|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ | | | | | | | | |
|  | | |  | | |  | | |
| ФГБОУ ВО «Пермский государственный национальный исследовательский университет» | | | | | | | | |
|  | | |  | | |  | | |
|  | | ОТЧЕТ  по лабораторной работе №7 «Транзакции» по дисциплине «Базы данных и СУБД» | | | | |  | |
|  | | |  | | |  | | |
|  | Работу выполнил  студент гр. ПМИ-6  Шукшина М.И.  «29» марта 2018 | | |  | Проверил ассистент кафедры МОВС  Костарева Т.А.  «29» марта 2018 | | |  |
|  |  | | |  |  | | |  |
| Пермь 2018 | | | | | | | | |

ПОСТАНОВКА ЗАДАЧИ

1. Продемонстрировать выполняемость всех основных свойств транзакций (ACID).

* Для A рекомендуется провести операции добавления и изменения, откатить ее, убедиться, что полностью откатилась.
* Для C рекомендуется создать таблицу с ограничением на столбец (CONSTRAINT NAME\_GE\_Zero CHECK (ColumnName>= 0)), провести транзакцию, приводящую к несогласованности, убедиться, что откатилась.
* Для I необходимо параллельно запустить несколько транзакций, посмотреть как они все отработают (см. WAITFOR DELAY '00:02:00').
* Для D рекомендуется не разделяя скрипт операторами ‘go’ аварийно завершить работу сервера (SHUTDOWN WITH NOWAIT;) после выполнения скрипта, перезапустить (Запустить заново сервер можно с помощью SqlServerConfigurationManager, утилита в комплекте с ManagementStudio), убедиться, что применились изменения.

2. Привести примеры для всех 4 проблем параллельного доступа.

3. Решить проблемы из пункта 2 включая необходимые режимы изоляции данных.

СВОЙСТВА ТРАНЗАКЦИЙ (ACID)

1. Для A рекомендуется провести операции добавления и изменения, откатить ее, убедиться, что полностью откатилась.

DECLARE @errorMessage nvarchar(4000), @errorSeverity int

begin transaction ins\_up

begin try

insert into [Данные\_о\_товарах] values ('Русалка',5,2, 'шт',789)

update [Данные\_о\_товарах] set [код группы товара] = 1

where [наименование товара] = 'Кошечка' OR [наименование товара] = 'Кошка'

select \* from [Данные\_о\_товарах]

end try

begin catch

if @@TRANCOUNT > 0 rollback transaction ins\_up

select @errorMessage = ERROR\_MESSAGE(), @errorSeverity = ERROR\_SEVERITY()

raiserror (@errorMessage, @errorSeverity, 1)

end catch

rollback transaction ins\_up

select \* from [Данные\_о\_товарах]

На *Рисунке 1* мы можем увидеть, что товар «Кошка» имеет код группы товара «2» и нет товара с наименованием «Русалка». Это таблица «*Данные\_о\_товарах*» до транзакции.

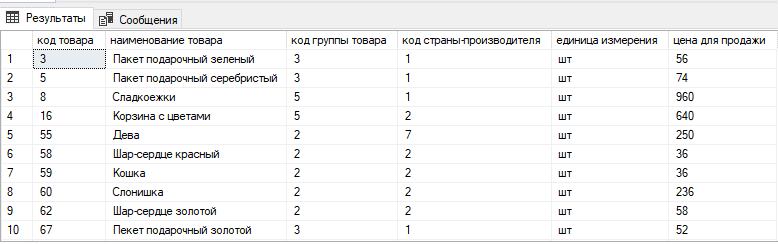


Рисунок 1 - Таблица «Данные\_о\_товарах» до транзакции

После выполнения транзакции код товара с наименованием «Кошка» - «1» (*Рисунок 2*), и есть добавился товар с наименованием «Русалка» (*Рисунок 3*).

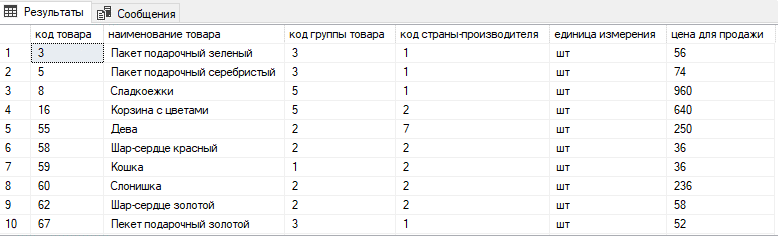


Рисунок 2 - Таблица «Данные\_о\_товарах» после транзакции

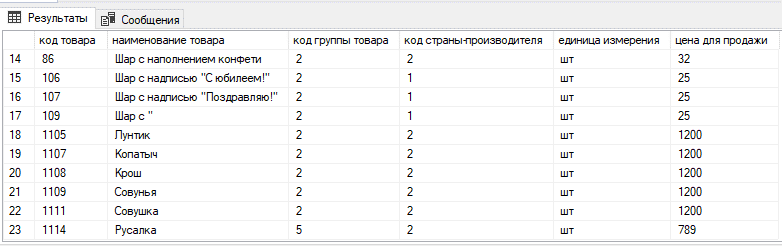


Рисунок 3 - Таблица «Данные\_о\_товарах» после транзакции

После отката транзакции все стало так, как было до транзакции. Можно увидеть на *Рисунке 4* и *Рисунке 5*.

