

# Триггеры

- В лабе 3 мы использовали constraint'ы типа check для создания ограничений целостности.
- Они удобны, когда ограничение задаётся в пределах одной таблицы, но если требуемое ограничение связывает несколько таблиц, они абсолютно бесполезны.
- В этом случае нас выручают триггеры. Что это такое?
- Триггер – это особая хранимая процедура, запускаемая автоматически при возникновении на сервере определённого события.
- Рассмотрим триггеры, которые нужно уметь писать для выполнения обязательного и дополнительного заданий.

# Что такое триггер?

Триггеры делятся на следующие типы:

- Триггеры DML, т. е. триггеры, срабатывающие при изменении данных, содержащихся в базе. Они нужны для соблюдения ограничений целостности, затрагивающих более одной таблицы. Именно такие триггеры и просят создать в этой лабе.
- Триггеры DDL, т. е. триггеры, срабатывающие при изменении определений данных. Например, триггер, который при создании таблицы будет создавать где-то запись с именем сделавшего это пользователя.
- Триггеры входа, т. е. триггеры, срабатывающие при попытке входа на сервер. Позволяют вести статистику или, например, совсем запретить вход...

# Как выполнять задание?

- В задании триггер должен исполняться после обновления данных и проверять корректность этого выполнения. В случае обнаружения неправильных данных среди добавленных, изменение должно быть возвращено. Причём надо обрабатывать как вставку данных, так и изменение; это можно сделать с помощью двух триггеров.
- Синтаксис создания триггера:

```
CREATE TRIGGER <имя триггера> ON dbo.<имя таблицы>  
AFTER INSERT /* или UPDATE; или через запятую*/  
AS  
IF <>  
    BEGIN  
  
    END
```

# Некоторые пояснения

- Блок IF – необязательный. Условия вызова может и не быть. Здесь можно написать, например, существование "неправильных" строк.
- Между BEGIN и END находится тело триггера, в котором, собственно, и происходит поиск строк, не удовлетворяющих ограничениям целостности и их удаление (если выполнялся INSERT) или возврат к тому, что было (если выполнен UPDATE).
- Триггер не должен ничего печатать. Даже сообщение об ошибке. Не для этого он предназначен. Его задача – выполнение ограничений целостности. Да, при отладке (триггер – это уже полноценная программа, которую нужно отлаживать) вам понадобится, например, увидеть выборку "что было" и "что стало". Но тов. КБТ именно этого (в том числе) и ждёт, поэтому к сдаче всё должно быть чисто.

# Inserted и deleted

- Таблицы inserted и deleted существуют только внутри триггера и имеют ту же структуру, что и исходная таблица. Что там содержится, ясно из названия. Очевидно, при выполнении INSERT таблица deleted пуста; при выполнении UPDATE изменённые строки исходной таблицы попадают и туда, и туда: в deleted – старая версия, в inserted – новая.
- Обращаться с ними можно как с обычными таблицами, даже если это триггер на представление (на таких остановимся позже).

# Типичные ошибки

Стоимость каждой – 1 балл и go-around!

- Внесение изменений в те строки, которые не имеют отношения к запросу, вызвавшему срабатывание триггера.
- Работа только при добавлении/обновлении одной строки. Я не знаю, как создать эту проблему в полном объёме, вроде у кого-то из моих одноклассников получилось...
- Потеря всех изменений при ошибке только в части из них. Возникает обычно, если мы не предусмотрим такой вариант и на радостях, в спешке совершим глупость (как это часто бывает с людьми...) "ага, надо вернуть всё назад – давайте напишем IF <нарушение> ROLLBACK". На нашем потоке это была одна из наиболее распространённых ошибок.
- Неточный возврат строк в таблицу при некорректном UPDATE. В 99% случаев в таблице есть поле с IDENTITY, поэтому если просто вставить старые строки из таблицы deleted, то они будут изменены, т. к. IDENTITY присвоит новое значение.

