

Задача 30.7. Зад.

30.7. Зад. F a1 a2 a3  
or ships WHERE ( SELECT b.date FROM battles b1  
WHERE b1... = a1... )  
> ( SELECT ... )

9.04.2024

Семинар 7

Ногирвикачн на дати.

Добавяне (възпроизв.) на нови кортени → да добавим стойност на всеки

атрибут:

1 начин: INSERT INTO <име на таблица> с ногирвка да възпроизв. от вече Е таблица

INSERT INTO <име на таблица>

дополнително (атрибути за които не е зададен стойност)

{VALUES (<VALUE OF COL1>, ...)}

SELECT statement

или {DEFAULT VALUES}

Пример 1: да възпроизв. за актрисата ... участнико във филми

INSERT INTO starsin (starname, movieTitle, movieYear)

VALUES ('Anne Hathaway', 'Intersteller', 2014);

Трябва да съзвикаме този ред. Ако го правим Insert INTO starsin  
и поиска инициалите включват # атрибут.

ВЪЗДИХАНЕ НА ЕДИН КОРТЕН: Ако няма ограничение за NULL  
стойност, пощади е към. Starsin (starname, movieTitle) X

редът е валиден! съд от този символет <sup>сега</sup> <sub>нез</sub> не <sup>не</sup> <sub>не</sub> <sup>#</sup> няма как да е ≠.

ако пропуснем в Starsin (...) Трябва да зададем ТОЛКО ВОЛ  
колкото са брои на атрибути на таблицата.

може да присъди null  
↓ ↓  
Product N (maker, model, type)  
default 'pc'      не може null

→ ако придаваш null default-на ще е null?

Insert into productN (maker, model)

Values ('A', '1234')      може и 'A', null

ще бъде възстановена ('A', '1234', 'pc')

2 начин : INSERT INTO productN

VALUES ('A', '1234', default)

означава да ги даде стойност  
за всеки един отривут

в ръчица същия резултат

Ако е : INSERT INTO productN

VALUES ( DEFAULT, DEFAULT, DEFAULT )

ще бъде възстановена ( null, null, pc )

ако котре да напечата null, default = null.

Ако отривут котре да напечата default стойност нами чрез

default: INSERT INTO productN

DEFAULT VALUES:  $\rightarrow$  котретн с default-ни стойности

ПОВЕЧЕ КОТРЕНИ:

INSERT INTO productN (maker, model, type)

VALUES ('A', '1234', 'laptop'),  
( 'A', 578, 'printer'),  
( 'A', 593, 'laptop' )

възстановка 3

котретни

2 вариант чрез подзаписка :

VALUES тъй като не се използват, вместо него ще използвате

подзаписка - ще възстановяте котретните извлечени от подзаписката

в Product N ще възстанови всичко от Product за pc

INSERT INTO productN (maker, model, type)  $\rightarrow$  котрет в търсено и

SELECT maker, model, type

FROM product

WHERE type = 'pc'

пег да избере

възстановено е да

възстанови котретни

ВАЛЮ! ще възстанови в SELECT пеги, в същата

мощност и броя

productN (type, model, maker)

SELECT (type, model, maker)

Не може да има редови с груп:

INSERT INTO productN

SELECT type, model, maker

|  
↳ ако е maker, model, type може

от тази стока ред, брои, тип да съвпадат. Ако и двете са от тази символна група, тогава то е правилно. Редът е валиден!!!

## ПРОМЯНА НА СТОЙНОСТИ В СЪЩЕСТВУВАЩИ КОРТЕНИ

UPDATE <tablename>

SET {<columnname>} = {<expression> | NULL | DEFAULT } ; ignore  
attempts

Може да има подзаписка [ { WHERE <predicate>} ] ; тя се изпълнява  
като този кортентури удавлят всички where условието.

! Не може с 1 UPDATE да променят 2 кортента.