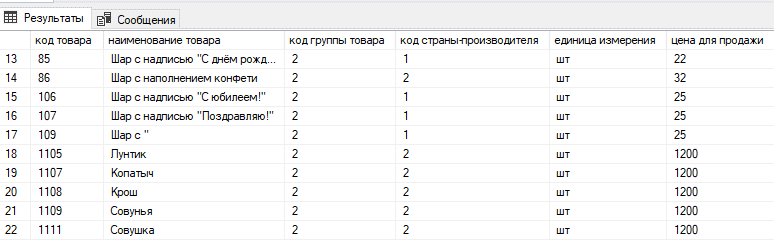


Рисунок 4 - Таблица «Данные\_о\_товарах» после отката

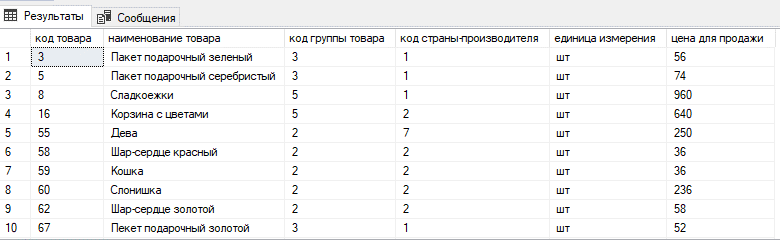


Рисунок 5 - Таблица «Данные\_о\_товарах» после отката

1. Для C рекомендуется создать таблицу с ограничением на столбец (CONSTRAINT NAME\_GE\_Zero CHECK (ColumnName>= 0)), провести транзакцию, приводящую к несогласованности, убедиться, что откатилась.

Создадим ограницение для таблицы «Данные\_о\_товарах» на столбец «цена для продажи».

ALTER TABLE [dbo].[Данные\_о\_товарах] WITH CHECK ADD CONSTRAINT [FK\_Данные\_о\_товарах\_цена\_для\_продажи] CHECK ([цена для продажи] >= 0)

Запустим транзакцию, приводящую к несогласованности (*Рисунок 6*), и на *Рисунке 7* видим, что она откатилась.

DECLARE @errorMessage nvarchar(4000), @errorSeverity int

begin transaction ins\_zero

begin try

insert into [Данные\_о\_товарах] values ('Русалка',5,2, 'шт', -1)

select \* from [Данные\_о\_товарах]

commit

end try

begin catch

if @@TRANCOUNT > 0 rollback transaction ins\_zero

select @errorMessage = ERROR\_MESSAGE(), @errorSeverity = ERROR\_SEVERITY()

raiserror (@errorMessage, @errorSeverity, 1)

end catch

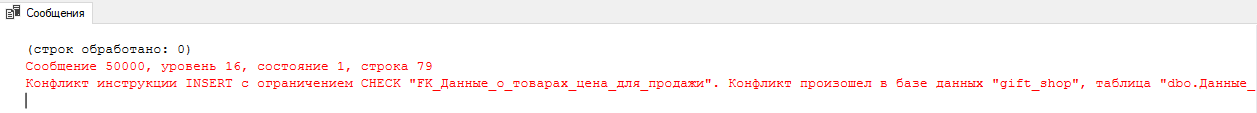


Рисунок 6 - Несогласованность

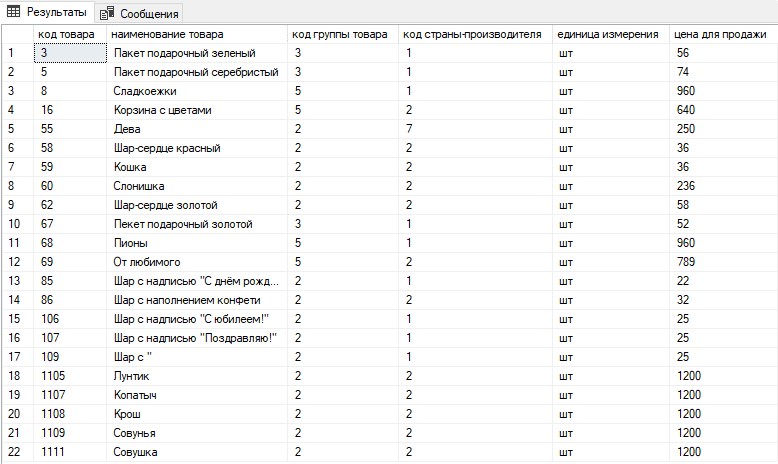


Рисунок 7 - После отката транзакции

1. Для I необходимо параллельно запустить несколько транзакций и посмотреть, как они все отработают (см. WAITFOR DELAY '00:02:00').

