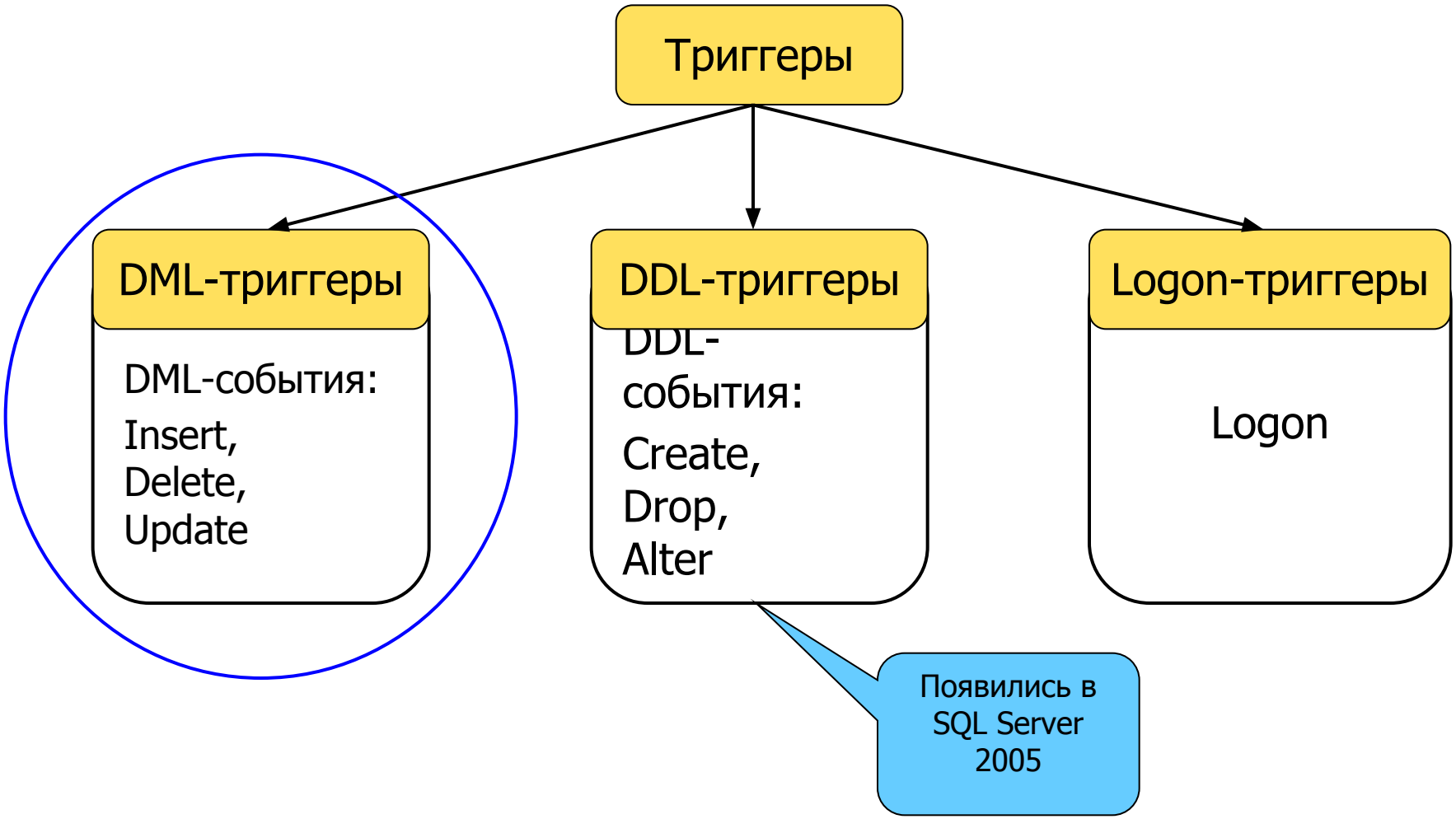
Триггеры в MS SQL Server

Что такое триггер

- ***Триггер*** – это откомпилированная SQL-процедура
- Исполнение обусловлено наступлением определенных событий внутри реляционной базы данных
- Не имеет параметров
- Становится «одним целым» с вызвавшей операцией

Виды триггеров



Назначение триггеров

- Проверка корректности введенных данных и выполнение сложных *ограничений целостности данных*, которые трудно, если вообще возможно, поддерживать с помощью ограничений целостности, установленных для таблицы.
- Накопление аудиторской информации посредством фиксации сведений о внесенных изменениях и тех лицах, которые их выполнили.
- Автоматического оповещения других модулей о том, что делать в случае изменения информации содержащейся в таблице БД.
- Для реализации так называемых "бизнес правил".
- Для организации каскадных воздействий на таблицы БД (могут действовать не только при равенстве значений).
- Поддержка репликации.

