

```sql
--устанавливаем уровень изоляции
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
GO

--начинаем транзакцию
BEGIN TRANSACTION LostUpdate_T1;

--читаем текущую цену контракта ID=2
PRINT 'Читаем цену контракта ID=2:';
SELECT ID, status_, price FROM Contract WHERE ID = 2;

-- ждем 5 секунд
PRINT 'Ждем 5 секунд...';
WAITFOR DELAY '00:00:05';

--увеличиваем цену на 5000
PRINT 'Увеличиваем цену на 5000...';
UPDATE Contract
SET price = price + 5000
WHERE ID = 2;

--фиксируем изменения
PRINT 'Фиксируем изменения (COMMIT)';
COMMIT TRANSACTION;

-- показываем результат
PRINT 'Результат после COMMIT:';
SELECT ID, status_, price FROM Contract WHERE ID = 2;
GO
```

90 % Проблемы не найдены. Стр: 14 Симв: 2

Результаты Сообщения

	ID	status_	price
1	2	Пересмотр	79000,00

```sql
-- устанавливаем уровень изоляции
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
GO

--ждем 1 секунду
PRINT 'Ждем 1 секунду...';
WAITFOR DELAY '00:00:01';

-- начинаем транзакцию
BEGIN TRANSACTION LostUpdate_T2;

-- читаем ту же цену
PRINT 'Читаем цену контракта ID=2:';
SELECT ID, status_, price FROM Contract WHERE ID = 2;

-- увеличиваем цену на 3000
PRINT 'Увеличиваем цену на 3000...';
UPDATE Contract
SET price = price + 3000
WHERE ID = 2;

--фиксируем изменения
PRINT 'Фиксируем изменения (COMMIT)';
COMMIT TRANSACTION;

--показываем результат
PRINT 'Результат после COMMIT:';
SELECT ID, status_, price FROM Contract WHERE ID = 2;
GO
```

90 % Проблемы не найдены. Стр: 1

Результаты Сообщения

	ID	status_	price
1	2	Пересмотр	79000,00

	ID	status_	price
1	2	Пересмотр	82000,00

Что выполнялось:

Оба окна читают: price = 79000

Окно 1: 79000 + 5000 = 84000

Окно 2: 79000 + 3000 = 82000 ← перезаписывает изменения Окна 1

Итог: price = 82000 вместо 84000 ← потеряные изменения

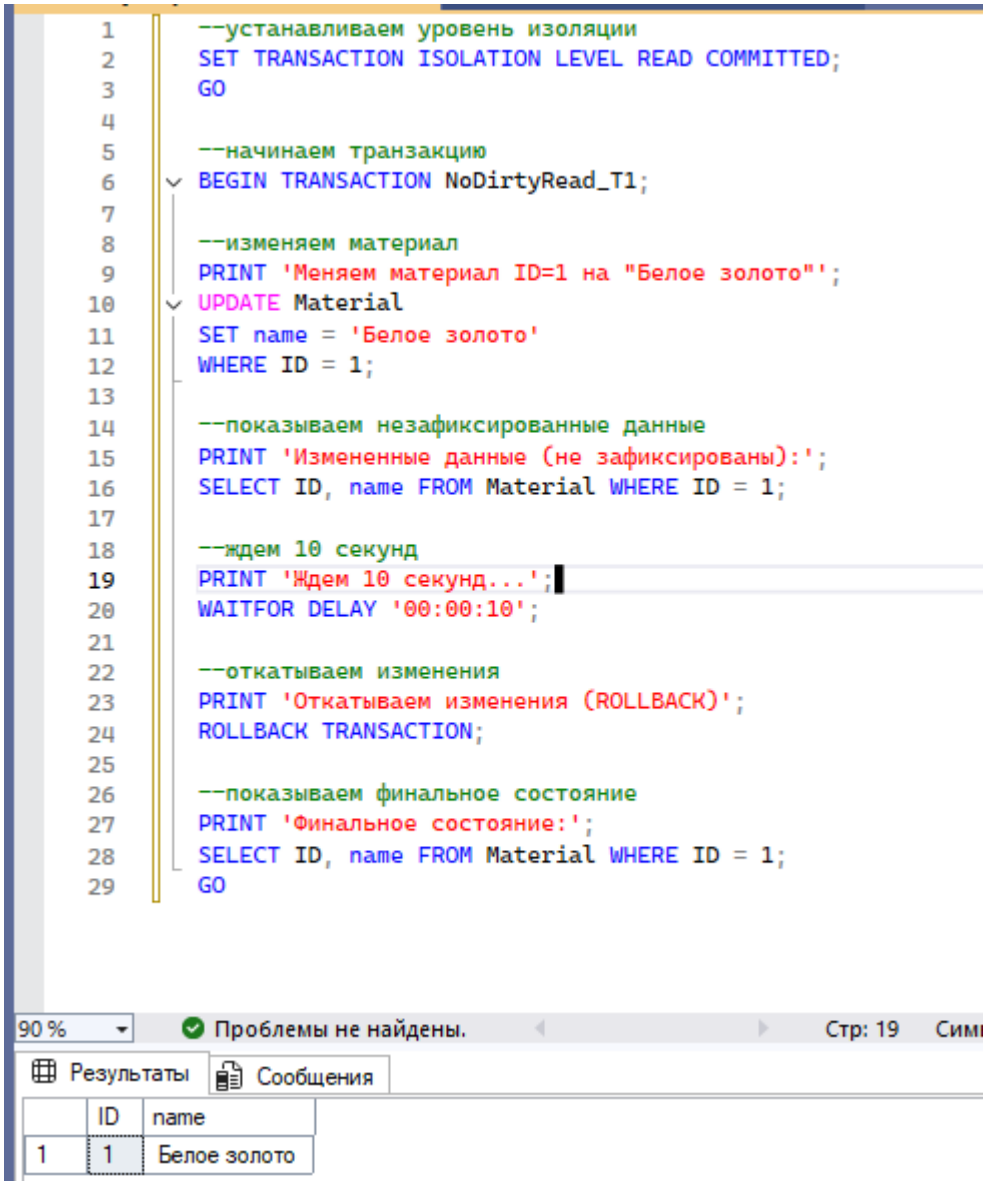
Уровень READ UNCOMMITTED позволяет читать незафиксированные данные других транзакций. Это приводит к грязному чтению - можно увидеть изменения, которые потом будут отменены (ROLLBACK).