Пример 2: да подадем цената на всички лаптопи с размер на екрана < 200

UPDATE laptop

SET price = price \* 0,9, hd = hd / 2, ram = default

WHERE hd < 200

Пример 3 UPDATE laptop не кога груп. от нн се з погрешка.

SET speed = (SELECT MAX(speed)  
FROM laptop)

одинаква 1 стойност

Всички редове се кодифицират като тип на where клузда.

## ИЗТРИВАНЕ НА КОРТЕНН

от 1 таблица

DELETE FROM <table name>

WHERE <predicate> Такъто може да се изтрива  
цялата таблица или да се изтриват определени редове.

Пример 4: DELETE FROM starsin

WHERE starname = 'Anne' AND  
movietitle = 'Interstellar' AND  
movieyear = 2014 ;

да изтриваме всички редове от таблицата starsin

DELETE FROM starsin  
WHERE starname = 'Anne' AND  
movietitle = 'Interstellar' AND  
movieyear = 2014 ;

да изтриваме всички редове от таблицата movie

DELETE FROM movie  
WHERE (SELECT MIN(year)  
FROM moviem m)

WHERE m.producer# = producer# )  
което означава

16.04.2024

Семинар 8

• `decimal ( precision , scale )`

контролиране на чифрите

дробно число с контрола над отблизо към десятичната запетая, различно от FLOAT

• `decimal ( 5,2 )` се организира във вид

дробна част 5 цифри

123,456 това число ще се запише е 123.46

1234.52 → ще изведе грешка защото стойността е 5 и 3 нули

цифри след запетая. Но проблема е че precision = 5 и в този  
 случай ще имаме повече от scale. Нека вместо 3 да имаме  
 4 дист.

• `char(n)` За символни низове с фиксирана дължина

• `varchar(n)` За символни низове с нависяща n символа

char(3) некато ще запиши 'a' и ще е здравише дали

се добавят ли за да стате 3.

ако е varchar(3) и некато ще запиши 'a' ще се дади само 1  
byte за 'a'

char е ефективен по памет и като символните низове с по-голяма  
дължина.

- INT / INTEGER - цяло число 32 bit със знак
- FLOAT / REAL
- DATETIME / DATE, TIME

Varchar е идентичен като ще срещните с различни разновидности за засега видели и като такова това трябва.

nchar / nvarchar за символнi извори които се кодират като кога ДЕСРИПРАНЕ НА РЕЛACIONНА СХЕМА.

CREATE DATABASE <името база данни>

USE <име на база данни>

CREATE TABLE <име на таблица>

(име на колона 1 тип на данни (размер)

[име на колона 2 тип на данни (размер)

...]

):

пример:

CREATE TABLE moviestar

(name, varchar(30),

address, varchar(50),

gender, char(1) name goes to пропуск

birthday, DATE

);

• SELECT име на колона 1, [ име на колона 2, ... ]

INTO име на нова таблица

FROM име на 3 таблица

[where ...];

IDENTITY

id (seed, increment) NO default e(1,1)

от този столбец възможност конто га управлена еку ми  
на бързо

Идентификационните което ще гарантира униканност.

за генериране на уникадин столбци за 1 колона в таблица

```
CREATE TABLE moviestar  
(id int IDENTITY (1,1),  
name ....  
...);
```

identity ќе ни позволява да  
възстановим стойността

```
SET IDENTITY_INSERT  
moviestar OFF  
ON
```

да ли или сири и възможността да  
възстановим го използвам, е допълнително

искаме да възстановим стойността в moviestar  
INSERT INTO moviestar (name, address, gender, birthdate)  
VALUES ('A', 'B', 'F', '2000-01-01');

ако id беше настъпил в средата може работи, тъй като значение  
за id ще се запази стойност от нас а се генерира  
ако се опитаме да им дадем стойност ще даде грешка.  
Може и отрицателни числа тип seed, increment

2 варант за обявяване на схема на редовни от Е таблица  
SELECT title, year, length, ...  
INTO coloredmovie  
FROM movie  
WHERE incolor = 'y';

Ще се създаде нова таблица с същият формат и атрибути на  
Е таблица. ще се запазят и ограниченията за преносите на NULL  
стойности.

