Бази от данни

Упражнение 12:

Тригери

Димитър Димитров

Предупреждение

- За днешния материал е необходимо трите примерни БД да имат първоначалните ограничения (PK, FK, …)
- Изпълнете оригиналните скриптове за създаване на базите

Дефиниция

- Процедурен код, който се изпълнява автоматично при настъпването на дадено събитие
- Ще разгледаме DML тригери – дефинират се за:
  - конкретна таблица или изглед и
  - конкретна операция/и – INSERT, UPDATE и/или DELETE

Някои приложения

- Проследяване на промените, извършени върху дадени таблици
- Много по-сложни ограничения
  - Например: актьор не може да играе във филм, преди да се е родил (или преди да е навършил определена възраст)
  - В CHECK не можехме да използваме подзаявки
- Отменяне на операции
- Позволяване на INSERT/UPDATE/DELETE върху непроменяем изглед
- И много други

Синтаксис

```
CREATE TRIGGER <name>
ON <table_or_view>
{AFTER|INSTEAD OF} <DML_operations>
AS <body>;
```

- **<DML_operations>** е INSERT / UPDATE / DELETE или комбинация от някои от тях
- Изтриване:
  DROP TRIGGER <name>;

Кога се изпълняват?

- Някой изпълнява DML операция върху дадена таблица
  - Тригер не се извиква по име
- AFTER – след успешно изпълнение на DML операцията
- INSTEAD OF – вместо операцията
- В други СУБД има и BEFORE

AFTER тригери

- Изпълняват се след успешно завършване на INSERT/UPDATE/DELETE
  - Ако заявката нарушава ограничение, не се стига до изпълнение на AFTER тригера
- Може да имаме много AFTER тригери върху една и съща комбинация от таблица и операция
  - Допълнителен материал: можем да укажем кой да бъде изпълнен първи и кой – последен
- Само върху таблици

INSTEAD OF тригери

- DML операцията, която е предизвикала изпълнението на тригера, не се изпълнява
  - Не се проверяват ограничения за нея
- Най-много един тригер
- Както за таблици, така и за изгледи
- С тях можем да реализираме BEFORE тригери

Ред на изпълнение на операциите

- Извиква се операция INSERT/UPDATE/DELETE
- Ако е дефиниран INSTEAD OF тригер, той се изпълнява вместо операцията
- Ако модификациите нарушават ограничения, операцията се прекратява
- Ако са дефинирани AFTER тригери, те се изпълняват
- Транзакцията се затвърждава (COMMIT)
  - т.е. дори AFTER тригер може да анулира успешно изпълнена DML операция с ROLLBACK TRANSACTION

Тяло на тригера

- Процедурен код, който може да съдържа:
  - INSERT, UPDATE и DELETE заявки
  - BEGIN … END блокове
  - IF … [ELSE]
  - Деклариране на променливи, цикли и др.

- Как тригерът знае какви промени по данните са били извършени?
- MS SQL: временните таблици INSERTED и DELETED
- Същата схема като таблицата, за която е дефиниран тригерът

	INSERTED	DELETED
INSERT	Редовете, които са (били / предвидени да бъдат) вмъкнати	Празна (но все пак съществува)
DELETE	Празна	Редовете, които са били / щели да бъдат изтрити
UPDATE	Променените редове след промяната	Променените редове преди промяната

INSERTED и DELETED

- DML операцията, която е предизвикала изпълнението на тригера, може да е засегнала няколко реда
- Трябва така да дефинираме тригера, че да работи както при един ред, така и при няколко
- Допълнителен материал: в други СУБД може да се укаже дали тригерът да се изпълни за всеки ред поотделно или за цялата операция наведнъж

AFTER тригери – пример №1

- Да се направи така, че при добавяне на нов клас автоматично да се добавя и нов кораб със същото име и с година на пускане на вода = null

```
CREATE TRIGGER tr1
ON Classes
AFTER INSERT
AS
   INSERT INTO Ships(name, class)
   SELECT class, class
   FROM Inserted;
```

Пример №1 – тестване

- Да тестваме:

```
INSERT INTO Classes
VALUES ('Test 1', 'bb', 'Bulgaria', 20, 20, 50000),
       ('Test 2', 'bc', 'Bulgaria', 18, 21, 45000);

SELECT *
FROM Ships
WHERE name LIKE 'Test %';
```