Когда нужны триггеры

- Чтобы оценить состояние таблицы до и после изменения данных и предпринять действия на основе этого различия.
- Для предотвращения действий, нарушающих бизнес-логику приложения
- Несколько DML-триггеров одинакового типа (INSERT, UPDATE или DELETE) для таблицы позволяют предпринять несколько различных действий в ответ на одну инструкцию изменения данных.

DML-триггеры

- Триггер создается по одной таблице базы данных
- Может осуществлять доступ и к другим таблицам и объектам других баз данных.
- Триггеры нельзя создать по временным таблицам или системным таблицам, а только по определенным пользователем таблицам или представлениям.
- Таблица, по которой определяется триггер, называется *таблицей триггера.*

DML - trigger

- Объект - таблица, VIEW
- Событие - insert, update, delete для таблицы и для VIEW.
- Время активации – до (вместо) или после выполнения оператора.

DML-триггеры

- Триггер – блок, выполняемый автоматически каждый раз, когда происходит определенное событие
  - в отличие от процедуры, которая должна быть вызвана явно
- Событие – INSERT, UPDATE и DELETE для таблицы, представления
  - для запроса нельзя определить триггер

Когда нужны триггеры

- Для каскадных изменений в связанных таблицах БД (если их нельзя выполнить при помощи каскадных ограничений ссылочной целостности).
- Для предотвращения случайных или неправильных операций INSERT, UPDATE и DELETE
- Для реализации ограничений целостности, которые нельзя определить при помощи ограничения CHECK. DML-триггеры могут ссылаться на столбцы других таблиц.

Еще…

- Журнализация и аудит. С помощью триггеров можно отслеживать изменения таблиц, для которых требуется поддержка повышенного уровня безопасности. Данные об изменении таблиц могут сохраняться в других таблицах и включать, например, идентификатор пользователя, время операции обновления; сами обновляемые данные и т. д.
- Согласование и очистка данных. С любым простым оператором SQL, обновляющим некоторую таблицу, можно связать триггеры, производящие соответствующие обновления других таблиц.
- Операции, не связанные с изменением базы данных. В триггерах могут выполняться не только операции обновления базы данных. Стандарт SQL позволяет определять хранимые процедуры (которые могут вызываться из триггеров), посылающие электронную почту, печатающие документы и т. д.

Когда не надо использовать триггеры

- Не нужно реализовывать триггерами возможности, достигаемые использованием декларативных средств СУБД (ограничения целостности или внешние ключи)
- Избегайте сложных цепочек триггеров

Советы

- Не используйте триггеры, если можно применить *проверочное ограничение* CHECK
- Не используйте ограничение CHECK, если можно обойтись ограничением UNIQUE.

Основные параметры триггера

- Имя триггера
- Имя таблицы (или представления)
- Время срабатывания:
  AFTER(FOR) или INSTEAD OF
- Событие: INSERT, UPDATE, DELETE (TRUNCATE TABLE – это не удаление !)
- Тело триггера

! Последовательность срабатывания однотипных триггеров произвольна

Группировка событий

- Например, вы можете создать триггер, который будет активизироваться, когда происходит выполнение оператора UPDATE или INSERT, и такой триггер мы будем называть триггером UPDATE/INSERT. Вы можете даже создать триггер, который будет активизироваться при возникновении любого из трех событий модификации данных (триггер UPDATE/INSERT/DELETE).

Правила работы триггера

- Триггеры запускаются после завершения оператора, который вызвал их активизацию. Например, UPDATE-триггер не будет активизироваться, пока не будет выполнен оператор UPDATE.
- Если какой-либо оператор пытается выполнить операцию, которая нарушает какое-либо ограничение по таблице или является причиной какой-то другой ошибки, то связанный с ним триггер не будет активизирован.