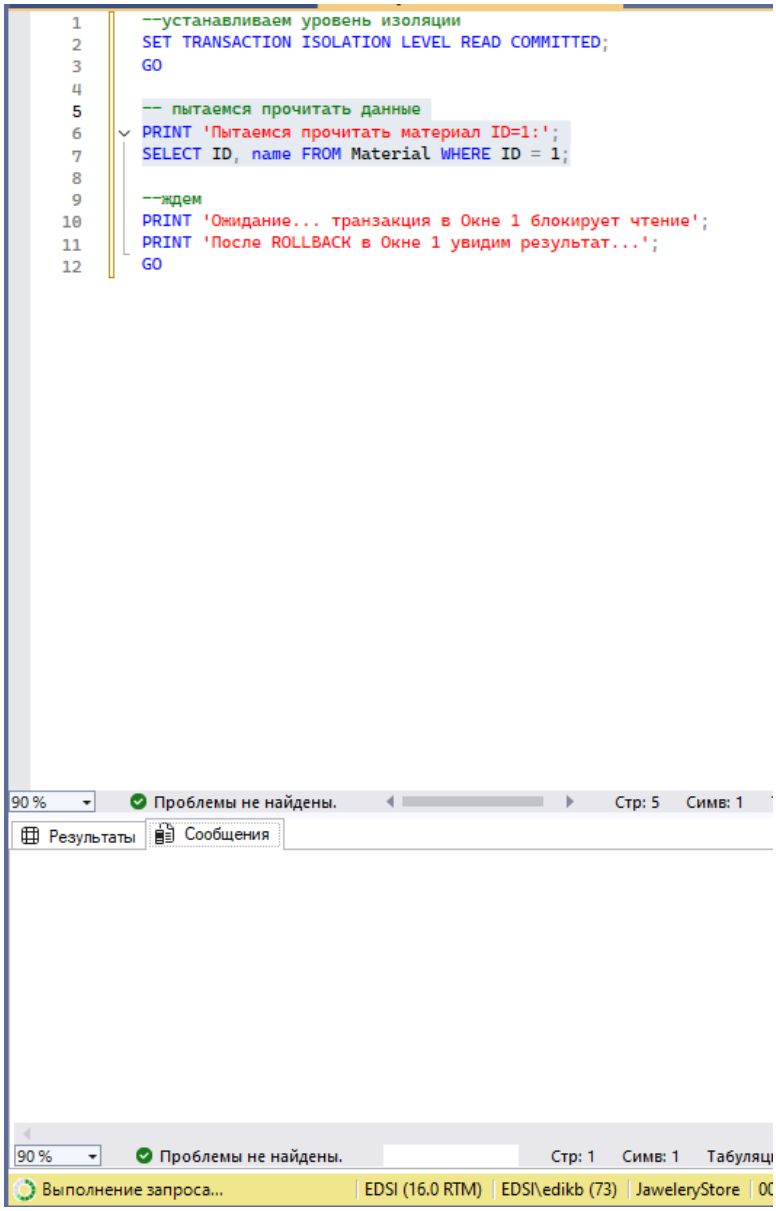
Например, транзакция видит новый статус контракта, который вскоре откатывается к исходному значению.

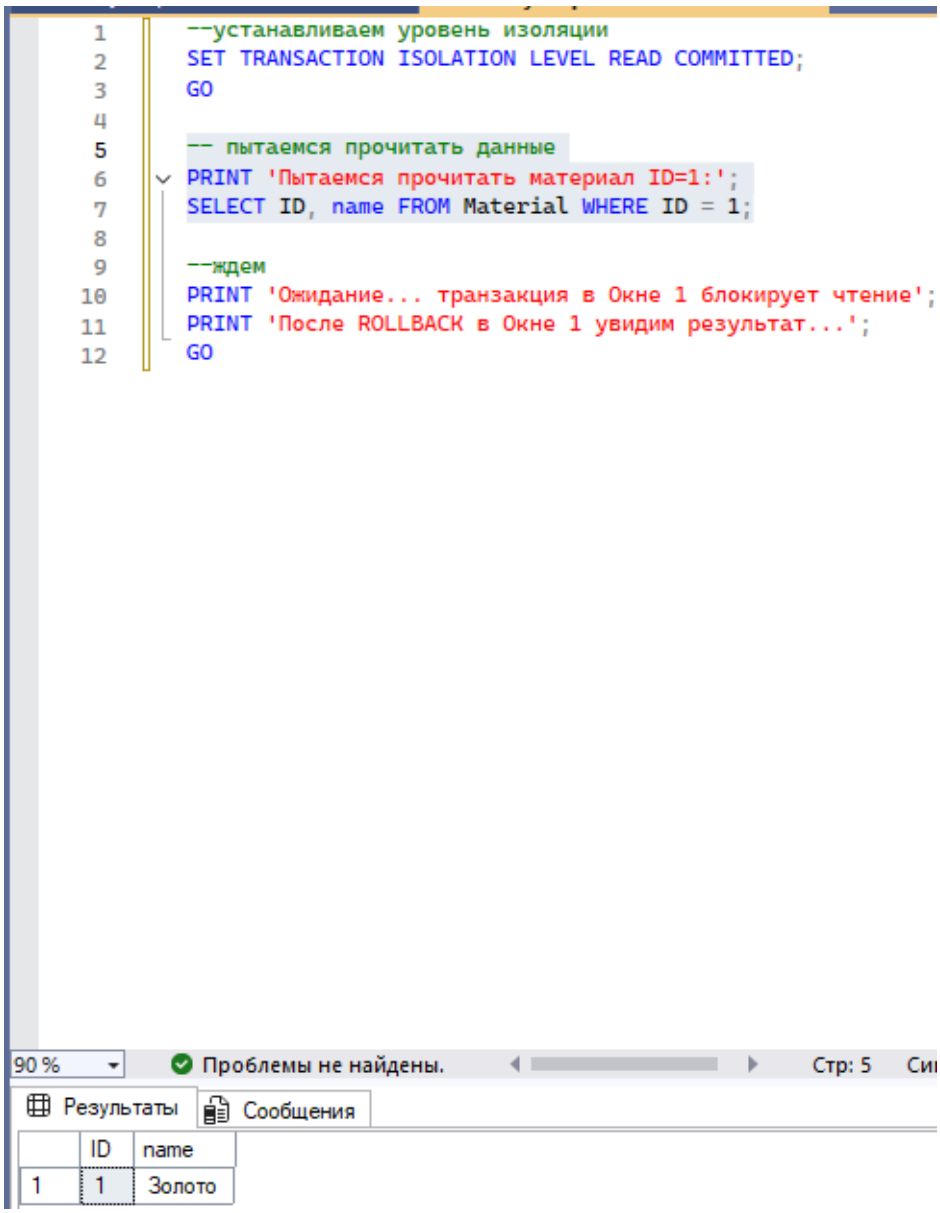
Вывод: *READ UNCOMMITTED обеспечивает высокую скорость работы, но полностью жертвует согласованностью данных. Подходит только для статистических запросов, где точность не критична.*