- Докато се упражняваме, е хубаво да изтриваме ненужните тригери, за да не се получават конфликти

```
DROP TRIGGER tr1;
```

- Демонстрация

AFTER тригери – пример №2

- При изтриване на кораб автоматично да се изтрива и неговият клас, ако няма повече кораби от този клас
-
```
CREATE TRIGGER tr2
ON Ships
AFTER DELETE
AS
   DELETE FROM Classes
   WHERE class NOT IN (SELECT class
                       FROM Ships)
         AND class IN (SELECT class
                       FROM Deleted);
```

Пример №2 – тестване

- Ще използваме тестовите класове и кораби от предишния пример

```
DELETE FROM Ships
WHERE name LIKE 'Test %';

SELECT *
FROM Classes
WHERE class LIKE 'Test %';

DROP TRIGGER tr2;
```

Уловка

- Да се направи така, че при изтриване на клас да се изтриват и всички кораби от този клас
- Да прочетем внимателно условието – никъде не е казано, че искаме тригер
- Тригер не е необходим – да си припомним ON DELETE CASCADE

Пример №3 – INSTEAD OF тригер (1)

- Да се направи така, че ако при добавяне на кораб годината му на пускане е по-голяма от текущата година, то годината да бъде променена на null
- Ако в MS SQL имаше BEFORE тригери, щяхме да ги използваме

Пример №3 (2)

```
CREATE TRIGGER tr3
ON Ships
INSTEAD OF INSERT
AS
  INSERT INTO Ships(name, class, launched)
  SELECT name, class,
    CASE
      WHEN launched > YEAR(getdate()) THEN NULL
      ELSE launched
    END
  FROM Inserted;
```

Пример №3 (3)

- Вариант без CASE:

```
...
INSERT INTO Ships
SELECT *
FROM Inserted
WHERE launched <= YEAR(getdate())
UNION ALL
SELECT name, class, null
FROM Inserted
WHERE launched > YEAR(getdate());
```

- Тестване:

```
INSERT INTO Ships VALUES ('Test', 'Iowa', 2250);
SELECT * FROM Ships WHERE name = 'Test';
DELETE FROM Ships WHERE name = 'Test';
DROP TRIGGER tr3;
```

Пример №3 (4)

- Вариант с AFTER:

```
CREATE TRIGGER tr3_2
ON Ships
AFTER INSERT
AS
   UPDATE Ships
   SET launched = NULL
   WHERE name IN (SELECT name
                  FROM Inserted
                  WHERE launched > YEAR(getdate()));
```

- Недостатък: ако има ограничение за датата?

Пример №4: тригер за UPDATE (1)

- При промяна на черно-бял филм на цветен съответният продуцент да получава $100000
- Ако в една UPDATE заявка са били променени няколко филма на един продуцент, той да получи само веднъж 100000

Пример №4: тригер за UPDATE (2)

- Как да разберем какво е било променено от следната заявка?

```
UPDATE Movie
SET length = 120, inColor = 'Y'
WHERE title LIKE 'M%';
```

Deleted				
title	…	length	inColor	producerC#
M1		120	N	123
M2		90	Y	456

Inserted				
title	…	length	inColor	producerC#
M1		120	Y	123
M2		120	Y	456

- Може да съединим Deleted и Inserted по първичния ключ
  - Кога това няма да работи?
- Допълнителен материал: може да проверим и с IF UPDATE (column), но тригерът ще се извика и при промяна на клетка със същата стойност

Пример №4: тригер за UPDATE (3)

```
CREATE TRIGGER tr4
ON Movie
AFTER UPDATE
AS
  UPDATE MovieExec
  SET networth = networth + 100000
  WHERE cert# IN (SELECT i.producerc#
                  FROM Deleted d
                  JOIN Inserted i ON d.title = i.title AND d.year = i.year
                  WHERE d.inColor = 'N' AND i.inColor = 'Y');
```

- Проверка:

```
SELECT * FROM MovieExec WHERE cert# IN (SELECT producerC#
                                        FROM Movie WHERE inColor = 'N');

UPDATE Movie SET inColor = 'Y';

SELECT * FROM MovieExec;

UPDATE Movie SET inColor = 'N' WHERE year = 2001;

SELECT * FROM MovieExec; -- няма промяна

DROP TRIGGER tr4;
```

