Упражнение 9 - Ограничения.

Един от сериозните проблеми, с които се срещат разработчиците на приложения са свързани с факта, че при промяна на базата от данни новата информация може да бъде "грешна" по различни начини. Най-логичният подход е така да се напишат приложни програми, чрез които се променя информацията в базата от данни, че те да не допускат въвеждането на грешни данни. Този подход обаче има редица недостатъци – изискванията за коректност обикновено са сложни и се извършват многократно – това би довело до забавяне на приложните програми и увеличаване на мрежовия трафик. В клиент/сървър среда необходимостта да се модифицира или да се реализира допълнително ограничение е неприятна задача, ако приложните програми са инсталирани на машини, разположени в различни географски региони. Съществуват и редица други усложнения, възникващи в резултат на много-потребителската и конкурентна среда за работа, възможността за въвеждане на данни чрез средства различни от програмите на дадено приложение и др. За щастие, SQL предоставя техники за изразяване на ограничения за цялостност като част от схемата на базата от данни. Какво представляват ограниченията? Те налагат определени правила на ниво база от данни – налагат се определени правила върху данните в таблиците и се следи за тяхното ненарушение, когато даден ред се въвежда, изтрива или променя. Предпазват изтриването на таблица, ако съществува друга таблица, която зависи от нея.

Ключове и външни ключове

Може би най-важният вид ограничение в една база от данни е декларацията, че даден атрибут или множество от атрибути формират ключ за съответната релация. Ако множеството от атрибути S е ключ за релацията R, то всеки два кортежа от R, трябва да се различават по отношение на стойността на поне един атрибут от множеството S. Това правило се прилага и по отношение на дублиращите се кортежи - ако R има деклариран ключ, то тя не може да съдържа еднакви кортежи.

Ключовите ограничения, като много други ограничения, се декларират чрез командата CREATE TABLE. Има 2 подобни начина за деклариране на ключове – чрез използване на ключовата дума PRIMARY KEY или ключовата дума UNIQUE. Но една таблица може да има само един първичен ключ и произволен брой уникални ключове. SQL използва още думата *ключ* във връзка с някой ограничения за "референтна" цялостност. Тези ограничения, наречени "външни ключове", твърдят че стойност, появяваща се в една релация, тярбва да се появява също като компонента на първичня ключ на друга релация.

Деклариране на първичен ключ (PK)

Една релация може да има само едни първичен ключ. Има 2 начина за деклариране на PK в оператора CREATE TABLE, които дефинира съхранена релация.

Можем да декларираме *един* атибут като първичен ключ, когато този атрибут се описва в релационната схема. За целта ключовата дума PRIMARY KEY се поставя след атрибута и неговият тип. Можем да добавим към списъка от елементи в схемата, допълнителна декларация, която казва, че конкретен атрибут или *множество от атрибути* формират първичния ключ на релацията. Ефекът от декларирането на PK на

релацията е двоен: Два коретжа от R не могат да съвпадат по отночение на всиките атрибути от ключа S. Всеки опит да въведем или променим кортеж, който нарушава това правило бива отхвърлен от СУБД със съобщение за грешка.
Атрибутите в S не могат да приемат NULL стойности.
Пример 1
Да разгледаме схемата MovieStar. Атрибутът name е един добър избор за първичен ключ.

```
-- по първия начин
CREATE TABLE MovieStar(
        name CHAR(30) PRIMARY KEY,
        address VARCHAR(255),
        gender CHAR(1),
        birthdate DATETIME
);
-- по втория начин
CREATE TABLE MovieStar(
        name CHAR(30),
        address VARCHAR(255),
        gender CHAR(1),
        birthdate DATETIME,
        PRIMARY KEY (name)
);
```