Первая транзакция:

DECLARE @errorMessage nvarchar(4000), @errorSeverity int

begin transaction ins

begin try

insert into [Данные\_о\_товарах] values ('Шар с надписью "Принцессе"',2,1, 'шт', 25)

select \* from [Данные\_о\_товарах]

WAITFOR DELAY '00:00:10'

select \* from [Данные\_о\_товарах]

commit transaction ins

end try

begin catch

if @@TRANCOUNT > 0 rollback transaction ins

select @errorMessage = ERROR\_MESSAGE(), @errorSeverity = ERROR\_SEVERITY()

raiserror (@errorMessage, @errorSeverity, 1)

end catch

Вторая транзакция:

DECLARE @errorMessage nvarchar(4000), @errorSeverity int

begin transaction ins

begin try

insert into [Данные\_о\_товарах] values ('Фея',5,2, 'шт',789)

select \* from [Данные\_о\_товарах]

commit transaction ins

end try

begin catch

if @@TRANCOUNT > 0 rollback transaction ins

select @errorMessage = ERROR\_MESSAGE(), @errorSeverity = ERROR\_SEVERITY()

raiserror (@errorMessage, @errorSeverity, 1)

end catch

Результат параллельного запуска выдал следующие результаты (*Рисунок 8*):

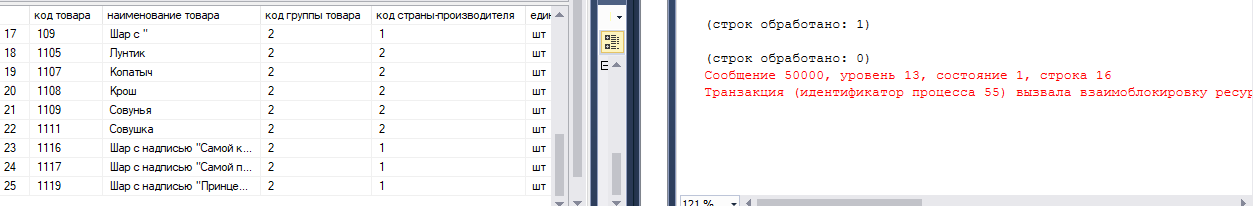


Рисунок 8 - Результат параллельного запуска

1. Для D рекомендуется, не разделяя скрипт операторами ‘go’, аварийно завершить работу сервера (SHUTDOWN WITH NOWAIT;) после выполнения скрипта, перезапустить (Запустить заново сервер можно с помощью SqlServerConfigurationManager, утилита в комплекте с ManagementStudio), убедиться, что применились изменения.

DECLARE @errorMessage nvarchar(4000), @errorSeverity int

begin transaction ins

begin try

insert into [Данные\_о\_товарах] values ('Шар с надписью "Самой милой"',2,1, 'шт', 25)

select \* from [Данные\_о\_товарах]

WAITFOR DELAY '00:00:10'

select \* from [Данные\_о\_товарах]

commit transaction ins

end try

begin catch

if @@TRANCOUNT > 0 rollback transaction ins

select @errorMessage = ERROR\_MESSAGE(), @errorSeverity = ERROR\_SEVERITY()

raiserror (@errorMessage, @errorSeverity, 1)

end catch

SHUTDOWN WITH NOWAIT

После запуска транзакции аварийно завершилась работа сервера (*Рисунок 9*). После запуска можем заметить, что изменения совершились (*Рисунок 10*).

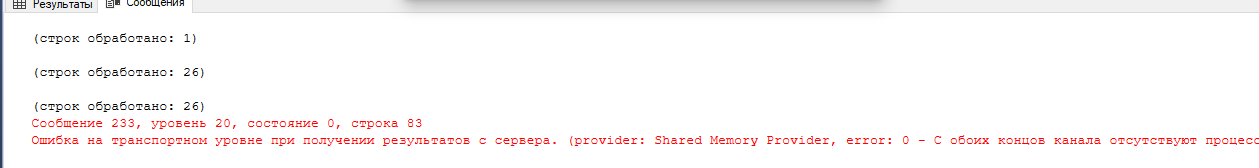


Рисунок 9 - Аварийное завершение



Рисунок 10 - Таблица «Данные о товарах» после аварийного завершения

ПРОБЛЕМЫ ПАРАЛЛЕЛЬНОГО ДОСТУПА

1. Потерянное обновление: при одновременном изменении одного блока данных разными транзакциями, одно из изменений теряется;

Первая транзакция:

DECLARE @errorMessage nvarchar(4000), @errorSeverity int

begin transaction ins\_up\_sl

begin try

WAITFOR DELAY '00:00:05'

update [Данные\_о\_товарах] set [наименование товара] = 'Шар с конфети'

where [наименование товара] = 'Шар с сюрпризом'

select \* from [Данные\_о\_товарах]

end try

begin catch

if @@TRANCOUNT > 0 rollback transaction ins\_up\_sl

select @errorMessage = ERROR\_MESSAGE(), @errorSeverity = ERROR\_SEVERITY()

raiserror (@errorMessage, @errorSeverity, 1)

end catch

Вторая транзакция:

DECLARE @errorMessage nvarchar(4000), @errorSeverity int

begin transaction ins\_up1

begin try

update [Данные\_о\_товарах] set [наименование товара] = 'Шар с запиской'

where [наименование товара] = 'Шар с сюрпризом'

select \* from [Данные\_о\_товарах]

commit transaction ins\_up1

end try

begin catch

if @@TRANCOUNT > 0 rollback

select @errorMessage = ERROR\_MESSAGE(), @errorSeverity = ERROR\_SEVERITY()

raiserror (@errorMessage, @errorSeverity, 1)

end catch

Результат параллельного доступа транзакций (*Рисунок 11*) при режиме изоляции данных:

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

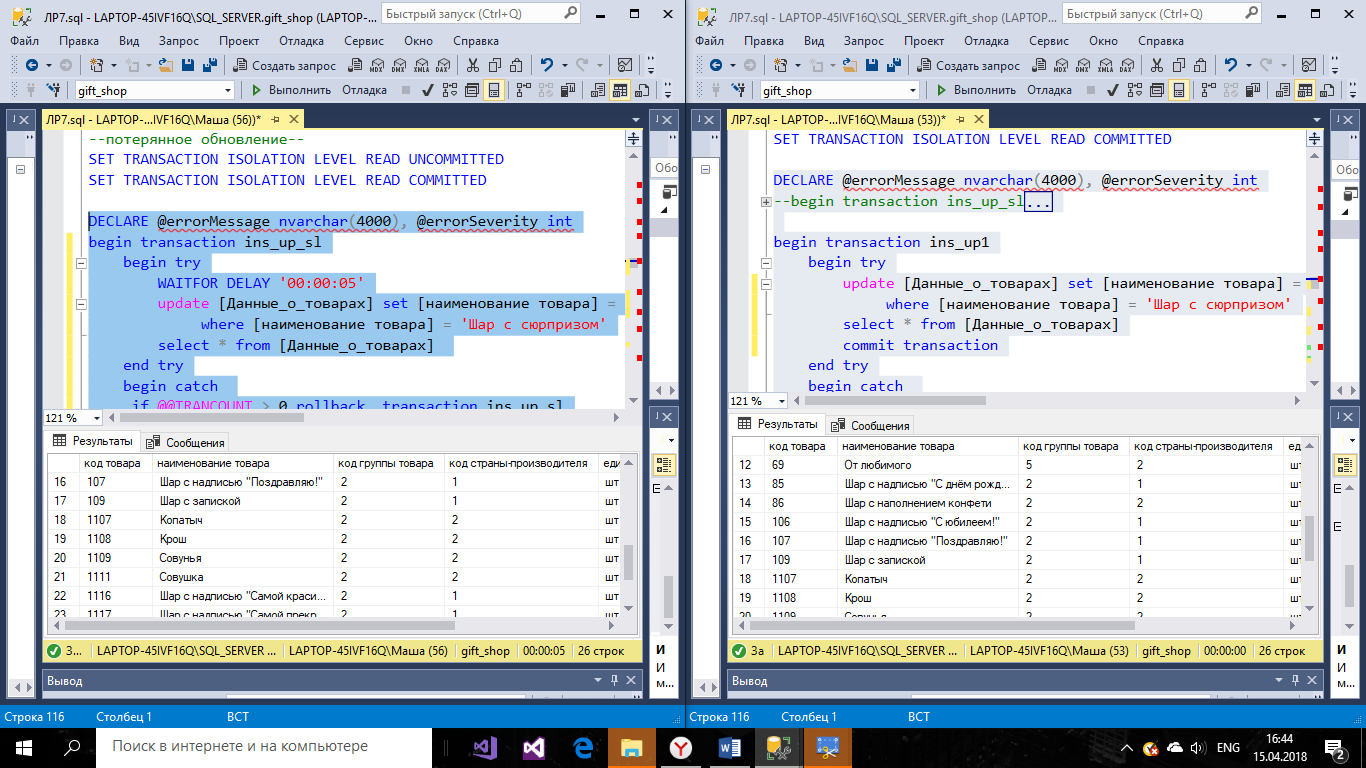


Рисунок 11 - Потерянное обновление

Результат параллельного доступа транзакций (*Рисунок 12*) при режиме изоляции данных:

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

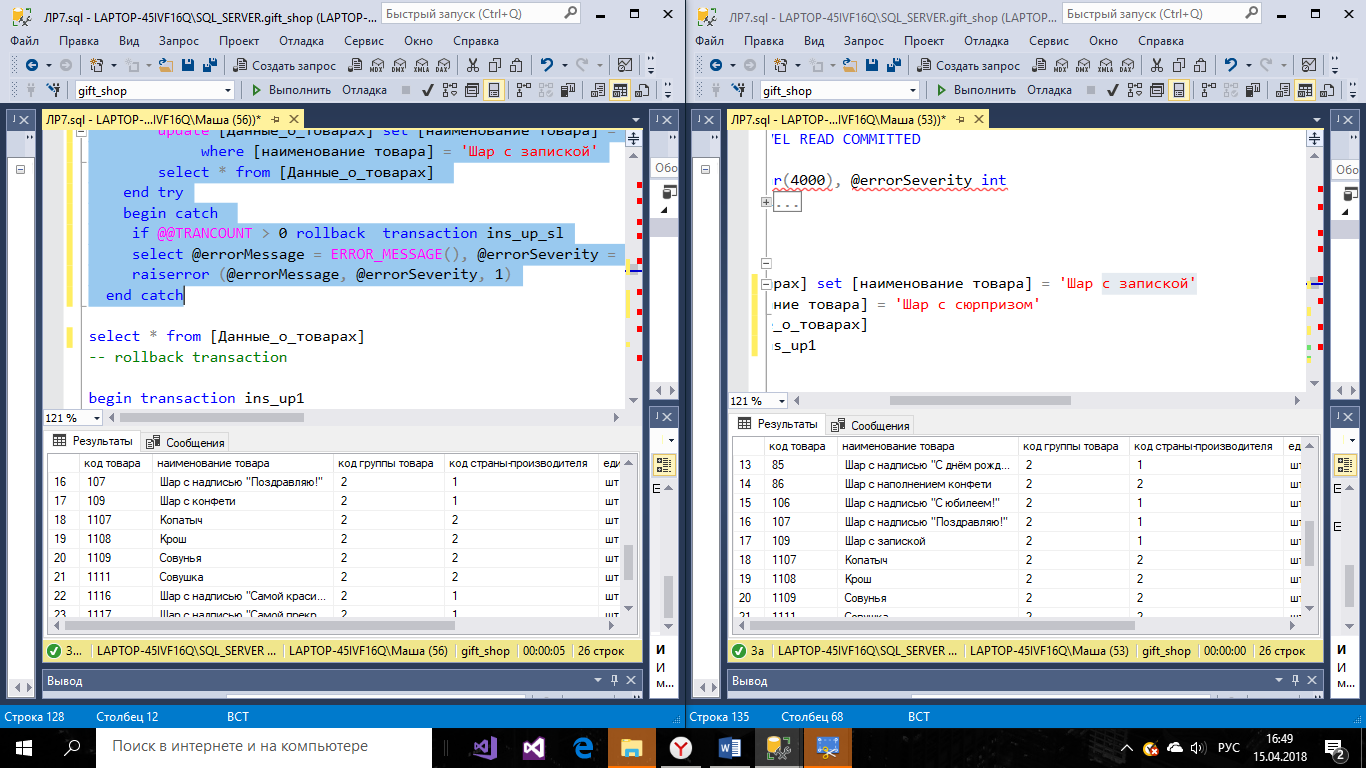


Рисунок 12 - Потерянное обновление

1. «Грязное» чтение: чтение данных, добавленных или изменённых транзакцией, которая впоследствии не подтвердится (откатится);

Первая транзакция:

DECLARE @errorMessage nvarchar(4000), @errorSeverity int

begin transaction ins\_up\_sl

begin try

declare @price int

select @price = [цена для продажи] from [Данные\_о\_товарах] where [наименование товара] = 'Кошка'

update [Данные\_о\_товарах] set [цена для продажи] = @price + 50

where [наименование товара] = 'Кошка'

waitfor delay '00:00:10'

rollback transaction ins\_up\_sl

select \* from [Данные\_о\_товарах]

end try

begin catch

if @@TRANCOUNT > 0 rollback transaction ins\_up\_sl

select @errorMessage = ERROR\_MESSAGE(), @errorSeverity = ERROR\_SEVERITY()

raiserror (@errorMessage, @errorSeverity, 1)

end catch

Вторая транзакция:

DECLARE @errorMessage nvarchar(4000), @errorSeverity int

begin transaction ins\_up1

begin try

declare @price int

select @price = [цена для продажи] from [Данные\_о\_товарах] where [наименование товара] = 'Кошка'

update [Данные\_о\_товарах] set [цена для продажи] = @price + 50

where [наименование товара] = 'Кошка'

commit transaction

select \* from [Данные\_о\_товарах]

end try

begin catch

if @@TRANCOUNT > 0 rollback

select @errorMessage = ERROR\_MESSAGE(), @errorSeverity = ERROR\_SEVERITY()

raiserror (@errorMessage, @errorSeverity, 1)

end catch

Результат параллельного доступа транзакций (*Рисунок 13*) при режиме изоляции данных:

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

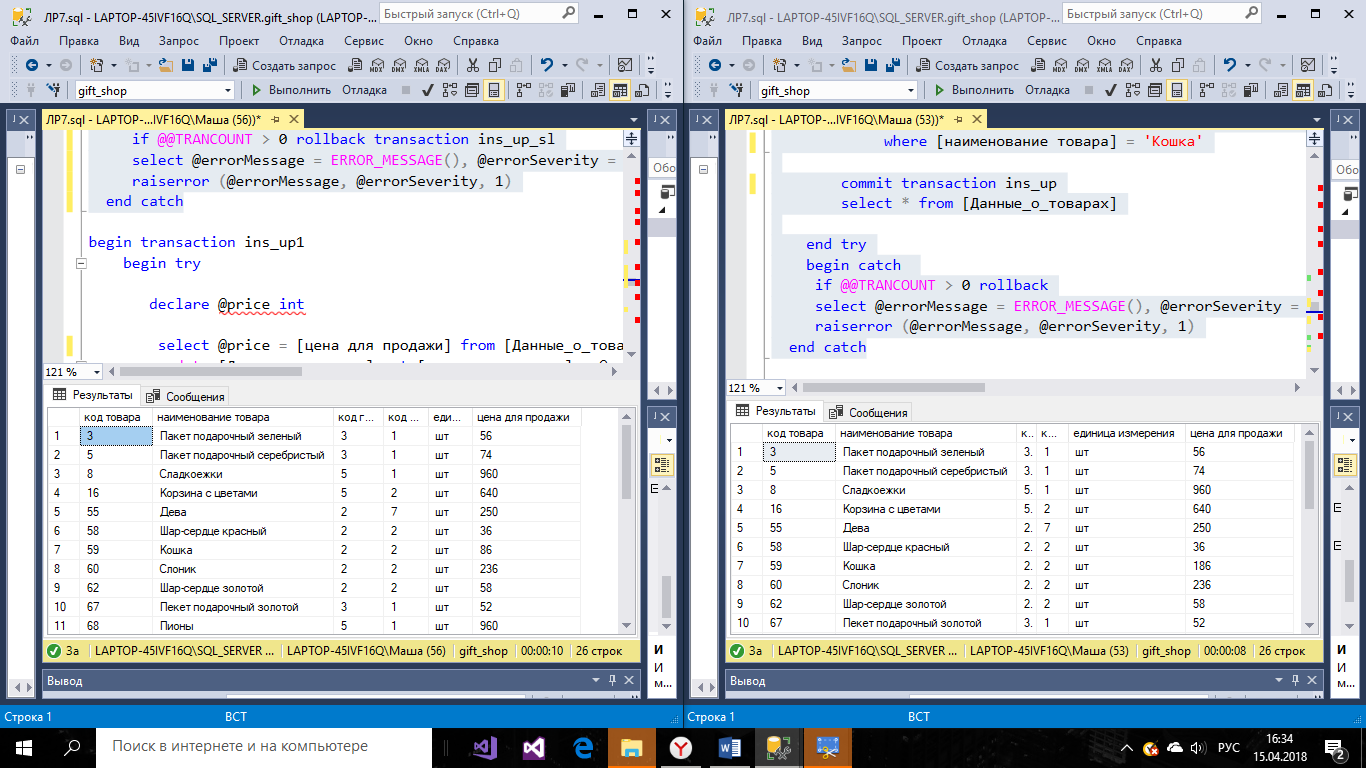


Рисунок 13 - "Грязное" чтение

Результат параллельного доступа транзакций (*Рисунок 14*) при режиме изоляции данных:

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

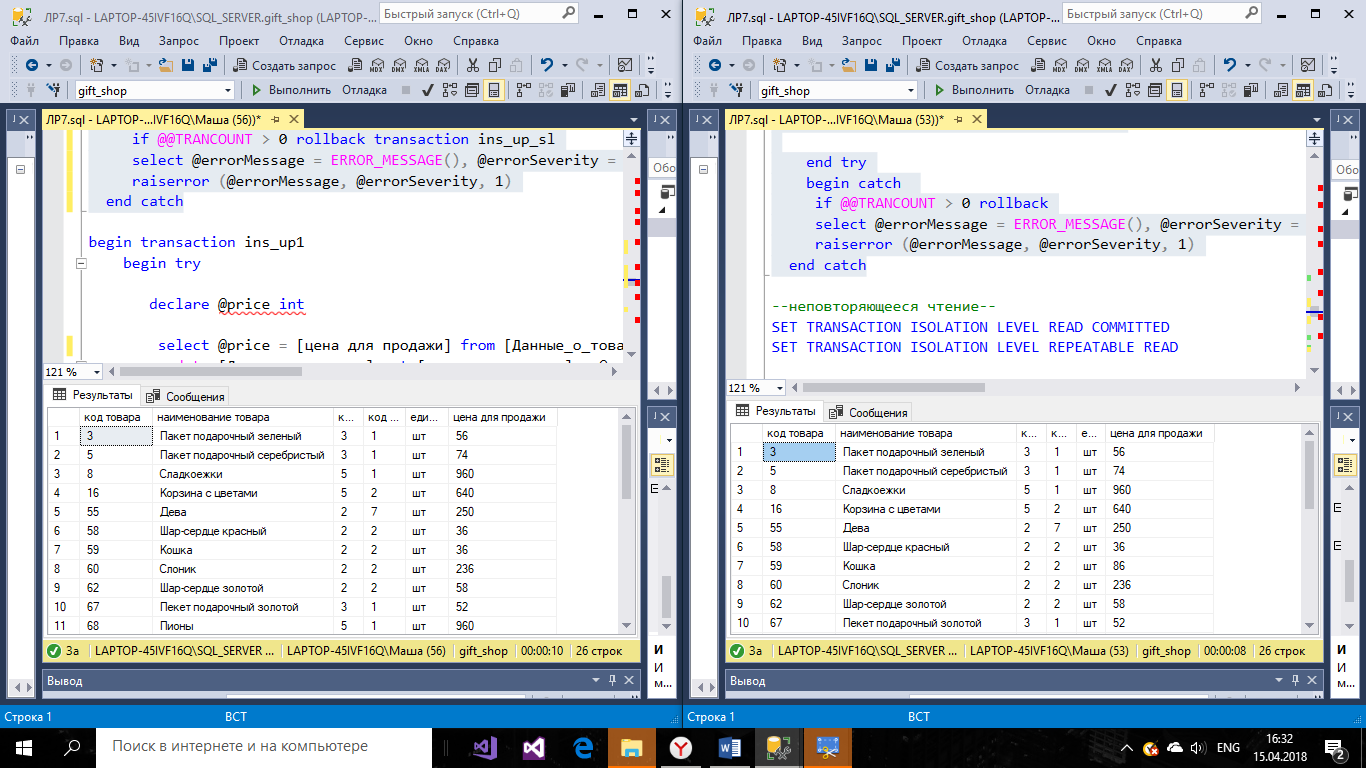


Рисунок 14 - "Грязное" чтение

1. Неповторяющееся чтение: при повторном чтении в рамках одной транзакции, ранее прочитанные данные, оказываются изменёнными или удалёнными;

Первая транзакция:

DECLARE @errorMessage nvarchar(4000), @errorSeverity int

begin transaction up

begin try

select \* from [Данные\_о\_товарах]

waitfor delay '00:00:10'

select \* from [Данные\_о\_товарах]

commit transaction up

end try

begin catch

if @@TRANCOUNT > 0 rollback transaction up

select @errorMessage = ERROR\_MESSAGE(), @errorSeverity = ERROR\_SEVERITY()

raiserror (@errorMessage, @errorSeverity, 1)

end catch

Вторая транзакция:

DECLARE @errorMessage nvarchar(4000), @errorSeverity int

begin transaction up

begin try

update [Данные\_о\_товарах] set [код группы товара] = 2

where [наименование товара] = 'Дева'

commit transaction

end try

begin catch

if @@TRANCOUNT > 0 rollback

select @errorMessage = ERROR\_MESSAGE(), @errorSeverity = ERROR\_SEVERITY()

raiserror (@errorMessage, @errorSeverity, 1)

end catch

Результат параллельного доступа транзакций (*Рисунок 15*) при режиме изоляции данных:

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

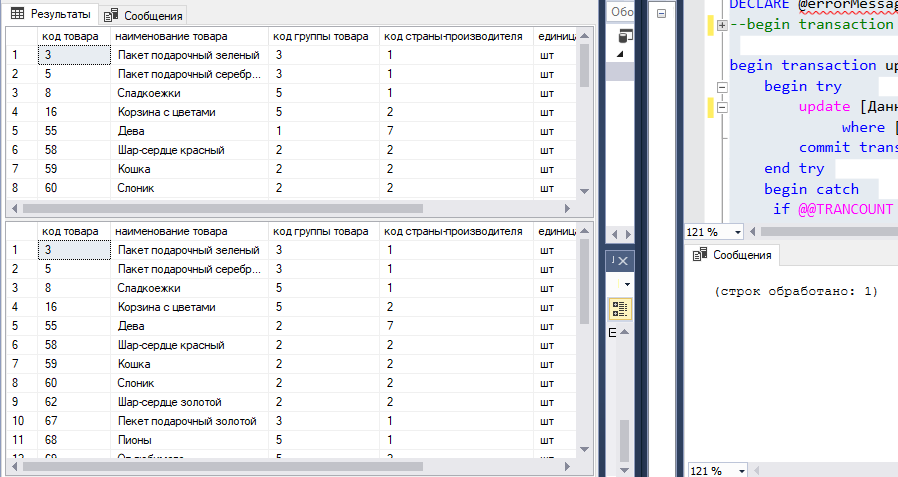


Рисунок 15 - Неповторяющееся чтение

Результат параллельного доступа транзакций (*Рисунок 16*) при режиме изоляции данных:

ET TRANSACTION ISOLATION LEVEL REPEATABLE READ

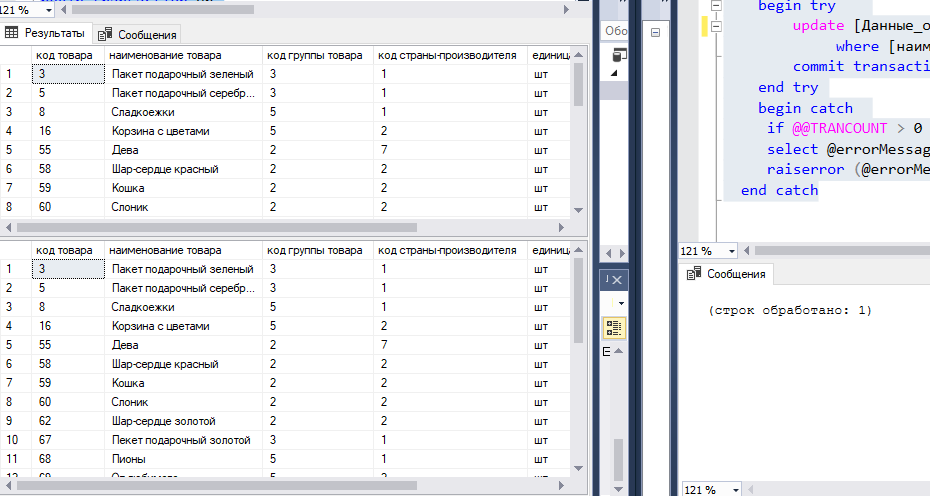


Рисунок 16 - Неповторяющееся чтение

1. Фантомное чтение: при повторном чтении в рамках одной транзакции прочитаны данные (новые "фантомные" строки), которых при предыдущих чтениях не было, хотя они удовлетворяли условиям отбора предыдущего чтения.

Первая транзакция:

DECLARE @errorMessage nvarchar(4000), @errorSeverity int

begin transaction ins

begin try

select SUM([цена для продажи]) from [Данные\_о\_товарах] where [код группы товара] = 2

WAITFOR DELAY '00:00:10'

select SUM([цена для продажи]) from [Данные\_о\_товарах] where [код группы товара] = 2

commit transaction ins

end try

begin catch

if @@TRANCOUNT > 0 rollback transaction ins

select @errorMessage = ERROR\_MESSAGE(), @errorSeverity = ERROR\_SEVERITY()

raiserror (@errorMessage, @errorSeverity, 1)

end catch

Вторая транзакция:

DECLARE @errorMessage nvarchar(4000), @errorSeverity int

begin transaction ins

begin try

insert into [Данные\_о\_товарах] values ('Лунтик', 2, 2, 'шт' ,1200)

commit transaction

end try

begin catch

if @@TRANCOUNT > 0 rollback

select @errorMessage = ERROR\_MESSAGE(), @errorSeverity = ERROR\_SEVERITY()

raiserror (@errorMessage, @errorSeverity, 1)

end catch

Результат параллельного доступа транзакций (*Рисунок 17*) при режиме изоляции данных:

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

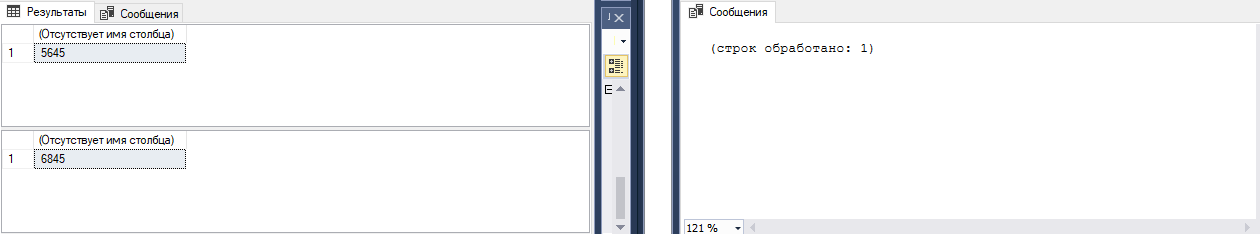


Рисунок 17 - Фантомное чтение

Результат параллельного доступа транзакций (*Рисунок 18*) при режиме изоляции данных:

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

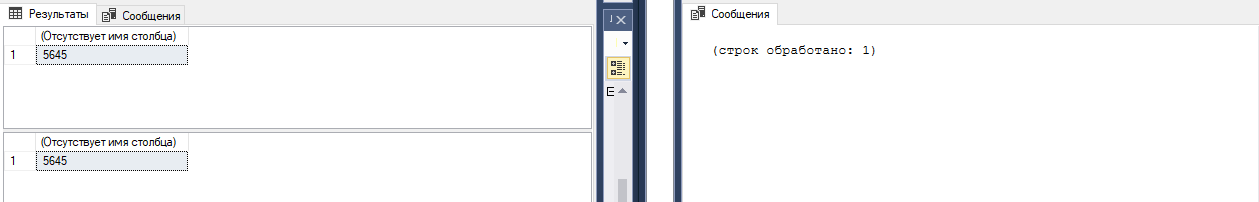


Рисунок 18 - Фантомное чтение