• Ако искаме да имаме различна място в новата таблица "две"  
последователни кръг SELECT.

Ако искаме да имаме различна схема на редовни  
1. добавяме на нови атрибути  
2. меняем за размер, ограничения  
3. меняхме място на колони

некоректни редови

## ПРОМЯНА НА СХЕМАТА НА РЕДАЦИЯ

да промените moviestar и да добавите нов отривът  
ALTER TABLE moviestar (има на таблица)

ADD new HD колона 1, тип (размер)

пример: ALTER TABLE moviestar

ADD phone CHAR (12); phone може да приема NULL стойност ако не едното експлицитно в таблицата

name address ... phone ←

Задава NULL.

A B A | NULL

съществуващ

ако всички тези

ADD phone char (12) NOT NULL;

ще дава грешка, ако няма стойност

но тази е буфер:

... NOT NULL DEFAULT 'Unlisted'

или

да изтрите ИЗТРНВАНЕ НА

ALTER TABLE moviestar

ATPN GYT.

DROP COLUMN phone, birthdate; ще изтрият 2 отривъта  
да подгрижат

ALTER TABLE moviestar

ALTER COLUMN phone char (16) NOT NULL

СТОЙНОСТИ ПО МОДРАЗБРАНЕ

CREATE TABLE moviestar

ако нямат стойноста по

(name VARCHAR (30),

модразбрани га не е NULL

phone char (12) DEFAULT 'unlisted' или regdate DATE DEFAULT

), INSERT INTO moviestar DEFAULT VALUES CURRENT\_DATE  
INSERT INTO moviestar VALUES (default regdate DATETIME GETDATE  
);

ALTER TABLE moviestar

ADD phone2 CHAR (12) DEFAULT 'unlisted';

за текущата  
гри.

ALTER TABLE moviestar

ADD DEFAULT 'myName'

FOR name;

## ИЗТРИВАНЕ НА РЕДАЦИИ

За да изтриват 1+ таблица  
DROP TABLE moviestar;  
DROP DATABASE име-на база-от данни.

DROP TABLE

DELETE

премахва цялото таблица

премахва всички горещи

24.04.2024

Семинар 9

## ОГРАНИЧЕНИЯ НА НИВО ТАБЛИЦА

- налагат правила върху данните в таблиците
- предизвикват от изтриване на таблица, ако е друга таблица която зависи от нея.
- могат да съдят заради ограниченията на редицата схема или да се добавят след това.
- съхраняват се в речника на данните.

### ВИДОВЕ

ПЪРВИЧЕН КЛЮЧ Primary key

Ограничение за уникантност

UNIQUE

- единствично определя всеки ред в таблицата
- не容许 NULL стойност
- таблица може да има само 1

- единствично определя 1 ред в таблица
- колоната ограничена с UNIQUE може да съдържи NULL стойност
- таблицата може да има 1+ уникатен клюц

ВЪНШЕН КЛЮЧ FOREIGN KEY

- атрибута от една таблица може да ресферира атрибут от друга таблица. допуска се мърквата и втората таблица да съвпадат ресферираният атрибут от 1<sup>o</sup> таблица трябва да са UNIQUE или PRIMARY KEY.

• стойностите на атрибутите, които са десктарирани като Foreign key в 1<sup>o</sup> таблица:

→ трябва да се използват като стойности на ресферираният атрибут в коректните на втората таблица

→ могат да приемат NULL ако има NOT NULL ограничение.