Уровень изоляции: READ COMMITTED

Сценарий 2: Проверка отсутствия грязного чтения




--устанавливаем уровень изоляции
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
GO
-- пытаемся прочитать данные
PRINT 'Пытаемся прочитать материал ID=1:';
SELECT ID, name FROM Material WHERE ID = 1;
--ждем
PRINT 'Ожидание... транзакция в Окне 1 блокирует чтение';
PRINT 'После ROLLBACK в Окне 1 увидим результат...';
GO
90 %
Проблемы не найдены.
Стр: 5
Симв: 1
Результаты
Сообщения
90 %
Проблемы не найдены.
Стр: 1
Симв: 1
Выполнение запроса...
EDSI (16.0 RTM)
EDSI\edikb (73)
JaweleryStore



Что выполнялось:

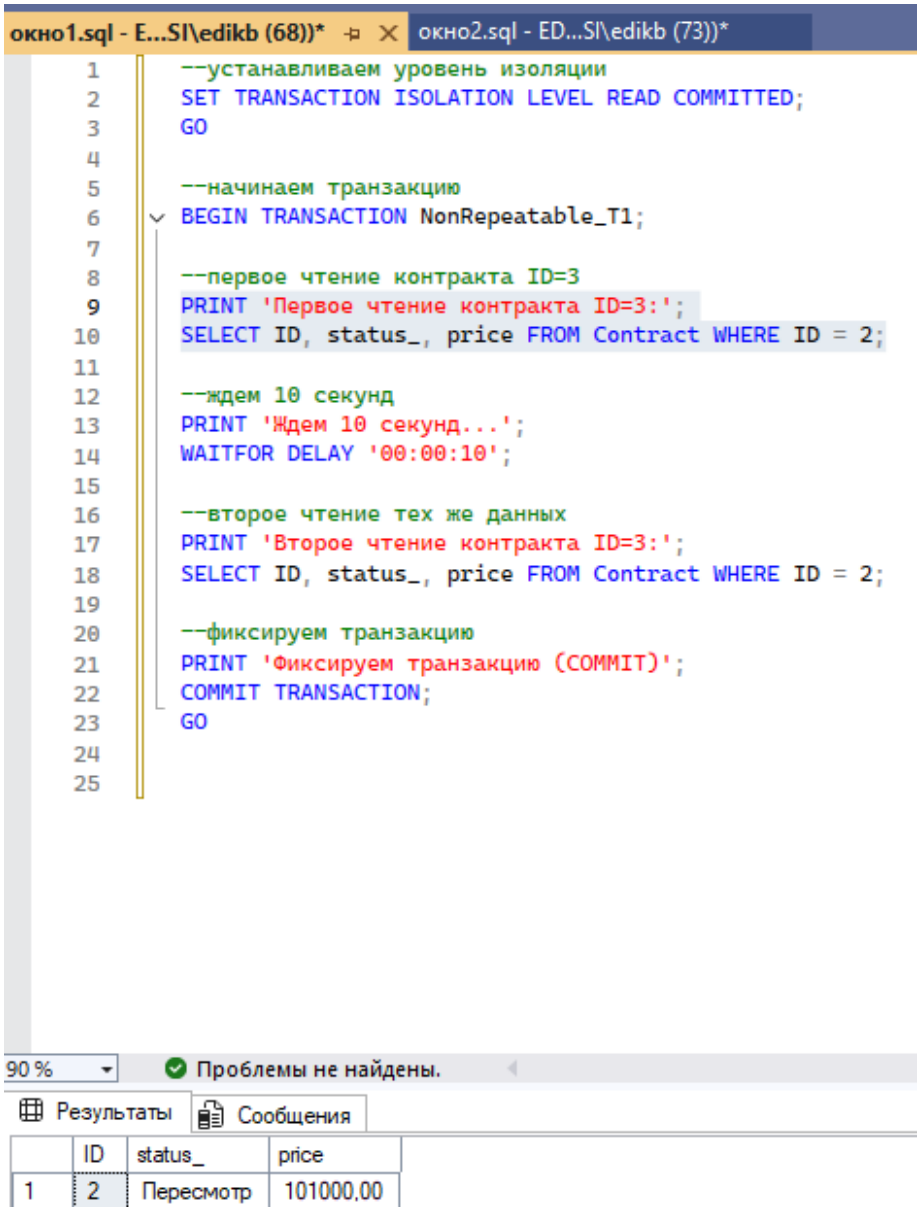
Окно 2 заблокировано на чтении

Окно 2 ждет завершения транзакции в Окне 1

После ROLLBACK в Окне 1 Окно 2 видит: name = 'Золото'