Пример №5: Валидация на данни с тригер (1)

- Даден кораб не може да участва в битка, преди да бъде пуснат на вода
- Да не се допуска промяна на данните в Outcomes, ако промените водят до нарушаването на горното изискване
- Ако се добавят/обновяват няколко реда и поне един от тях нарушава условието за коректност, цялата операция да бъде отменена

Пример №5: Валидация на данни с тригер (2)

```
CREATE TRIGGER tr5
ON Outcomes
AFTER INSERT, UPDATE
AS
  IF EXISTS (SELECT *
             FROM Inserted
             JOIN Ships ON ship = Ships.name
             JOIN Battles ON battle = Battles.name
             WHERE launched > YEAR(Battles.date))
     BEGIN
       RAISERROR('Error: ship is launched after the battle', 16, 10);
       -- има само едно "е"
       ROLLBACK;
     END;
```

- Проверка:

```
INSERT INTO Outcomes(ship, battle, result)
VALUES ('Iowa', 'North Atlantic', 'sunk');

SELECT * FROM Outcomes WHERE ship = 'Iowa';

DROP TRIGGER tr5;
```

- Въпрос: горният тригер гарантира ли ни, че изискването за коректност няма как да бъде нарушено?

- Даден тригер е дефиниран едновременно за повече от една операция, например INSERT и UPDATE
- Искаме в тялото да разграничим дали е бил извикан при INSERT или при UPDATE

```
IF EXISTS (SELECT *
           FROM Deleted)
   -- операцията е била UPDATE
```

Пример №6 (1)

- С помощта на INSTEAD OF тригерите може да се изпълняват INSERT, UPDATE и DELETE заявки върху всеки изглед
- Пример: Да се създаде изглед, който за всеки потънал кораб извежда името му и името на битката, в която е потънал
- Изгледът трябва да позволява INSERT, UPDATE и DELETE

Пример №6 (2)

```
CREATE VIEW SunkShips
AS
   SELECT ship, battle
   FROM outcomes
   WHERE result = 'sunk';
```

- Кои операции могат да се изпълнят върху изгледа?

Пример №6 (3)

```
CREATE VIEW SunkShips
AS
  SELECT ship, battle
  FROM outcomes
  WHERE result = 'sunk';

GO

-- UPDATE и DELETE могат да се изпълнят безпроблемно,
-- но при INSERT новият ред би имал result = null

CREATE TRIGGER tr6
ON SunkShips
INSTEAD OF INSERT
AS
  INSERT INTO OUTCOMES(ship, battle, result)
  SELECT ship, battle, 'sunk'
  FROM Inserted;
```

Извън света на MS SQL

- Допълнителен материал: пример за тригер в IBM DB2

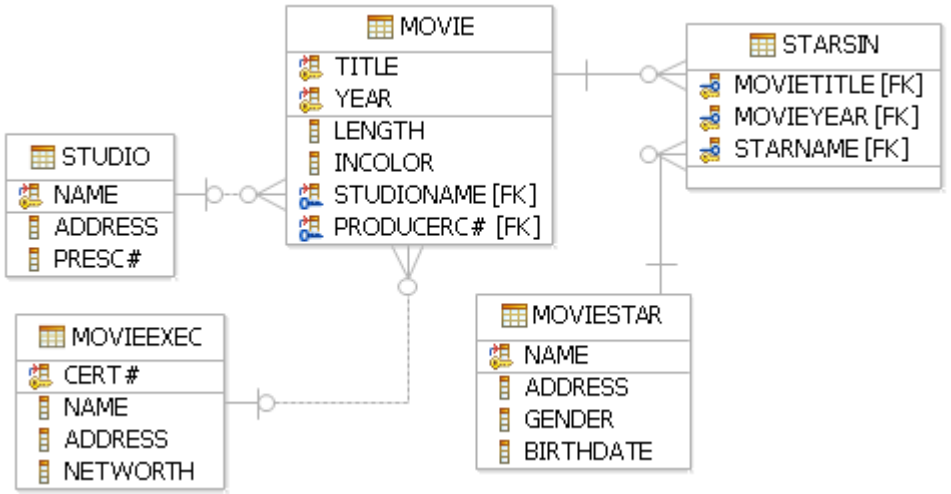
```
CREATE TRIGGER tr_ships_year
BEFORE INSERT
ON Ships
REFERENCING NEW AS n
FOR EACH ROW
WHEN (n.launched > YEAR(current_date))
SET n.launched = YEAR(current_date);
```

Въпроси?