## Аернаприоде Но Primary Key

- CREATE TABLE moviestar (  
name char (30) CONSTRAINT pk\_moviestar PRIMARY KEY,  
address ...  
);
- CREATE TABLE moviestar (  
name char (30),  
address varchar (255),  
CONSTRAINT pk-moviestar PRIMARY KEY (name));
- CREATE TABLE movie (  
title varchar (50),  
year INT,  
length INT,  
CONSTRAINT pk-movie PRIMARY KEY (title, year));

## Аернаприоде Но UNIQUE knroh.

- CREATE TABLE movie (  
title varchar (50),  
year INT,  
length INT,  
CONSTRAINT uk\_movie UNIQUE (title, year)  
);

## Аернаприоде Но Foreign Key

- Ако foreign key е създан от 1 страници, може да добавим  
съзнателно в таблици и генерирати:  
REFERENCES <parent-table> (<parent-table-attribute>)

- Альтернативно може да добавим към създаване от страници B.  
CREATE TABLE light или може да генерират, предвиден че икономство  
от страници A е външният knroh.

FOREIGN KEY (<child\_table\_attributes>) REFERENCES <parent\_table> (<parent\_table\_attributes>)

### ДОГАДЯНЕ НА ОГРАНИЧЕНИЯ

ALTER TABLE <име на таблица>

ADD CONSTRAINT <име на ограничение> <типа на ограничение>;

Не е задължително.

Пример 1: CREATE TABLE table1 (  
col1 INT NOT NULL,  
col2 char(2) NOT NULL);

CREATE TABLE table2 (  
col1 Char(10) NOT NULL,  
col2 INT);

ALTER TABLE table1

ADD CONSTRAINT pk-table1 PRIMARY KEY (col1)

ALTER TABLE table2

ADD CONSTRAINT uq-t2 UNIQUE (col1)

ALTER TABLE table2

ADD CONSTRAINT fk-t2-t1 FOREIGN KEY (col2) REFERENCES table1 (col1)

### ПОЛНТИКИ МПН ВЪНШНИ КЛЮЧ

- Задавате на ограничение по ресферентната членост.

- Определят поведението на СУБД при ON UPDATE / ON DELETE на ресферентна запис.

NO ACTION - т действа когато нарушава ресферентната членост се отхвърля

CASCADE - при изтриване / промяна на стойност в ресферентната колона се изтрива / обновява съответната в ресферентните записи. (\*\*)

SET NULL - (\*) се поставя NULL (\*\*). Ако това NOT NULL ограничение е наложено, то тази стойност не може да бъде поставена.

SET DEFAULT - т коремин коло сочат към коремин коло ще бъдат изтривани, получават стойността си по подразбиране.

Тази стойност трябва да е в ресферентната колона.

Полнитките мпн външни ключове могат да бъдат указанi при дефиниране на релационната схема или да се добавят след това.

REFERENCES <table\_name>(<col-name>)

[ON DELETE | UPDATE { CASCADE | NO ACTION | SET NULL | DEFAULT }]

Не е задължително незримо да се укачат политики за външни  
ключове, като да се посочи политика или за DELETE/UPDATE.  
По подразбиране е NO ACTION.

За един и същ <sup>външни</sup> ключ политиката при DELETE като  
да е една и при UPDATE различна.

**Пример:** ALTER TABLE t2

ADD FOREIGN KEY (col2)

REFERENCES t1 (col1)

ON DELETE CASCADE

ON UPDATE SET NULL;

**CHECK** - проверява дали записите която се възпроизват  
или обновяват, отговарят на предварително зададено условие.  
Условието трябва да е просто, като да се постави и в WHERE  
може да съдържа логически изрази създади с AND/OR  
и те могат да имат негативни.

CHECK - на ниво таблича / ограничение

**Пример**

Бапнат 1 Create table t1 ( **На ниво ограничение** )

col1 INT not null,

col2 Char(2) not null Check (col2 IN ('BG', 'FR'));

Бапнат 2 Create table t1 ( **На ниво таблича** )

col1 INT not null,

col2 Char(2) not null,

check (col2 IN ('BG', 'FR'))

);

Добавяне на CHECK тип ограничение на релационната структура  
- предиштвото: условието може да съдържа логически изрази, вкл. няколко колони от таблицата. чрез var. 1 е невъзможно.