грязного чтения нет

Неповторяющееся чтение


окно1.sql - E...SI\edikb (68))*
окно2.sql - ED...SI\edikb (73))*
--устанавливаем уровень изоляции
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
GO
--начинаем транзакцию
BEGIN TRANSACTION NonRepeatable_T1;
--первое чтение контракта ID=3
PRINT 'Первое чтение контракта ID=3:';
SELECT ID, status_, price FROM Contract WHERE ID = 2;
--ждем 10 секунд
PRINT 'Ждем 10 секунд...';
WAITFOR DELAY '00:00:10';
--второе чтение тех же данных
PRINT 'Второе чтение контракта ID=3:';
SELECT ID, status_, price FROM Contract WHERE ID = 2;
--фиксируем транзакцию
PRINT 'Фиксируем транзакцию (COMMIT)';
COMMIT TRANSACTION;
GO
90 %
Проблемы не найдены.
Результаты
Сообщения
ID
status_
price
1
2
Пересмотр
101000,00

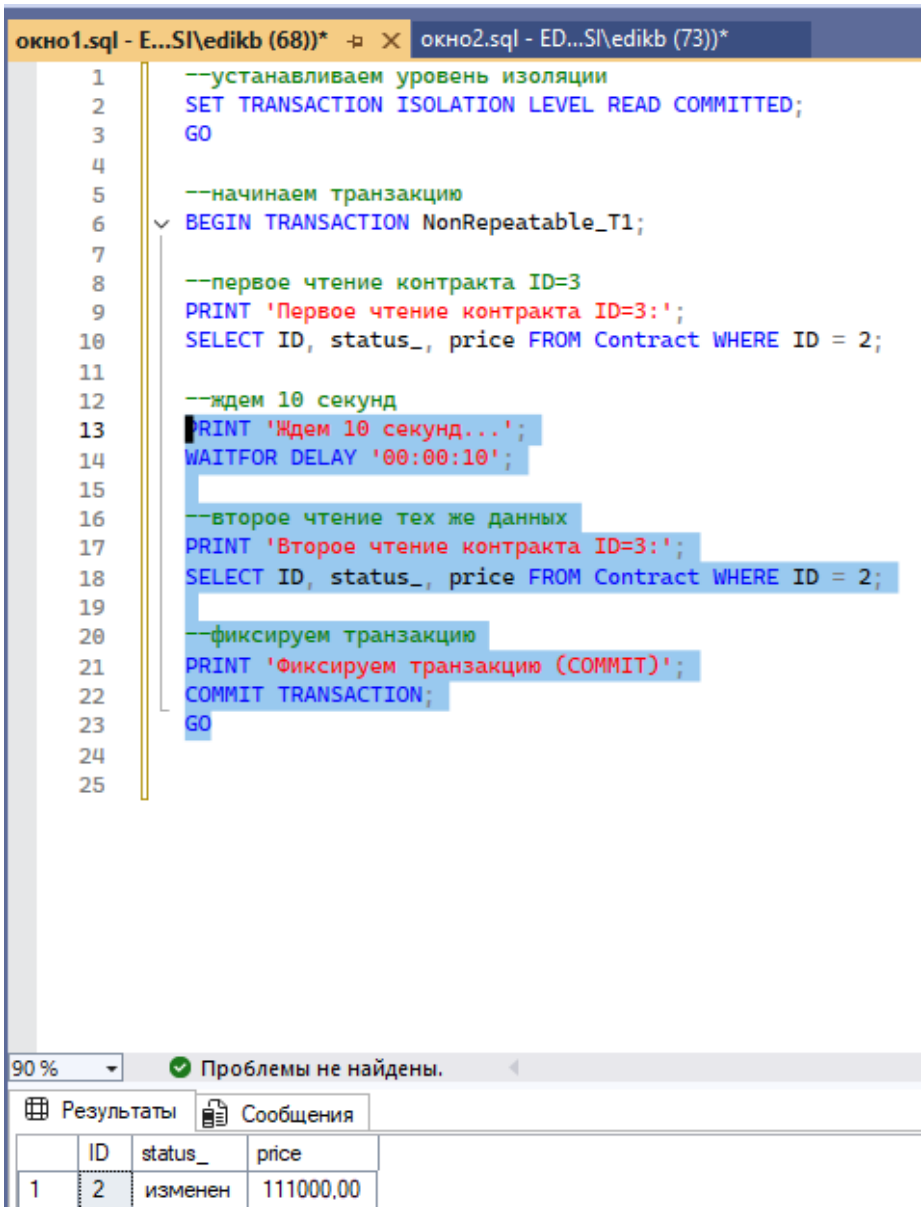
окно1.sql - ED...SI\edikb (68))* окно2.sql - E...SI\edikb (73))*

```sql
--устанавливаем уровень изоляции
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
GO

PRINT 'Ждем 2 секунды...';
WAITFOR DELAY '00:00:02';

--изменяем данные, которые читает Окно 1
PRINT 'Изменяем контракт ID=2:';
UPDATE Contract
SET
    status_ = 'изменен',
    price = price + 10000
WHERE ID = 2;

--показываем изменения
PRINT 'измененные данные:';
SELECT ID, status_, price FROM Contract WHERE ID = 2;
GO
COMMIT TRANSACTION;
```

90 % Проблемы не найдены.