# Избегаем ошибок в триггере

- Никакого ROLLBACK. Только DELETE:

DELETE FROM <имя таблицы>

WHERE <условие нарушения>

- При этом в условии ОБЯЗАТЕЛЬНО должно содержаться упоминание того, что удаляемая запись есть в таблице inserted.
- Это касалось триггеров как AFTER INSERT, так и AFTER UPDATE. В первом случае это всё, что нам нужно. Во втором же надо ещё вернуть в таблицу напрасно удалённые строки (ведь по смыслу, UPDATE = DELETE + INSERT). Делается это так:

SET IDENTITY INSERT <имя таблицы> ON

INSERT INTO <имя таблицы> (все поля)

SELECT (все поля)

FROM deleted

WHERE id IN

(SELECT id FROM inserted, ... WHERE <ошибка>)

SET IDENTITY INSERT <имя таблицы> OFF /\*это важно!\*/

# Доп. вопрос от Теймура

- Наш любимый Кирилл Борисович очень любит придумывать новые вопросы. По 9 лабе у него есть коронный, который он задавал всем, начиная с 13 апреля, и которым не один SQL Server был сломан, как любит говорить КБТ, без шансов (например, мой сегодня. Уже переустановил; не пытайтесь повторить, вам ещё 10 лабу сдавать, а переустановка не на каждом компьютере возможна). Как это происходило у меня 27 апреля сего года:

КБТ: напишите триггер, который будет запрещать вход в на сервер по выходным и после 8 вечера.

(написал)

КБТ: стирает все условия, запускает триггер, закрывает окно среды SQL.

КБТ: а теперь залогиньтесь!

- Что же делать?



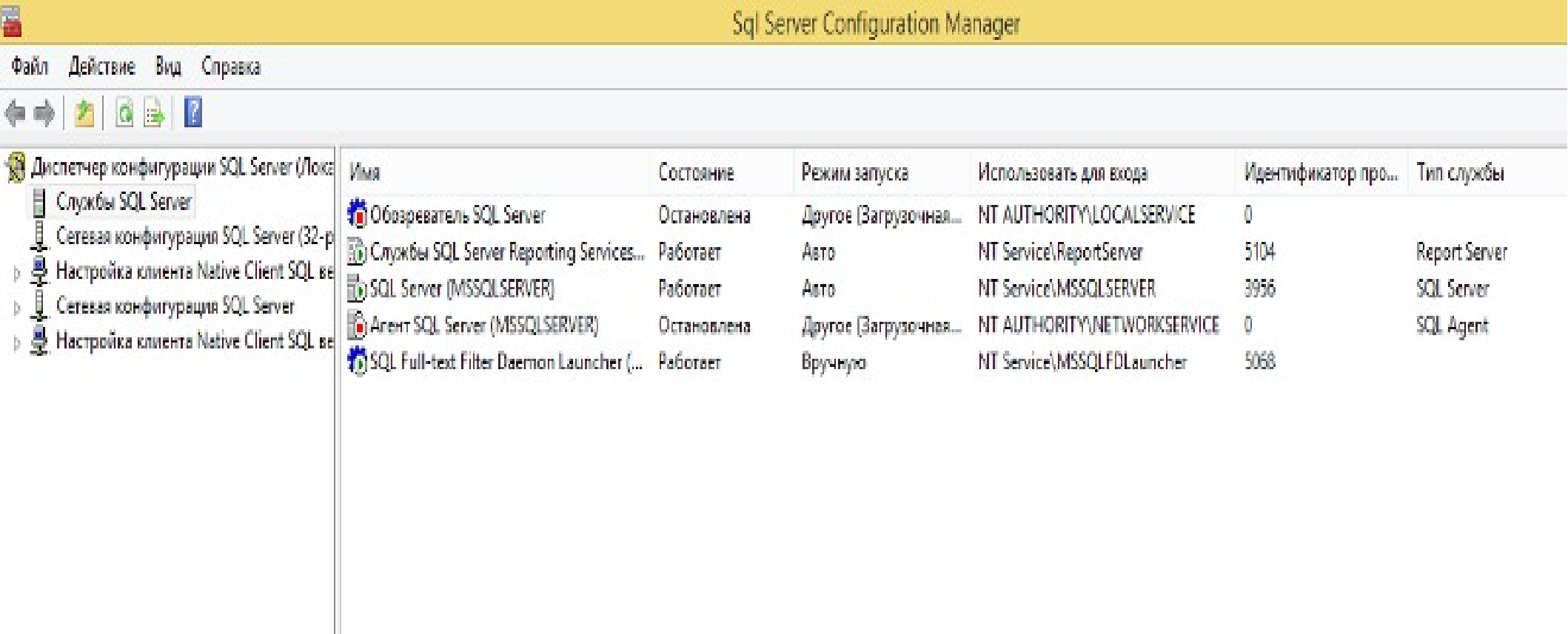
# Ответ: первая часть

```
DROP TRIGGER AntiLogon ON ALL SERVER  
GO  
CREATE TRIGGER AntiLogon  
ON ALL SERVER  
FOR LOGON  
AS  
BEGIN  
    ROLLBACK  
END
```