Пример: CREATE TABLE t1 (

col1 INT NOT NULL,

col2 CHAR(2) NOT NULL,

CHECK (col2 IN ('BG', 'FR') AND col1 > 1 AND col1 < 10));

или като таблицата е обявена:

ALTER TABLE t1

ADD CONSTRAINT ck\_mtl\_col2

CHECK (col2 IN ('BG', 'FR'));

За col2 може да се добави 1+ check. В този случай условията от check за egta и свидетелства колони са свързани с AND

ALTER TABLE ...

ALTER TABLE ...

Check (col2 IN ('BG', 'FR'));

Check (col2 IN ('GR', 'RU'));

ИЗТРИВАНЕ НА ОГРАНИЧЕНИЯ

ALTER TABLE <име на таблица>

DROP CONSTRAINT <име на ограничение>;

↓

pk-mytable

fk ...

ug ...

ck ...

## ИНДЕКСИ И ПЪЗЛЕДИ

Индексът е:

обект от базата от данни

сортирата структура от данни

използва се от СУБД да ускори връщането на редове  
като използва указатели към адреси в файла.

Може да нацили дисковите входно/изходни операции като

използва бърз метод за нациране на местоположението на данни.

### ТИПОВЕ ИНДЕКСИ

Уникатни | Несъвкупни

Клъстеризирани | Неклъстеризирани

Мрежи (1 атрибут), n съставни (1 + атрибут)

Създадени - при създаване на Primary Key или UNIQUE

ОБЩ СИНТАКСИС:

CREATE [UNIQUE]

[CLUSTERED | NONCLUSTERED]

? не за залагането

INDEX index-name

ON TABLE (column[, column], ...)

Пример:

CREATE INDEX idx ON starsin(movietitle)

### ИЗТРИВАНЕ НА ИНДЕКСИ

DROP INDEX table-name.Index-name;

DROP INDEX index-name ON table-name;

Пример:

DROP INDEX ships.indx;

DROP INDEX idx ON ships;

## ИЗБОР НА ИНДЕКСИ

предимство: ускоряват извлечението на данни.  
недостатък: забавят / усложняват DML операциите, защото след като променят таблиците, събг трябва да промени индекса.

### Кога да създадате индекси?

ако: данните в таблицата се променят редко.

колоната съдържа голям # ≠ стойност

1 или 1+ колони са често използвани заедно в WHERE клуза, в условие за свързване, в GROUP / ORDER BY.

таблицата е горна и повечето заявките се очаква да върнат по малко от 20% от редовете.

### Кога да не създадате индекси?

ако: данните са променят често

таблицата е малка

колоните не са използвани често в условия в заявките

заявките връщат повече от 20% от редовете

индексираният колони са използвани като част от изрази а не съкост от тях.

## ИЗГЛЕД

виртуални (логически) таблици

не са физически върху диска

не са собствени данни а прозорци през които данните от таблиците катат да съдържат гледанти / променятели.

създават се на базата на таблици или други изгледи или и реде съхраняват се като SELECT

испрат се използват като обикновени таблици в SELECT, INSERT, UPDATE, DELETE.

За това се използват избрани

За ограничаване на достъпа до данни

За нещо ги създава съдържанието

За осигуряване на точна независимост по отношение  
на структурата на данни

За представяне на несъществуващи  
данни

за "изчиняване" и "претваряне" на данни.

Създаване на избрани.

CREATE VIEW view-name

[[alias[, alias]...]]