Результаты Сообщения

	ID	status_	price
1	2	изменен	111000,00



Что было:

Окно 1 (первое чтение): status_ = 'Выполнен', price = 125000

Окно 2 изменяет: status_ = 'Изменен...', price = 135000

Окно 1 (второе чтение): status_ = 'Изменен...', price = 135000

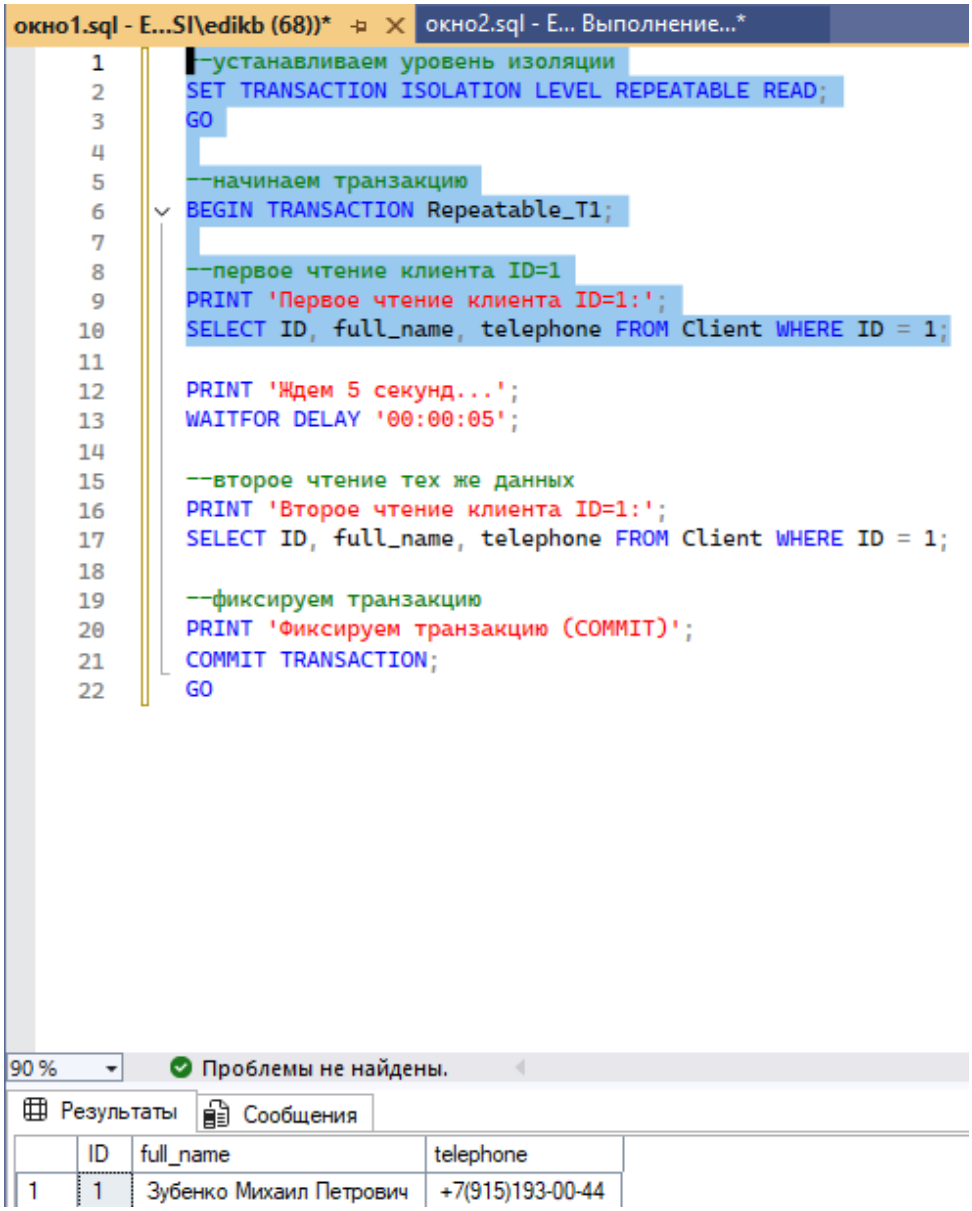
неповторяющееся чтение. данные изменились между чтениями

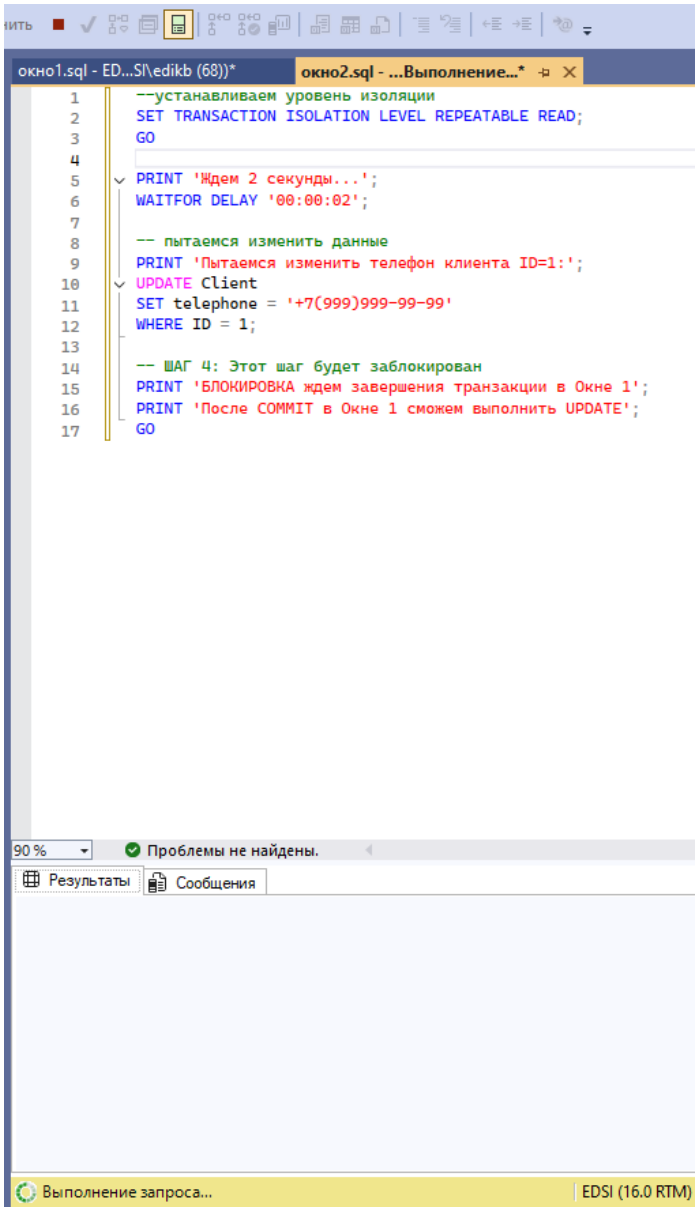
Уровень READ COMMITTED предотвращает грязное чтение - транзакции видят только зафиксированные данные. Незавершенные изменения других транзакций не отображаются.