Правила работы триггера

- Триггер рассматривается как часть одной транзакции вместе с оператором, который вызывает его. Поэтому из триггера можно вызвать оператор отката, и этот оператор выполнит откат как триггера, так и соответствующего события модификации данных.
- При возникновении ошибки при выполнении триггера автоматически выполняется откат всей транзакции.
- Триггер активизируется только один раз для одного оператора, даже если этот оператор влияет на несколько строк данных.

Пример

```
CREATE TRIGGER trg ON my_table
FOR INSERT, UPDATE, DELETE AS
select "this is trigger"
```

Рекурсия

- Косвенная рекурсия

  При косвенной рекурсии приложение обновляет таблицу T1. Это событие вызывает срабатывание триггера TR1, обновляющего таблицу T2. Это вызывает срабатывание триггера T2 и обновление таблицы T1.
- Прямая рекурсия

  При прямой рекурсии приложение обновляет таблицу T1. Это событие вызывает срабатывание триггера TR1, обновляющего таблицу T1. Поскольку таблица T1 уже была обновлена, триггер TR1 срабатывает снова и т. д.

- При вызове триггера будут выполнены операторы SQL, указанные после ключевого слова AS. Вы можете поместить сюда несколько операторов, включая программные конструкции, такие как IF и WHILE.

Выбор типа триггера

- Триггеры INSTEAD OF используются для:
  - Выборочного запрещения исполнения команды, для которой определен триггер (проверки пред-условия);
  - Подсчета значений столбцов до завершения команды INSERT или UPDATE.
- Триггеры AFTER используются для:
  - Учета выполненных операций;
  - Проверки пост-условий исполнения команды.

Циклы и вложенность

- SQL Server позволяет использовать вложенные триггеры, до 32 уровней вложенности. Если любой из вложенных триггеров выполняет операцию ROLLBACK, то последующие триггеры не запускаются.
- Запуск триггеров отменяется, если формируется бесконечный цикл.

Триггер INSTEAD OF

- Триггер INSTEAD OF выполняется вместо запуска оператора SQL. Тем самым переопределяется действие запускающего оператора.
- Можно задать по одному триггеру INSTEAD OF на один оператор INSERT, UPDATE или DELETE.
- Триггер INSTEAD OF можно задать для таблицы и/или представления
- Можно использовать каскады триггеров INSTEAD OF, определяя представления поверх представлений, где каждое представление имеет отдельный триггер INSTEAD OF.
- Триггеры INSTEAD OF не разрешается применять для модифицируемых представлений, содержащих опцию WITH CHECK.

Триггер AFTER

- Триггеры AFTER могут быть определены только в таблицах.
- Триггер AFTER активизируется после успешного выполнения всех операций, указанных в запускающем операторе или операторах SQL. Сюда включается весь каскад действий по ссылкам и все проверки ограничений.

Триггер AFTER

- Если у вас имеется несколько триггеров AFTER, определенных по таблице для определенного оператора или набора операторов, то вы можете задать, какой триггер будет активизирован первым и какой триггер – последним.
- Если у вас определено больше двух триггеров, то вы можете задать порядок активизации только первого и последнего триггера. Все остальные триггеры активизируются случайным образом.

Порядок AFTER-триггеров

- sp_settriggerorder @triggername = 'AnotherTrigger', @order = 'first'
- sp_settriggerorder @triggername = 'MyTrigger', @order = 'last'
- sp_settriggerorder @triggername = 'MyOtherTrigger', @order = 'none'
- sp_settriggerorder @triggername = 'YetAnotherTrigger', @order = 'none'

Использование таблиц deleted и inserted

- При создании триггера вы имеете доступ к двум временным таблицам с именами deleted и inserted. Они хранятся в памяти, а не на диске.
- Эти две таблицы имеют одинаковую структуру с таблицей (одинаковые колонки и типы данных), по которой определяется данный триггер.