AS

Subquery

[WITH CHECK OPTION];

Важното пример на избрани със средните брой оръдия на производителите

1. knowe no country

CREATE VIEW v\_1

AS

SELECT AVG(numguns) AS avgguns,

country

FROM classes

GROUP BY country;

Важното

CREATE VIEW v\_1

(avgguns, country)

AS

SELECT ... -||-

... -||-

Задача 8.1 Узнать

SELECT \*  
FROM v1  
ORDER BY numguns DESC;

Создание Виды на базе

CREATE VIEW v-ships-full-info  
(name, type, numguns ...)  
AS

SELECT s.name, c.type, ...  
FROM classes c JOIN ships s ON c.class = s.class,

Многие

назначение с + американским кораблем

CREATE VIEW v-USA

AS

SELECT \*

FROM classes

Where country = 'USA'

WITH CHECK OPTION;

INSERT INTO v (class, type, country)  
VALUES ('...', '...', '...');

Удаление Виды на базе. DROP VIEW view-name

Изменение Виды на базе

ALTER VIEW view-name [ (alias [alias]...) ]  
AS

Subquery

[WITH CHECK OPTION];

или

DROP + CREATE

## ВИДОВЕ ИЗГЛЕДИ

	прост изгледи	съдържащи изгледи
# Таблица	1	1+
Съдържащ филтър	Не	ga
съдържащ групиратъ филтър	Не	да
когато да се използват	да	Почти да
DML операции върху изгледа		

## МОДИФИЦИРАНЕ НА ДАННИТЕ ЧРЕЗ ИЗГЛЕДИ

INSERT, UPDATE, DELETE

изгледи с UNION, INTERSECT, EXCEPT

изгледи съдържащ GROUP BY, AVG, SUM, MAX, DISTINCT,

изгледи съдържащ върху повече от 1 таблица

Не могат да се използват за модифициране

Правила:

Не може да се изтърива ред от изглед само изгледът

съдържащ групова филтър и групирани на единния, както в DISTINCT.

Също и за модифициране + или константа, дескриптори чрез израз.

Също и за добавяне на единния чрез изглед + или в базовата таблица или NOT NULL константа която не да е включена в изгледа.

ТВОЕДЕНИЯ и ПРИГЕРН

Create ASSERTION sumlength

**CHECK** (10000 >= ALL (SELECT SUM (length)  
FROM movie

а определена в группу по студии (Group By studioname);

Check-а се проверява при insert и при update. Не се проверява при delete.

**Твърдение (assertion)** – ограничение на типове съда на логически SQL заявки като външни съди.

**CREATE ASSERTION** <имя\_таб-результата>  
**CHECK** <условие>

условните на твърдениято трябва вниманието да бъде изпълнено, и в  
когато то създаде възможност за твърдението да е при разрешаването на някои  
от бъдещите етапи.

**Пример 1:** # на резервните За даден полет трябва да не е по-голям от # на кристата в самолета за съответния полет:

CREATE ASSERTION booking.s.length = 5

CHECK ( NOT EXISTS

SELECT \*

FROM FLIGHTS as F, AIRPLANES as A

WHERE F. AIRPLANE = A. CODE

AND SEATS < (SELECT COUNT(\*)

FROM BOOKINGS

WHERE FLIGHT\_NR = F\_NR));

## TBbpfeline vs Check at batheane

За разлика от Check, твърдението са външни верти. При Check, към условната съвръната подразделка, условното на ограниченията не външни верти ще са редуцирани на стойностите като се виждат в таблиците.

Задължима предизвик да е част от условното на Check ограничение.  
Твърдението се дефинира на ниво схема на табличка.

Твърденията позволяват да се зададат на таблица и  
твърденията се проверяват винаги като поддържат  
попълните участници в условието, докато Check  
се проверяват само при Insert / UPDATE на табличката за  
която са дефинирани.

### ИЗТРИВАНЕ НА ТВЪРДЕНИЕ

Условията на твърденията не могат да бъдат променени /  
изтрити. Ако искате да промените едно твърдение  
трябва да го изтрите и да създадете отново с новото  
условие. Се изтрява с командата:

DROP ASSERTION <име\_твърдението>

**ТРИГЕРН (Triggers)** са обекти в бд RDBMS се различават  
от другите ограничения. На това се използват при настъпването  
на както събитие, вследствие на което се поставянето  
условие в тригера е удовлетворено се изполнява определено  
действие.