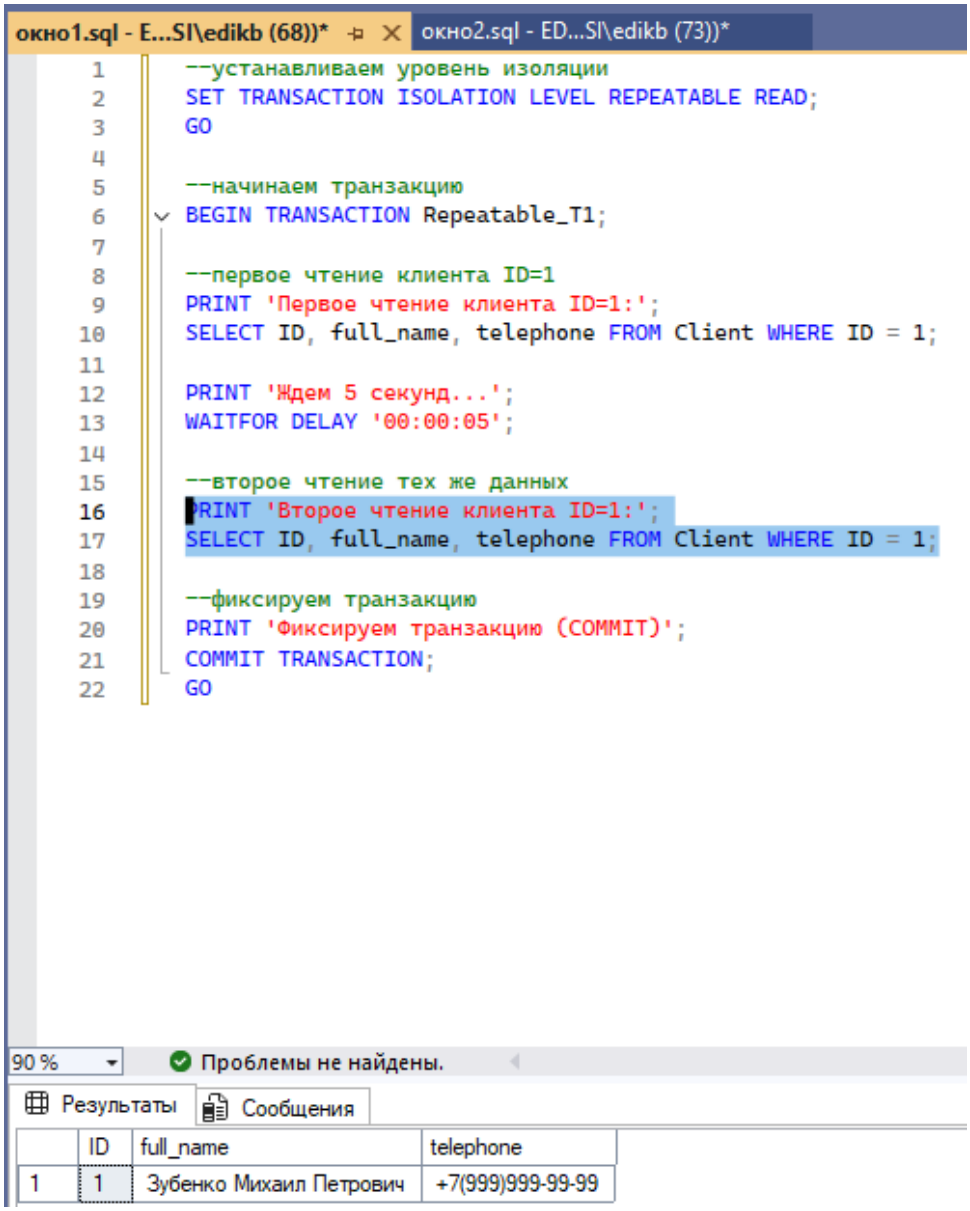
Однако допускается неповторяющееся чтение: между двумя чтениями одной транзакции другие транзакции могут изменить данные. Например, статус контракта может измениться между первым и вторым запросом.

Вывод: READ COMMITTED защищает от чтения "грязных" данных, но не гарантирует их стабильность в течение транзакции.