Использование inserted, deleted

Специальные таблицы:

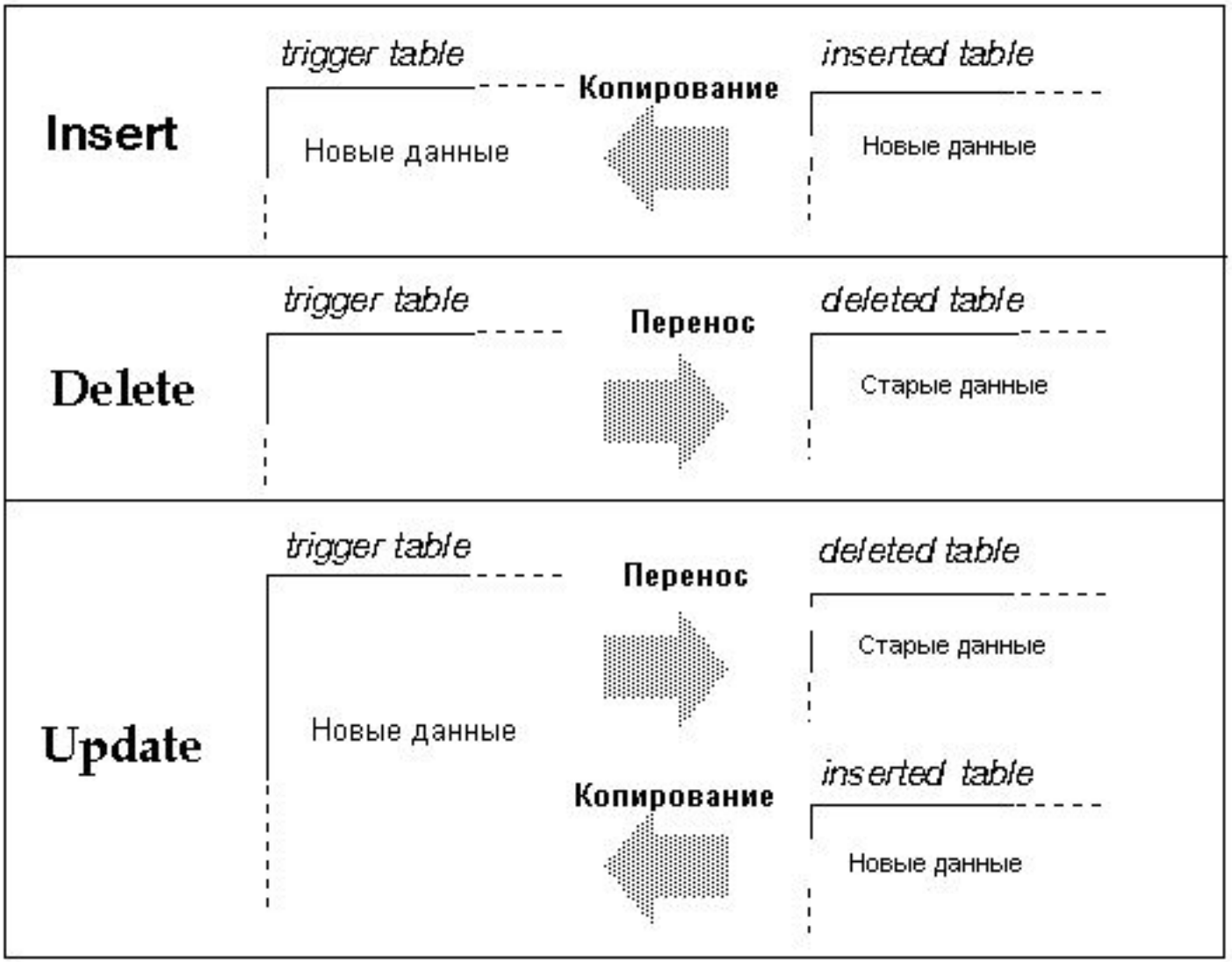
- inserted – вставленные значения (для INSERT, UPDATE)
- deleted – удаленные значения (для UPDATE, DELETE)

Использование таблиц deleted и inserted

- Таблица deleted содержит копии строк, на которые повлиял оператор DELETE или UPDATE. Строки, удаляемые из таблицы данного триггера, перемещаются в таблицу deleted. После этого к данным таблицы deleted можно осуществлять доступ из данного триггера.
- Таблица inserted содержит копии строк, добавленных к таблице данного триггера при выполнении оператора INSERT или UPDATE. Эти строки добавляются одновременно в таблицу триггера и в таблицу inserted.