- Полезный совет: прежде чем проверять, как это работает (и даже запускать триггер без выхода – вдруг ноутбук перезагрузится?), добавьте перед ROLLBACK какое-нибудь необременительное условие, которое через некоторое время нарушится само (типа "системное время не превосходит сейчас + 20 минут", только проверьте его работу в безопасных случаях!)

# Ответ: вторая часть

- Хорошо, ломать сервер научились. Теперь надо научиться его восстанавливать.
- Основная идея (её можно увидеть на MSDN): при запуске с флагом -f триггеры отключены, и войти нам никто не мешает.
- Правда, с таким флагом не получится соединить с сервером и обозреватель объектов, и запрос. Ну что же, удалим триггер через обозреватель.
- Всё, больше он нам жизнь не портит. Осталось перезапустить сервер, не забыв убрать флаг.
- Вот и всё. Теперь всё должно работать!
- Посмотрим на картинки. Нам понадобится ещё одно приложение, оно устанавливается вместе со средой SQL и называется Sql Server Configuration Manager – его можно найти стандартным средством Windows.



- Вот так выглядит окно этой программы. Находим SQL Server, жмём правой кнопкой, выбираем "Свойства"

The screenshot shows the 'Properties: SQL Server (MSSQLSERVER)' dialog box with the 'Startup Parameters' tab selected. The 'Startup Parameters' section has a text box containing '-f' and a 'Добавить' (Add) button. Below it, the 'Existing Parameters' section shows a list of parameters: '-dC:\Program Files\Microsoft SQL Server\MSSQL 12.MSS', '-eC:\Program Files\Microsoft SQL Server\MSSQL 12.MSS', and '-IC:\Program Files\Microsoft SQL Server\MSSQL 12.MSS'. There is a 'Удалить' (Remove) button to the right of the list. At the bottom of the dialog are buttons for 'OK', 'Отмена' (Cancel), 'Применить' (Apply), and 'Справка' (Help).

Вход Служба FILESTREAM Высокий уровень доступности AlwaysOn

Параметры запуска Дополнительно

Укажите параметр запуска:

-f

Добавить

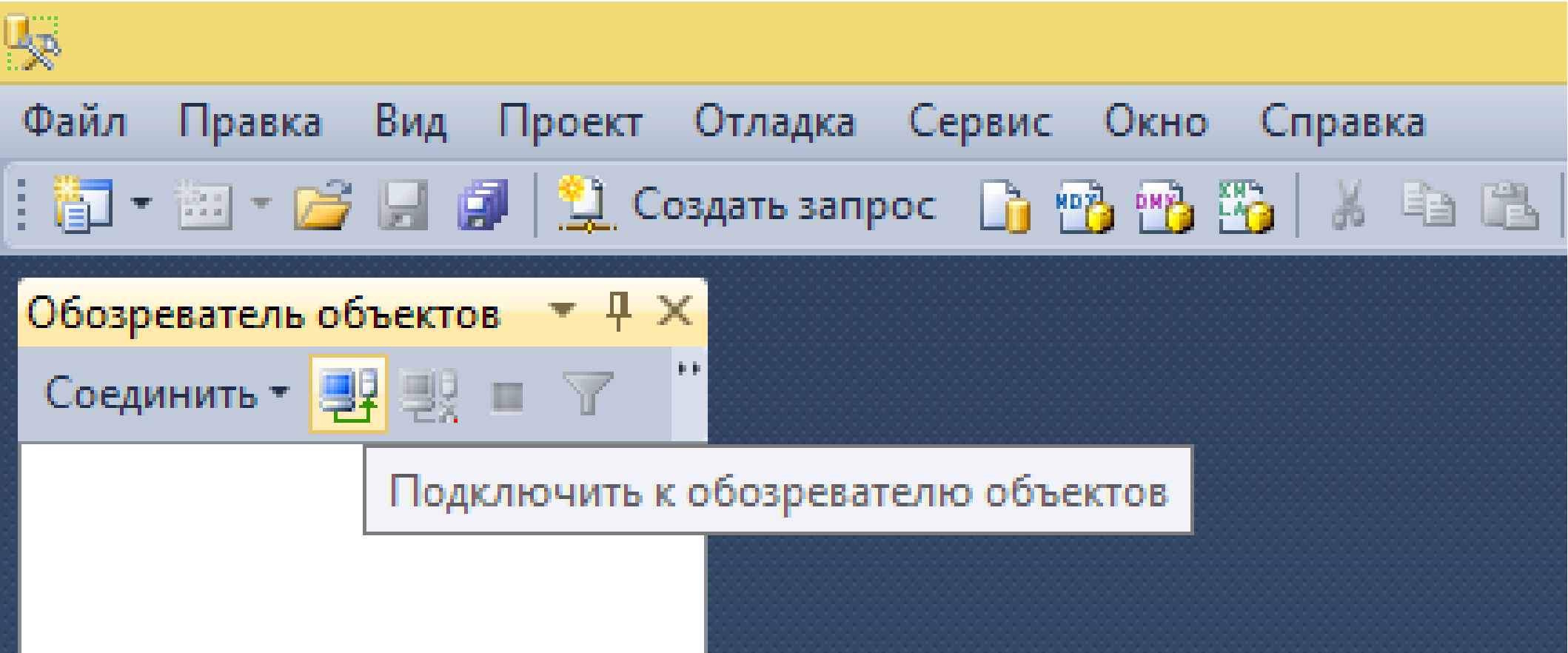
Существующие параметры:

-dC:\Program Files\Microsoft SQL Server\MSSQL 12.MSS  
-eC:\Program Files\Microsoft SQL Server\MSSQL 12.MSS  
-IC:\Program Files\Microsoft SQL Server\MSSQL 12.MSS

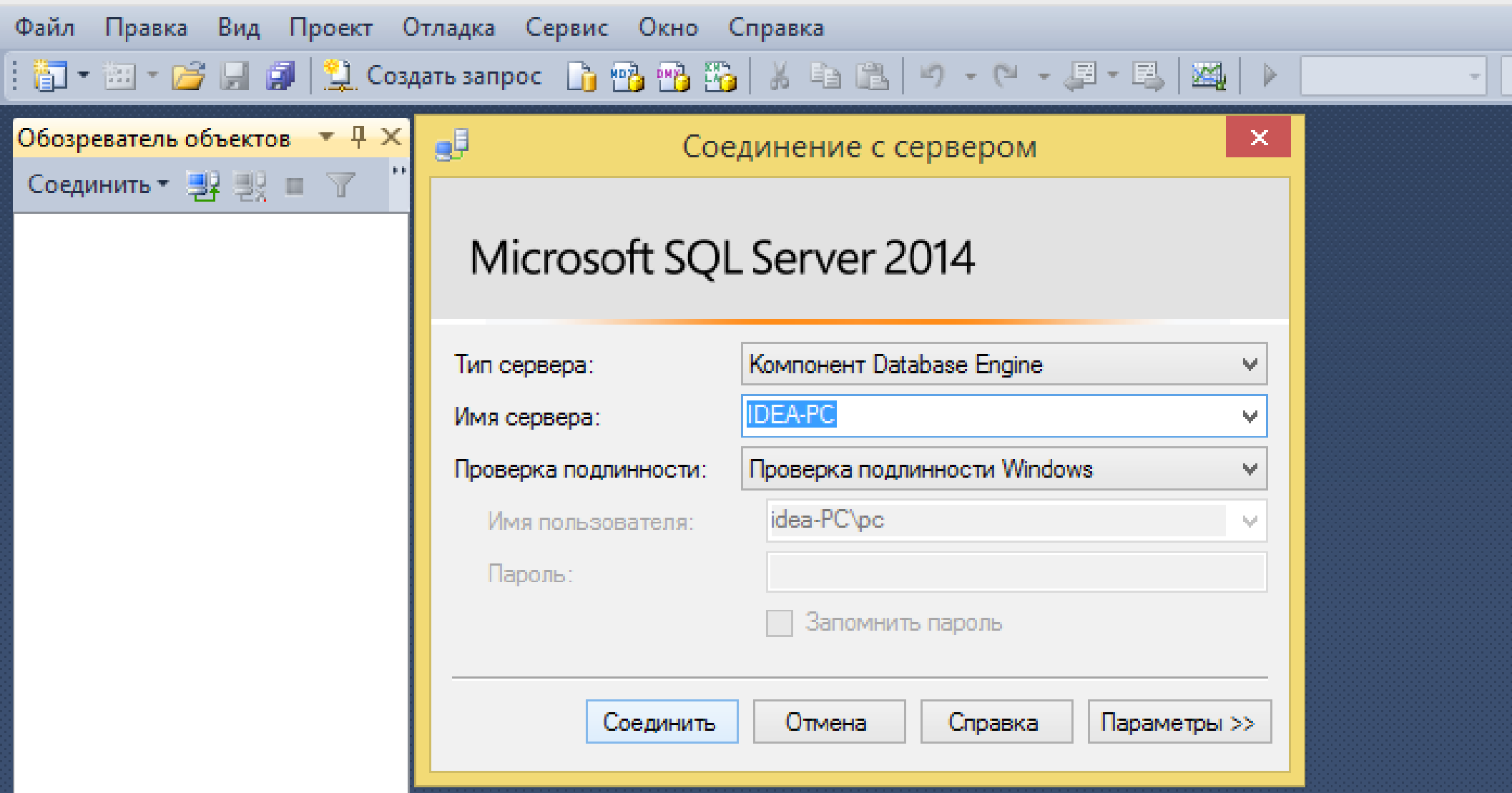
Удалить

OK Отмена Применить Справка

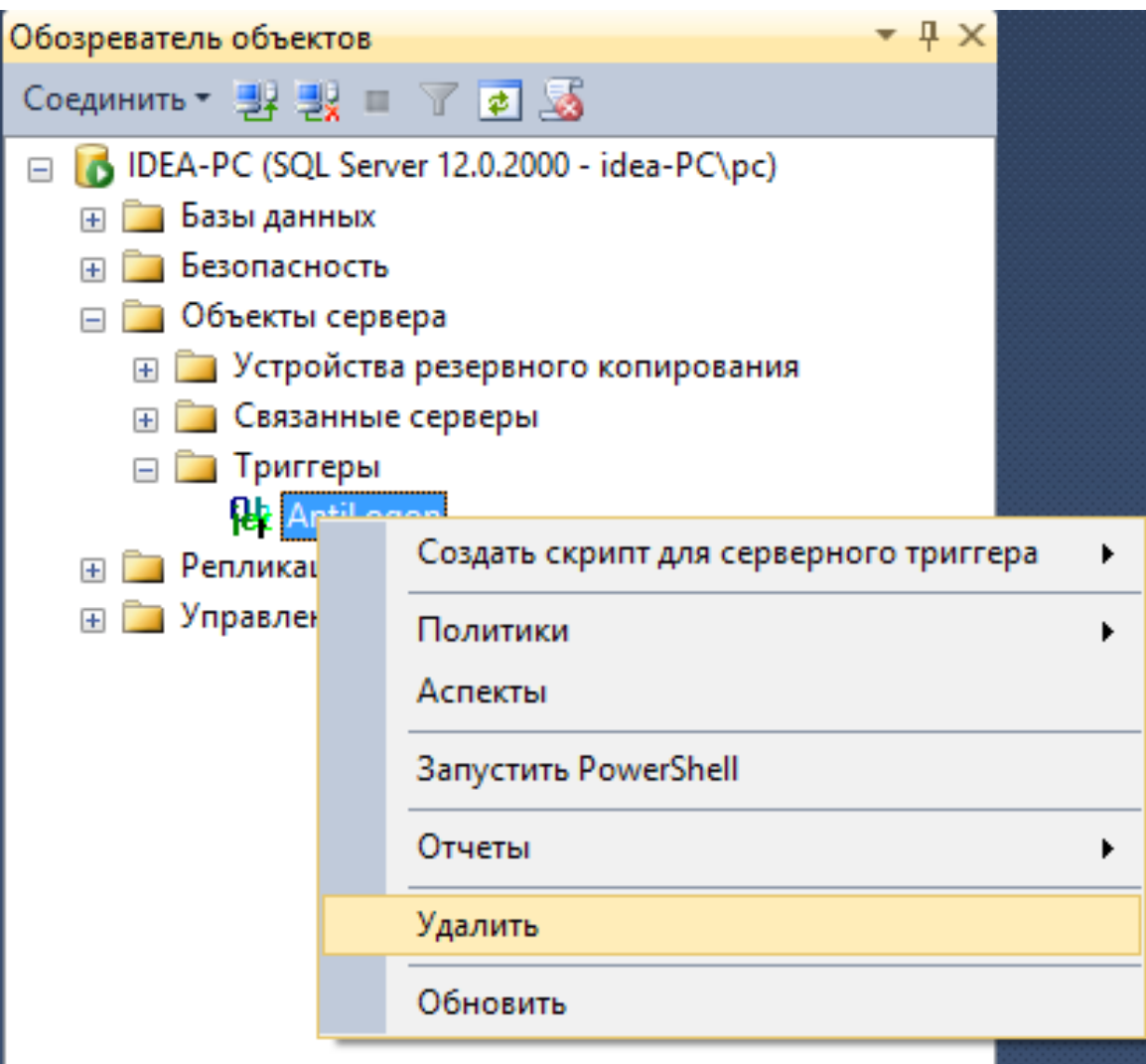
- В свойствах выбираем вкладку "Параметры запуска", указываем -f, жмём "Добавить" и "OK".
- Перезапускаем данную службу (правая кнопка – "Перезапустить")
- Запускаем SQL Management Studio (это то, в чём мы всегда работали). На всякий случай отключаем все предложения залогиниться – режим однопользовательский.