Уровень изоляции: REPEATABLE READ




окно1.sql - ED...SI\edikb (68))*
окно2.sql - ...Выполнение...*
--устанавливаем уровень изоляции
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
GO
PRINT 'Ждем 2 секунды...';
WAITFOR DELAY '00:00:02';
-- пытаемся изменить данные
PRINT 'Пытаемся изменить телефон клиента ID=1:';
UPDATE Client
SET telephone = '+7(999)999-99-99'
WHERE ID = 1;
-- ШАГ 4: Этот шаг будет заблокирован
PRINT 'БЛОКИРОВКА ждем завершения транзакции в Окне 1';
PRINT 'После COMMIT в Окне 1 сможем выполнить UPDATE';
GO
90 %
Проблемы не найдены.
Результаты
Сообщения
Выполнение запроса...
EDSI (16.0 RTM)



Что происходило:
Окно 1 блокирует строку клиента ID=1

Окно 2 не может изменить данные до COMMIT в Окне 1

Оба чтения в Окне 1 дают одинаковый результат

Неповторяющееся Чтение Устранено

Фантомное чтение

окно1.sql - E...SI\edikb (68))* | окно2.sql - ED...SI\edikb (73))*

```sql
--  уровень изоляции
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
GO

--начинаем транзакцию
BEGIN TRANSACTION Phantom_T1;

--первое чтение колец
PRINT 'Первое чтение: все кольца';
SELECT ID, product_name, type_of_product
FROM Jewelery
WHERE type_of_product = 'Кольцо'
ORDER BY ID;

PRINT 'Ждем 10 секунд...';
WAITFOR DELAY '00:00:05';

--второе чтение
PRINT 'Второе чтение: все кольца снова';
SELECT ID, product_name, type_of_product
FROM Jewelery
WHERE type_of_product = 'Кольцо'
ORDER BY ID;

-- ШАГ 6: Фиксируем транзакцию
PRINT 'Фиксируем транзакцию (COMMIT)';
COMMIT TRANSACTION;
GO
```

90 % Проблемы не найдены.

Результаты | Сообщения

	ID	product_name	type_of_product
1	1	Обручальное кольцо "Самсунг"	Кольцо
2	6	Перстень "Ведьмак"	Кольцо
3	9	Кольцо "Всевластия"	Кольцо
4	13	Кольцо "Пустое"	Кольцо
5	14	Кольцо "Роскошь"	Кольцо

окно1.sql - ED...SI\edikb (68))* | окно2.sql - E...SI\edikb (73))*

```sql
--устанавливаем уровень изоляции
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
GO

PRINT 'Ждем 2 секунды...';
WAITFOR DELAY '00:00:02';

--добавляем новое кольцо
PRINT 'Добавляем новое кольцо:';
INSERT INTO Jewelery (product_name, type_of_product, workshop_ID)
VALUES ('Кольцо "Аметистовое"', 'Кольцо', 1);

--показываем все кольца
PRINT 'Все кольца после добавления:';
SELECT ID, product_name, type_of_product
FROM Jewelery
WHERE type_of_product = 'Кольцо'
ORDER BY ID;
GO
```

90 % | Проблемы не найдены.

Результаты | Сообщения

```
Ждем 2 секунды...
Добавляем новое кольцо:

(Затронута 1 строка)
```

окно1.sql - E...SI\edikb (68))* | окно2.sql - ED...SI\edikb (73))*

```
1   --  уровень изоляции
2   SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
3   GO
4
5   --начинаем транзакцию
6   BEGIN TRANSACTION Phantom_T1;
7
8   --первое чтение колец
9   PRINT 'Первое чтение: все кольца';
10  SELECT ID, product_name, type_of_product
11  FROM Jewelery
12  WHERE type_of_product = 'Кольцо'
13  ORDER BY ID;
14
15  PRINT 'Ждем 10 секунд...';
16  WAITFOR DELAY '00:00:05';
17
18  --второе чтение
19  PRINT 'Второе чтение: все кольца снова';
20  SELECT ID, product_name, type_of_product
21  FROM Jewelery
22  WHERE type_of_product = 'Кольцо'
23  ORDER BY ID;
24
25  -- ШАГ 6: Фиксируем транзакцию
26  PRINT 'Фиксируем транзакцию (COMMIT)';
27  COMMIT TRANSACTION;
28  GO
```

90 % | Проблемы не найдены.

Результаты | Сообщения

	ID	product_name	type_of_product
1	1	Обручальное кольцо "Самсунг"	Кольцо
2	6	Перстень "Ведьмак"	Кольцо
3	9	Кольцо "Всевластия"	Кольцо
4	13	Кольцо "Пустое"	Кольцо
5	14	Кольцо "Роскошь"	Кольцо
6	17	Кольцо "Аметистовое"	Кольцо

Что происходило:

Окно 1 видит 5 колец (ID: 1, 6, 9, 13, 14)

Окно 2 добавляет новое кольцо

Окно 1 при втором чтении видит 6 колец

фантомное чтение появилась "фантомная" запись

Уровень REPEATABLE READ предотвращает изменение уже прочитанных данных другими транзакциями до завершения текущей. Это защищает от

неповторяющегося чтения - при повторном запросе данные остаются неизменными.

Однако уровень допускает фантомное чтение: другие транзакции могут добавлять новые записи, соответствующие условиям выборки. Например, при поиске колец может появиться новое кольцо между двумя чтениями.

Вывод: *REPEATABLE READ подходит для операций, требующих стабильности данных, но не защищает от появления новых записей*.

Уровень изоляции: SERIALIZABLE

Проверка отсутствия фантомного чтения

окно1.sql - E...SI\edikb (68))* | окно2.sql - ED...SI\edikb (73))*

```sql
--устанавливаем уровень изоляции
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
GO

--начинаем транзакцию
BEGIN TRANSACTION Serializable_T1;

--чтение диапазона контрактов
PRINT 'Первое чтение: контракты с ценой 50000-100000';
SELECT ID, status_, price
FROM Contract
WHERE price BETWEEN 50000 AND 100000
ORDER BY price;

PRINT 'Ждем 10 секунд...';
WAITFOR DELAY '00:00:05';

--повторное чтение
PRINT 'Второе чтение: тот же диапазон';
SELECT ID, status_, price
FROM Contract
WHERE price BETWEEN 50000 AND 100000
ORDER BY price;

--фиксируем транзакцию
PRINT 'Фиксируем транзакцию (COMMIT)';
COMMIT TRANSACTION;
GO
```

90 % Проблемы не найдены.