Использование таблиц deleted и inserted

- Поскольку оператор UPDATE обрабатывается как DELETE, после которого следует INSERT, то при использовании оператора UPDATE старые значения строк копируются в таблицу deleted, а новые значения строк – в таблицу триггера и в таблицу inserted.
- Триггер INSERT => deleted пуст
- Триггер DELETE => inserted пуст
- но сообщение об ошибке не возникнет !


Insert
trigger table
Новые данные
Копирование
inserted table
Новые данные
Delete
trigger table
Перенос
deleted table
Старые данные
Update
trigger table
Новые данные
Перенос
deleted table
Старые данные
Копирование
inserted table
Новые данные

Создание триггера

```
CREATE TRIGGER [ schema_name.]trigger_name ON
    { table | view }
    { FOR | AFTER | INSTEAD OF }
    { [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
    AS { sql_statement}
```

```
CREATE TRIGGER plus_1
ON table1
instead of insert
AS
insert table1 (id, col1)  select id+1, col1
from inserted;
```

- *Триггер* представляет собой специальный тип хранимых процедур, запускаемых сервером автоматически при попытке изменения данных в таблицах, с которыми *триггеры* связаны. Каждый *триггер* привязывается к конкретной таблице. Все производимые им модификации данных рассматриваются как одна *транзакция*. В случае обнаружения ошибки или нарушения целостности данных происходит *откат* этой транзакции. Тем самым внесение изменений запрещается. Отменяются также все изменения, уже сделанные *триггером*.

Обработка исключений

- Команда **ROLLBACK** указывает серверу остановить обработку модификации и запретить транзакцию.
- Существует также команда **RAISEERROR,** с помощью которой вы можете отправить сообщение об ошибке пользователю.
- TRY…CATCH

Обработка исключений сообщение об ошибке

```
RAISERROR ('Error raised because of wrong data.', -- Message text.
           16, -- Severity.
           1 -- State.);
```

Severity – число от 0 до 25

Определенный пользователем уровень серьезности ошибки.
0 до 18 может указать любой пользователь.
19 до 25 могут быть указаны только sysadmin
20 до 25 считаются неустранимыми - соединение с клиентом обрывается и регистрируется сообщение об ошибке в журналах приложений и ошибок.

State Целое число от 0 до 255. Отрицательные значения или значения больше 255 приводят к формированию ошибки. Если одна и та же пользовательская ошибка возникает в нескольких местах, то при помощи уникального номера состояния для каждого местоположения можно определить, в каком месте кода появилась ошибка.

Функции об ошибках

- Функция ERROR_LINE() возвращает номер строки, в которой произошла ошибка.
- Функция ERROR_MESSAGE() возвращает текст сообщения, которое будет возвращено приложению. Текст содержит значения таких подставляемых параметров, как длина, имена объектов или время.
- ERROR_NUMBER() возвращает номер ошибки.
- Функция ERROR_PROCEDURE() возвращает имя хранимой процедуры или триггера, в котором произошла ошибка. Эта функция возвращает значение NULL, если данная ошибка не была совершена внутри хранимой процедуры или триггера.
- ERROR_SEVERITY() возвращает уровень серьезности ошибки.
- ERROR_STATE() возвращает состояние.

Пример триггера

```
CREATE TRIGGER LowCredit ON
Purchasing.PurchaseOrderHeader AFTER INSERT AS
BEGIN
DECLARE @creditrating tinyint, @vendorid int ;

SELECT @creditrating = v.CreditRating, @vendorid = p.VendorID
FROM Purchasing.PurchaseOrderHeader p
     JOIN inserted i ON p.PurchaseOrderID = i.PurchaseOrderID JOIN
     Purchasing.Vendor v ON v.VendorID = i.VendorID ;

IF @creditrating = 5
     RAISERROR ('This vendor''s credit rating is too low to accept new purchase
     orders.', 16, 1) ;

END
```

Управление триггерами

- Отключение/включение триггера:
  - DISABLE/ENABLE TRIGGER trigger_name ON object_name
- Отключение/включение всех триггеров таблицы:
  - DISABLE/ENABLE TRIGGER ALL ON object_name
- Изменение триггера:
  - ALTER TRIGGER trigger_name …
- Удаление триггера:
  - DROP TRIGGER trigger_name