Следват задачи

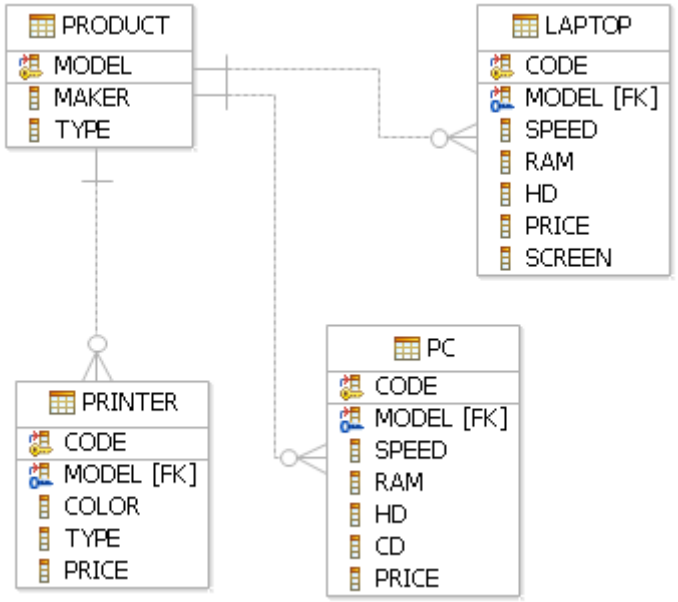
1. Задачи - Movies

1. Добавете Брус Уилис в базата. Направете така, че при добавяне на филм, чието заглавие съдържа “save” или “world”, Брус Уилис автоматично да бъде добавен като актьор, играл във филма.
2. Да се направи така, че да не е възможно средната стойност на Networth да е по-малка от 500 000 (ако при промени в таблицата MovieExec тази стойност стане по-малка от 500 000, промените да бъдат отхвърлени).
3. MS SQL не поддържа ON DELETE SET NULL. Да се реализира с тригер за външния ключ Movie.producerc#.
4. При добавяне на нов запис в StarsIn, ако новият ред указва несъществуващ филм или актьор, да се добавят липсващите данни в съответната таблица (неизвестните данни да бъдат NULL). Внимание: има външни ключове!



2. Задачи - PC

1. Да се направи така, че при изтриване на лаптоп на производител D автоматично да се добавя PC със същите параметри в таблицата с компютри. Моделът на новите компютри да бъде ‘1121’, CD устройството да бъде ‘52x’, а кодът - със 100 по-голям от кода на лаптопа.
2. При промяна на цената на някой компютър се уверете, че няма по-евтин компютър със същата честота на процесора.
3. Никой производител на компютри не може да произвежда и принтери.
4. Всеки производител на компютър трябва да произвежда и лаптоп, който да има същата или по-висока честота на процесора.
5. При промяна на данните в таблицата Laptop се уверете, че средната цена на лаптопите за всеки производител е поне 2000.
6. Ако някой лаптоп има повече памет от някой компютър, трябва да бъде и по-скъп от него.
7. Да приемем, че цветните матрични принтери (type = 'Matrix') са забранени за продажба. При добавяне на принтери да се игнорират цветните матрични. Ако с една заявка се добавят няколко принтера, да се добавят само тези, които не са забранени, а другите да се игнорират.



3. Задачи - Ships

1. Ако бъде добавен нов клас с водоизместимост по-голяма от 35000, класът да бъде добавен в таблицата, но да му се зададе водоизместимост 35000.
2. Създайте изглед, който показва за всеки клас името му и броя кораби (евентуално 0). Направете така, че при изтриване на ред да се изтрие класът и всички негови кораби.
3. Никой клас не може да има повече от два кораба.
4. Кораб с повече от 9 оръдия не може да участва в битка с кораб, който е с по-малко от 9 оръдия. Напишете тригер за Outcomes.
5. Кораб, който вече е потънал, не може да участва в битка, чиято дата е след датата на потъването му. Напишете тригери за всички таблици, за които е необходимо.