- Нажимаем на значок "Подключить к обозревателю объектов"



- Получаем обычное наше окно. Нажимаем "Соединить", и – о чудо! – обозреватель объектов смог.



- Выбираем вкладку "Объекты сервера", потом "Триггеры". Скорее всего (если мы ничего больше не создавали) наш будет единственным.
- Удаляем. Вылезает какое-то окошко, в нём надо просто согласиться.
- Возвращаемся к нашему Sql Server Configuration Manager

## Свойства: SQL Server (MSSQLSERVER)

Вход Служба FILESTREAM Высокий уровень доступности AlwaysOn  
Параметры запуска Дополнительно

Укажите параметр запуска:

Добавить

Существующие параметры:

-dC:\Program Files\Microsoft SQL Server\MSSQL 12.MSS  
-eC:\Program Files\Microsoft SQL Server\MSSQL 12.MSS  
-fC:\Program Files\Microsoft SQL Server\MSSQL 12.MSS  
-f

Удалить

ОК

Отмена

Применить

Справка

- Снова открываем свойства SQL Server. Находим параметр -f, удаляем его.
- Опять соглашаемся с предупреждением; перезапускаем службу.
- Запускаем SQL, и ничто не напоминает нам о былом кошмаре.
- Кроме, конечно, фото на память...



Подключиться к компоненту Database Engine

## Microsoft SQL Server 2014

Тип сервера: Компонент Database Engine

Имя сервера: IDEA-PC

Проверка подлинности: Проверка подлинности Windows


Имя пользователя: idea-PC\pc

Пароль:

☐ Запомнить пароль




Соединить Отмена Справка Параметры >>

Подключиться к компоненту Database Engine

 Невозможно подключиться к IDEA-PC.

**Дополнительные сведения:**

→ Не удалось выполнить вход для имени входа "idea-PC\pc" из-за выполнения триггера .  
Контекст базы данных изменен на "master".  
Параметры языка изменены на "русский". (Microsoft SQL Server, ошибка: 17892)

OK

# Зачем ещё триггеры?

- Триггеры нужны не только для того, чтобы соблюдать сложные ограничения целостности (их зачастую называют бизнес-правилами) или мучить бедных детей.
- Помните, я упоминал вслед за авторами статьи на MSDN про триггеры в контексте обновления представлений?
- Там идеология простая: вместо AFTER пишем INSTEAD OF. Таблицы inserted и deleted создаются автоматически, как если бы это был обычный запрос к обычной таблице.
- В теле триггера просто расписываем обновление представления как обновление базовых таблиц.
- Если база данных была спроектирована правильно, и все строки уникальны, то сложностей быть не должно.
- Заметим, что это всё никак не зависит от того, было ли представление обновляемым в обычном смысле!

# Спасибо за внимание!

Надеюсь, вам не составит труда найти ответы на вопросы из документа на MSDN.