Изменение триггера

```
ALTER TRIGGER tr_name
ON on_board
 after   UPDATE
AS
update on_board set iks='b' where id in (select
   id from inserted)
```

Удаление триггера

- DROP TRIGGER tr_name

Активация/деактивация триггера

- DISABLE TRIGGER {trigger_name [,...n] | ALL } ON { object_name} ;
- ENABLE TRIGGER {trigger_name [,...n] | ALL } ON { object_name}

Применение триггеров

- Защита
  - Запрещение доступа в зависимости от значений данных
- Учет
  - Ведение журналов изменений
- Целостность данных
  - Сложные правила целостности
  - Сложная ссылочная целостность
- Производные данные
  - автоматическое вычисление значений

Типы триггеров

Функция	**Триггер AFTER**	**Триггер INSTEAD OF**
Сущности	Таблицы	Таблицы и представления
Количество триггеров на таблицу/представление	Несколько на одно событие	Один триггер на одно событие
Каскадные ссылки	Нет ограничений	INSTEAD OF UPDATE и DELETE нельзя определять для таблиц, на которые распространяются каскадные ограничения ссылочной целостности.
Выполнение	После следующих операций: • Обработка ограничений. • Декларативные ссылочные действия. • Создание таблиц **inserted** и **deleted**. • Действие, запускающее триггер.	Перед следующей операцией: • Обработка ограничений. Вместо следующей операции: • Действие, запускающее триггер. После следующих операций: • Создание таблиц **inserted** и **deleted**.

DDL - trigger

- Триггеры DDL могут быть использованы в административных задачах, таких как аудит и регулирование операций базы данных.
- Действие этих триггеров распространяется на все команды одного типа во всей базе данных или на всем сервере.

DDL - триггеры

- Триггеры DDL, как и обычные триггеры, вызывают срабатывание хранимых процедур в ответ на событие.
- Срабатывают в ответ на разнообразные события языка определения данных (DDL).
- Эти события в основном соответствуют инструкциям языка Transact-SQL, начинающимся ключевыми словами CREATE, ALTER или DROP.

Задачи для DDL - триггеров

- Предотвратить внесение определенных изменений в схему базы данных.
- Выполнить в базе данных некоторые действия в ответ на изменения в схеме базы данных.
- Записывать изменения или события схемы базы данных.
- Триггеры DDL срабатывают только после выполнения соответствующих инструкций DDL. Триггеры DDL нельзя использовать в качестве триггеров INSTEAD OF.

```
CREATE TRIGGER trigger_name
ON { DATABASE | ALL SERVER }
{ FOR | AFTER } { event_type | event_group }
AS
{ sql_statement  [ ; ] [ ,...n ]  [ ; ] }
```

Создание/удаление DDL-тр

```
CREATE TRIGGER ddl_trig_database
ON ALL SERVER
FOR CREATE_DATABASE
AS
   PRINT 'Database Created.'


DROP TRIGGER ddl_trig_database
ON ALL SERVER;
```

DDL - trigger

```
CREATE TRIGGER safety
   ON DATABASE
   FOR DROP_TABLE, ALTER_TABLE AS
   PRINT 'You must disable Trigger "safety" to
   drop or alter tables!'
   ROLLBACK ;
```

- Для одной инструкции Transact-SQL можно создать несколько триггеров DDL.
- Триггер DDL и инструкция, приводящая к его срабатыванию, выполняются в одной транзакции.
- Откат событий ALTER DATABASE, возникших внутри триггера DDL, невозможен.
- Триггеры DDL выполняются только после завершения инструкции Transact-SQL. Триггеры DDL нельзя использовать в качестве триггеров INSTEAD OF.
- Триггеры DDL не создают таблицы inserted и deleted.

Logon - trigger

- Триггеры входа выполняют хранимые процедуры в ответ на событие LOGON. Это событие вызывается при установке пользовательского сеанса с экземпляром SQL Server.
- Триггеры входа срабатывают после завершения этапа проверки подлинности при входе, но перед тем, как пользовательский сеанс реально устанавливается.

Logon - trigger

```
CREATE TRIGGER trigger_name
ON ALL SERVER
{ FOR| AFTER } LOGON
AS { sql_statement }
```