Модификации на данните в базата данни

До този момент, вниманието ни бе насочено към възможността да извличаме данни посредством SELECT-FROM-WHERE израза в SQL. Съществуват и редица други оператори, които не връщат резултат, а по-скоро променят състоянието на базата от данни. В днешното упражнение ще разгледаме три типа оператори, които ни позволяват да:

- Вмъкваме кортежи в релация;
- Изтриваме известни кортежи от релация;
- Променяме стойностите на някои компоненти в съществуващи кортежи.

Вмъкване на данни
Основната форма на оператора за вмъкване се състои от:
Ключовата дума INSERT INTO,
Името на релацията R, Списък от атрибути на релацита R, заграден в скоби,
Ключовата дума VALUES и Кортежен израз, който представлява списък с конкретни стойности, заграден в скоби, по една стойност за всеки атрибут от списъка (3).
Това означава, че основната форма на оператора изглежда по следния начин:
INSERT INTO R(A1,...,An) VALUES (v1,...,vn);
Кортежът се създава, като се използват стойностите vi за атрибутите Ai, i=1,2,..., n. Ако списъкът от атрибути не включва всички атрибути на релацията R, тогава създаденият кортеж има стойности по подразбиране за всички липсващи атрибути. Обикновено тази стойност е NULL, но има и други възможностти, които ще разгледаме в следващите упражнения.

Пример 1.
Да предположим, че искаме да вмъкнем *Sydney Greenstreet* в списъка от актьорите, участвали в *The Maltese Falcon*.

```
INSERT INTO StarsIn (movieTitle, movieYear, starName);
VALUES ('The Maltese Falcon', 1942, 'Sydney Greenstreet');
```

Ефектът от изпълнението на този оператор е вмъкването на кортеж с трите компоненти от ред (2) в релацията StarsIn. Тъй като всички атрибути от StarsIn са споменати в ред (1), няма нужда от компоненти по подразбиране. Стойностите от ред (2) съответстват на атрибутите от ред (1) в даденият ред, така че 'The Maltese Falcon' става стойност на компонентата за атрибута movieTitle и т.н.
Ако, както е в даденият пример, даваме стойности на всички атрибути в релацията, тогава можем да изпуснем списъка от атрибути, следващ името на релацията:

```
INSERT INTO StarsIn
VALUES ('The Maltese Falcon',1942,'Sydney Greenstreet');
```

Ако използваме тази възможност обаче, трябва да сме сигурни, че редът на стойностите е същия като стандартния ред на атрибутите в релацията. Ако не сте сигурни в него, по-добре е да изброите атрибутите в INSERT клаузата в реда, които сте избрали за техните стойности във VALUES клаузата.
Простият INSERT, показан в този пример само вмъква кортеж в релацията. Вместо да използвате експлицитни стойности за един кортеж, можете да определите множество от кортежи, които да бъдат вмъкнати, като използвате подзаявка. Подзаявката заменя ключовата дума VALUES и кортежния израз в по-горе описания оператор INSERT.

Пример 2.

Да предположим, че искаме да добавим към релацията Studio(name,address,presC#) всички студия на филми споменати в релацията

Movie(title,year,length,incolor,studioName,producerC#),

които все още ги няма в релацията Studio. Тъй като няма начин да определим адреса или президента на всяко студио, ще сложим NULL стойности за атрибутите address и presC#.

```
INSERT INTO Studio(name)
SELECT DISTINCT studioName
FROM Movie
WHERE studioName NOT IN
        (SELECT name
         FROM Studio);
```

Като повечето SQL заявки, в които имаме влагане е по-лесно да ги изследваме от вътре навън. Редове (5) и (6) генерират всички имена на студия в релацията Studio. По този начин, ред (4) проверява дали името на студио от релацията Movie не е измежду тях.
Редове (2)-(6) изкарват множеството от студия намиращи се в релацията Movie, но не и в Studio. Употребата на DISTINCT в ред (2) ни осигурява, че всяко студио ще се появи точно веднъж в това множество, без значение колко филми притежава. Накарая, ред (1) вмъква всяко от тези студия в релацията Studio с NULL за атрибутите address и presC# (, при условие, че при дефинирането на схемата на релацията Studio, не е указано нещо друго).
Забележка.
Пример2 илюстрира тънък момент от семантиката на SQL операторите. По принцип, изпълнението на заявката от (2) до (6) би трябвало да се изпълни преди изпълнението на вмъкването от ред (1). По този начин, няма вероятност нововъведените кортежи да окажат въздействие върху условието от ред (4). Въпреки това, от съображения за ефективност, е възможно ред (1) да се изпълни преди изпълнението на подзаявката да е завършило, така че направените в Studio промени да бъдат открити по време на изпълнинието на редове (2)-(6). В конкретният пример, няма значение дали вмъкването се отлага, докато заявката завърши изпълнението си или не. Но има други заявки, където резултатът се променя и ще зависи от времето на изпълнение на вмъкванията. Например, ако махнем ключовата дума DISTINCT от ред (2). Ако изпълним заявката (2)-(6), преди да сме направили някакво вмъкване,тогава ново име на студио, появяващо се няколко кортежа на Movie, ще се появи няколоко пъти в резултатът от изпълнинието на заявката и ще бъде въведено няколко пъти в релацията Studio. Обаче, ако вмъкнем нови студия в Studio веднага след като ги огкрием впо време на изпълнинието на редове (2)-(6), тогава същото име на студио няма да бъде въведено два пъти, тъй като името му няма вече да удовлетволява условието от ред (4).