Результаты | Сообщения

	ID	status_	price
1	1	В обработке	50000,00
2	13	Выполнен	50000,00
3	18	Ожидание	50000,00
4	9	Выполнен	54300,00
5	12	Ожидание	67400,00
6	11	Выполнен	89200,00
7	5	Выполнен	95600,00

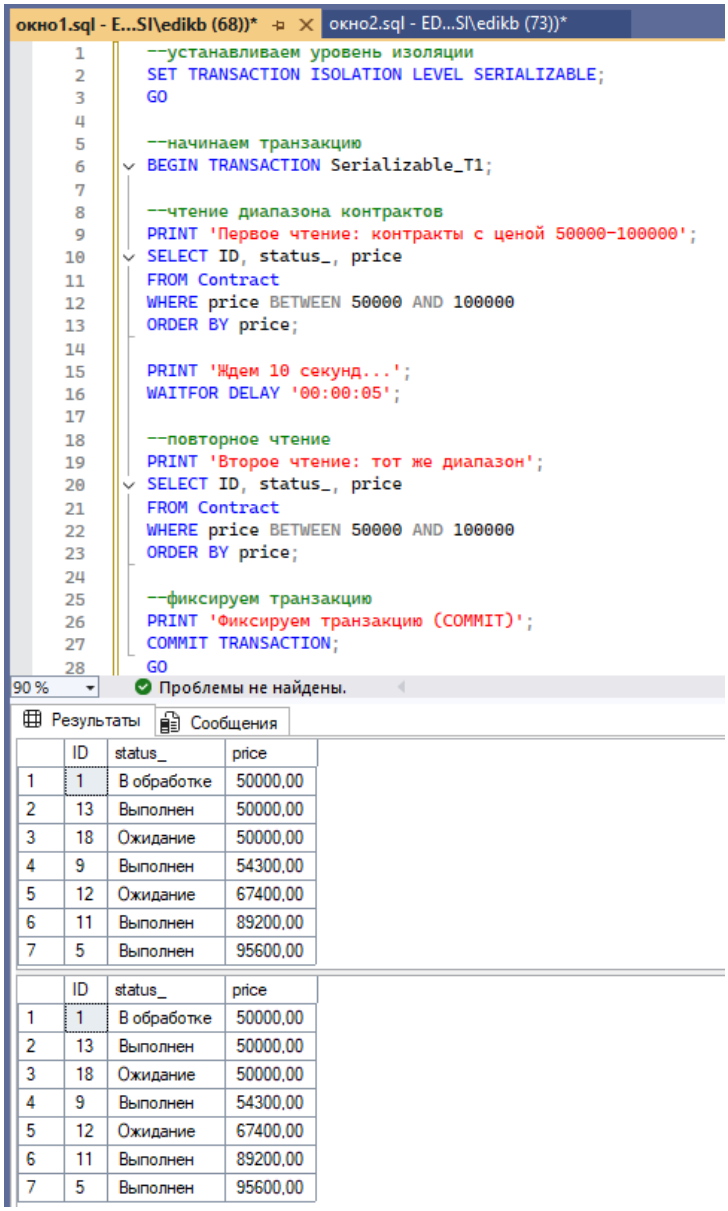
окно1.sql - ED...SI\edikb (68))* | окно2.sql - E...SI\edikb (73))*

```
1  SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
2  GO
3
4  PRINT 'Ждем 5 секунд...';
5  WAITFOR DELAY '00:00:05';
6
7  --начинаем транзакцию
8  BEGIN TRANSACTION Serializable_T2;
9
10 --пытаемся добавить контракт в диапазон
11 PRINT 'Пытаемся добавить контракт с ценой 75000:';
12 INSERT INTO Contract (data, status_, price, client_ID, jewelery_ID, discount_ID)
13 VALUES (GETDATE(), 'Сериализуемый', 75000, 2, 3, 2);
14
15 --этот шаг будет заблокирован
16 PRINT 'БЛОКИРОВКА! Ждем завершения транзакции в Окне 1...';
17
18 --фиксируем (после разблокировки)
19 COMMIT TRANSACTION;
20 PRINT 'Транзакция зафиксирована после разблокировки';
21 GO
```

0 % | Проблемы не найдены. | Стр: 21 | Си

Сообщения

```
Ждем 5 секунд...
Пытаемся добавить контракт с ценой 75000:

(Затронута 1 строка)
БЛОКИРОВКА! Ждем завершения транзакции в Окне 1...
Транзакция зафиксирована после разблокировки
```



Что произошло:
Окно 1 устанавливает диапазонную блокировку на price 50000 - 100000

Окно 2 не может вставить данные в этот диапазон

Оба чтения в окно 1 дают одинаковый результат

фантомное чтение устранено

При уровне изоляции SERIALIZABLE все аномалии параллельного доступа полностью предотвращаются, в том числе фантомные чтения. Это

означает, что если одна транзакция читает данные по определенному условию (например, диапазон значений), другие транзакции не смогут добавлять, изменять или удалять строки, подпадающие под это условие, до завершения первой транзакции.

Такой эффект достигается благодаря диапазонным блокировкам, которые "замораживают" не только конкретные строки, но и потенциальные места для новых записей, соответствующих условию выборки. В результате система обеспечивает полную изолированность транзакций, выполняя их так, будто они обрабатываются строго по очереди.

Итог: Уровень SERIALIZABLE гарантирует максимальную целостность данных и защиту от всех видов аномалий, однако существенно ограничивает параллельную обработку и снижает общую производительность системы. Поэтому он применяется только для критически важных операций, где абсолютная корректность данных имеет приоритет над скоростью работы.

Приобретенные навыки:

- Практическая работа с уровнями изоляции
- Понимание блокировок
- Умение выбирать уровень изоляции под задачу
- Навык отладки проблем параллельного доступа

Содержание отчета

- Сценарий и протокол его выполнения.
- Краткие выводы о навыках, приобретенных в ходе выполнения работы.