**Триггеры**

**Триггер** – это хранимая процедура, выполняющаяся автоматически в ответ на действия пользователя

Виды триггеров:

* DML – триггеры уровня БД, срабатывают при модификации данных (INSERT UPDATE DELETE)
* DDL – триггеры, это триггеры уровня БД и сервера (CREATE ALTER DROP)

**DML триггеры**

**Область применения:**

* Обеспечение целостности данных
* Вывод предупреждений при модификации
* Логирование действий пользователя
* Поддержка синхронизации копий БД

**Особенности:**

* Выполняются **медленнее** чем проверки **CHECK**
* Могут выполняться рекурсивно
* Вызываются один раз для всей таблицы, а не для отдельных строк
* Всегда выполняются внутри внешней транзакции

**Системные таблицы, использующиеся в DML триггеров:**

MSSQL / MySQL

inserted / new (INSERT, UPDATE)

Заполняются новыми версиями строк

Deleted / old (DELETE, UPDATE)

Заполняются старыми версиями строк

Эти таблицы создаются и используются только в триггерах, заполняются автоматически при попытке модифицировать данные. Структура таблиц совпадает со структурой изменяемой таблицы

**Виды DML триггеров:**

* INSTEAD OF – заменяет собой команду которая вызвала триггер
* AFTER – выполняется после команды которая вызвала триггер

**Порядок операций при выполнении DML команды:**

1. Заполнение системных таблиц inserted/deleted
2. Срабатывание INSTEAD OF триггера, если он есть, позволяет проверить данные на корректность, сохранить предыдущие версии строк
3. Проверка ограничений целостности CHECK
4. Выполнение DML команды
5. Срабатывание AFTER триггера, если он есть для ведения лога и изменения данных связанных таблиц (SELECT ORIGINAL\_LOGIN() – возвращает текущего пользователя)

**Общая форма:**

CREATE TRIGGER ИмяТриггера

ON ИмяТаблицы

INSTEAD OF / AFTER DELETE, INSERT, UPDATE

AS

BEGIN

--действия

END

**Примеры:**

1. Вывод количества изменённых строк

CREATE TRIGGER trChangedGamesCount

ON Game

AFTER DELETE, INSERT, UPDATE

AS

BEGIN

PRINT CONCAT\_WS(' ', '~Изменено строк:', @@ROWCOUNT)

END;

1. Сохранение истории изменения цен
2. Создать таблицу для хранения старых цен, столбцы GameId, Price, ChangingDate – DEFAULT GetDate(); GameId, ChangingDate – PK

CREATE TRIGGER trSavePrice

ON Game

AFTER UPDATE

AS

BEGIN

IF UPDATE(Price)

INSERT INTO GamePrice(GameId, Price)

SELECT GameId, Price

FROM deleted;

END;

1. Запись удалённых строк в таблицу

Создать таблицу DeleteCategory – CategoryId, Name, DeletedDate, Login; CategoryId – PK

CREATE TRIGGER trSaveCategory

ON Category

AFTER DELETE

AS

BEGIN

INSERT INTO DeletedCategory(CategoryId, Name, DeletedDate, Login)

SELECT CategoryId, Name, GetDate(), ORIGINAL\_LOGIN()

FROM deleted;

END;

1. Создание триггера, заменяющего удаление игры на пометку строки как удалённой

Изменяем таблицу Game, добавив столбец из Deleted типа bit, IsDeleted, значение по умолке 0.

Создание триггера

CREATE TRIGGER trDeleteGame

ON Game

INSTEAD OF DELETE

AS

UPDATE Game

SET IsDeleted = 1

WHERE GameId IN (SELECT GameId FROM deleted); -- получение идентификатора удалённой игры

1. Изменение записи в связанных таблицах

Требуется при выполнении покупки уменьшать кол-во ключей из таблицы Game TotalKeys

Создали таблицу Sale

CREATE TABLE [dbo].[Sale](

[SaleId] [int] IDENTITY(1,1) NOT NULL,

[GameId] [int] NOT NULL,

[SaledKeys] [int] NOT NULL,

PRIMARY KEY CLUSTERED

(

[SaleId] ASC

)) ON [PRIMARY]

Добавили в Game столбец TotalKeys INT, значение по умолчанию: 100

**1 ВАРИАНТ** – оформление продажи без учета имеющегося количества ключей

CREATE TRIGGER TrChangeKeysCount

ON Sale

AFTER INSERT

AS

UPDATE Game

SET TotalKeys -= SaledKeys

FROM Game JOIN inserted ON Game.GameId = inserted.GameId

**2 ВАРИАНТ** – добавление проверки имеющегося кол-ва ключей

CREATE TRIGGER [dbo].[TrChangeKeysCountWithCheck]

ON [dbo].[Sale]

INSTEAD OF INSERT

AS

BEGIN

DECLARE @saledKeys INT;

DECLARE @totalKeys INT;

SELECT @saledKeys = SaledKeys, @totalKeys = TotalKeys

FROM Game JOIN inserted ON Game.GameId = inserted.GameId;

IF @totalKeys >= @saledKeys

BEGIN

INSERT INTO Sale(GameId, SaledKeys)

SELECT GameId, SaledKeys

FROM Sale;

UPDATE Game

SET TotalKeys -= SaledKeys

FROM Game JOIN inserted ON Game.GameId = inserted.GameId;

END;

ELSE

RAISERROR('%s, недостаточно ключей для покупки %d ', 18, 1, 'user', @totalKeys);

END;

**RAISERROR** – позволяет выводить сообщение об ошибке

**Параметры:**

1. Сообщение/код стандартного сообщения SELECT \* FROM sys.messages можно указывать параметры, они пишутся после третьего
2. Уровень серьезности/severity – число от 1 до 25.

1-10 – вывод в консоль.

11-18 – вывод в окно сообщения

19-25 – сообщение записывается в лог

20-25 – связь с сервером прерывается

1. Приоритет обработки