Изтриване на данни
Операторът за изтриване се състои от :

Ключовите думи DELETE FROM,
Името на релацията, да кажем R,
Ключовата дума WHERE и
Условие

Това означава, че операторът изглежда по следния начин:

DELETE FROM R WHERE <condition>;

Ефектът от изпълнението на този оператор е изтриването то релацията R на всеки кортеж, които удоблетворява условието (4).

Пример 1.
Можем да изтрием от релацията StarsIn(movieTitle,movieYear,starName)
фактът, че Sydney Greenstreet е актьор в The Maltese Falcon чрез следният SQL оператор:

```
DELETE FROM StarsIn
WHERE movieTitle='The Maltese Falcon' AND
  movieYear=1942 AND
  starName ='Sydney Greenstreet';
```

Забележете, че за разлика от оператора за вмъкване, не можем просто да определим кортежа, който искаме да изтрием. Вместо това, трябва да опишем как трябва да изглежда той в WHERE клаузата.
Ако имаме вмъкнати 2 или повече записи за *Sydney Greenstreet*, то ще ги изтрием всичките.

Пример 2.
Този път ще изтрием от релацията

MovieExec(name,address,cert#,netWorth)

Няколок кортежа наведнъж, като използваме условие, което се удоблетворява от повече от един кортеж. Операторът:

```
DELETE FROM MovieExec
WHERE netWorth<10000000
```

Изтрива всички директори на филми, чиято печалба е по-малка от 10 милиона долара.

Промяна на данни
Можем да гледаме на вмъкванията и изтриванията на кортежи като на промени в базата от данни, но операторът UPDATE в SQL дава възможност за един по-специален вид промяна: един или повече кортежи, които съществуват в базата от данни, могат да променят някои от свойте компоненти.
Основната форма на оператора UPDATE се състои от:

Ключовата дума UPDATE,
Име на релация, да речем R,
Ключовата дума SET,
Списък от формули такъв, че всяко множество от атрибути на релацията R е равно на стойност от израз или константа,
Ключовата дума WHERE и
Условие.

Това означава, че операторът изглежда по следния начин:

UPDATE R SET <new-value assignments> WHERE <codition>;

Всяко присвояване на нова стойност (ред (4)) се състои от атрибут, знак за равенство и формула. Ако има повече от едно присвояване, тет рябва да се отделят със запетаи.
Ефектът от изпълнението на този оператор е намирането на всички кортежи в R,

удовлетворяващи условието (6) и промяната на всеки от тях. За целта се пресмятат стойностите на формулите от (4) и се присвояват на компонентите от кортежа за съответните атрибути на R.

Пример.
Нека модифицираме релацията
MovieExec(name,address,cert#,netWorth)
Като добавим заглавие Pres пред името на всеки директор, който е президент на студио. Условието, което желаните кортежи трябва да удоблетворяват е техните сертификационни номера, да се появяват в компонента presC# на някой кортеж от релацията Studio.

```
UPDATE MoviExec
SET name='Pres. '+name
WHERE cert# IN (SELECT presC# FROM Studio);
```

Ред (3) проверява дали сертификационните номера от кортежът в MoviExec е един от тези, които се появяват като сертификационни номера на президенти в Studio.
Ред (2) изпълнява промените в селектираните кортежи. Операторът “+” в MS SQL SERVER конкатенира низове, така че израдът следващ знака “=”в ред (2) слага символите “'Pres. “ пред старата стойност на компонентата name в конкртния кортеж. Новият низ става стойност на компонентата name в този кортеж.

Задачи

Извършете следните модификации в базата от данни Ships:

- За двата британски бойни кораби от класа Nelson – Nelson и Rodney въведете следните факти в базата от данни: пуснати на вода през 1927 година, имат девет 16-инчови оръдия и водоизместимост от 34 000 тона.
- За два от трите бойни кораби на италианския клас Vittorio Vento - Vittorio Vento и Italia, въведете следните факти в базата от данни: пуснати на вода през 1940 година; третият кораб от този клас, Roma, е пуснат през 1942. Всеки от тях има девет 15-инчови оръдия и водоизместимост от 41 000 тона.