Са дефинират за конкретна табличка / изглед.

Всички създавате тригера се задействат автоматично при  
INSERT, UPDATE, DELETE

Пример 2: Create table t (

t\_id INT NOT NULL IDENTITY

t\_name VARCHAR(20) NOT NULL

CONSTRAINT pk\_t PRIMARY KEY (t\_id)

);

CREATE table t\_audit (

audit\_id INT NOT NULL,

audit\_name VARCHAR(20) NOT NULL

trigger\_table CHAR(8) NOT NULL);

ограничение  
изтребител

таблица е пътна

при inserted / deleted



СБЗД АВАИЕ НА ТРНГЕРН

CREATE TRIGGER trigger name

ON {table | view}

{ FOR | AFTER | INSTEAD OF }

{ [ INSERT ] [,] [ UPDATE ] [,] [ DELETE ] }

AS { sql-statement }

нпример 3: CREATE TABLE employee (

emp-code VARCHAR (10) PRIMARY KEY,

name VARCHAR (50) NOT NULL,

3а позиции designation VARCHAR (20) NOT NULL,

deleted BIT NOT NULL DEFAULT 0);

2. INSERT INTO employee <sup>TYPE</sup> VALUES ('emp-code1', 'nameA', 'manager', 0),  
('emp-code2', 'nameB', 'clerk', 0);

3. CREATE TRIGGER trig\_employee\_delete ON employee  
INSTEAD OF DELETE

AS

UPDATE employee

SET deleted = 1

WHERE empcode IN (SELECT empcode  
FROM deleted);

РЕЗУЛЬТАТЫ

4. DELETE FROM employee

WHERE designation = 'manager';

OUTPUT deleted,\*

5. SELECT \* FROM employee;

empcode	name	designation	deleted
empcode1	nameA	manager	1
empcode2	nameB	clerk	0

## AFTER ТРИГЕРН:

изпълняват се след като дадена модификация (INSERT, UPDATE, DELETE) завърши успешно.

ограниченията определени върху табличата се проверяват преди да се изпълни команда INSERT | UPDATE | DELETE и ако тези ограничения не се удовлетворяват то тригерът не се изпълнява. Могат да се създават също върху табличи за дадена команда се позволява изпълнение от AFTER тригер INSTEAD OF тригер.

изпълняват въвеждат модификацията с INSERT, UPDATE или DELETE преди да бъдат проверени ограниченията върху табличата се създават за табличи или изгледи.

за дадена таблица и дадено действие (INSERT, UPDATE или DELETE) се позволява само 1 INSTEAD OF тригер.

В дефиницията на тригера се указва и кое е събитието кое то действие тригера - INSERT || UPDATE || DELETE

възможно е тригерът да бъде задействан и при тригър действа за които да се изпълнява едно и също действие. В този случай при дефиниране на тригера се указват и трите конюнции.

достъпът до стойностите на коректните преди и след настъпване на събитието се осъществява посредством специални табличи това са табличите INSERTED & DELETED

ДИКонюнција Еквивалентна таблица

Описани

INSERT

inserted

съдържа данните които току чу се били въведени

DELETE

deleted

съдържа данните които току чу са били изтрити

UPDATE

inserted

които на данните след, преди

deleted

обновяването.

## ТРИГЕРН vs. ТВЪРДЕНН

- Твърденията се проверяват при ~~и~~ произтича на резултати, включени в дефиницията на твърдениято. Това отнема иного време. Тригерите се изпълняват при определени събития, следовани от потребителя и при всяко друго извършване. Това отнема по малко време.
- Твърденията не показват данни, само проверяват дефинираното условие. Тригерите показват условия и да показват данни.