Пример 2
Да разгледаме схемата Movie. Първичният ключ е съставен – състой се от два атрибута title и year

```
-- по първия начин
   Тъй като ключът е съставен този начин не може да се използва!
-- по втория начин
CREATE TABLE Movie(
        title nvarchar(50),
        year  int,
        length int,
        inColor bit,
        studioName nvarchar(50),
        producerC# int,
        PRIMARY KEY (title,year)
)
```

Ако не се зададе име на ограничението, СУБД му поставя служебно име. Име на ограничение се задава като поставим ключовата дума CONSTRAINT, последвана от името на ограничението, преди основната декларация на ограничеието.
Всички ограничения се съхраняват в речника на данните (служебните таблици на СУБД).
Пример 3

```
CREATE TABLE MovieStar(
        name CHAR(30) CONSTRAINT pk_ms PRIMARY KEY,
```

```
        address VARCHAR(255),
        gender CHAR(1),
        birthdate DATETIME
);
CREATE TABLE Movie(
        title nvarchar(50),
        year  int,
        length int,
        inColor bit,
        studioName nvarchar(50),
        producerC# int,
        CONSTRAINT pk_m PRIMARY KEY (title,year)
);
```

Ако в таблицата, която създаваме нямаме подходяща колона за PK, можем да създадем изкувствена колона, която брой редовете в таблицата и да я обявим за ключ (т.нар. surrogate key). За въвеждането на подходящи стойности (последователности от уникални номера) в такава колона различните СУБД предоставят различни средства. Например в MS SQL Server за целта може да се използва фунцията Identity, типа uniqueidentifier и др.

Деклариране на UNIQUE ключове

Друг начин да декларирате ключ е като използвате ключовата дума UNIQUE. Тя може да се появява там където може и PRIMARY KEY. Значението на UNIQUE декларацията е почти същото като това на PRIMARY KEY. Все пак има 2 разлики:

Можем да имаме повече от един UNIQUE ключ;

Докато PRIMARY KEY забранява NULL стойности в ключовите атрибути, то за UNIQUE ключовете няма такова ограничение. Правилото, че два котрежа не могат да си съответстват по отношение на всичките си атрибути декларирани в UNIQUE ключа се прилага и може да се наруши, дори и ако един или повече компонента имат NULL стойности Разрешено е обаче два атрибута да имат NULL по отношение на всичките си атрибути от UNIQUE ключа (приема се, че NULL не е равно на нищо).

Забележка: В MS SQL Server не е така, повече от един ред със null стойности по отошение на атрибутите влизащи в UNIQUE ключа се приема за нарушение на ограничението.

Пример 4

```
CREATE TABLE Movie(
        title nvarchar(50),
        year  int,
        length int,
        inColor bit,
        studioName nvarchar(50),
        producerC# int,
       CONSTRAINT uk_ms UNIQUE(title,year)
)
```

Ако имаме един или повече съставни UNIQUE ключовете, то трябва да използваме да ги дефинираме на ниво таблица (вторият подход).

Деклариране на външни ключове (FK)
Вторият важен вид ограничения върху базата данни е искането стойностите на даден атрибут да имат смисъл. Например атрибутът presC# от релацията Studio се очаква да сочи конкретен директор от MovieExec.
В SQL можем да да декларираме арибут/и от една релация (child table) да бъдат външен ключ и да сочат ("реферират") атрибути от втора релация (parent table) (допуска се първата и втората таблица да съвпадат)
Реферираните атрибути от втората релация трябва да бъдат декларирани като UNIQUE или PRIMARY KEY ограничения. В противен случай те не могат да участват във FK декларация.
Стойностите на FK, появяващи се в първата релация, трябва да се появяват като стойности на реферираните атрибути в кортежите на втората релация.
Имаме два начина за дефиниране на външен ключ чрез CREATE TABLE:
Ако външния ключ се състой от един атрибут може да добавим след името и типа му декларацията:

REFERENCES <parent_table> (<parent_table_attributes>)

Алтернативно можем да добавим към списъка от атрибути в CREATE TABLE една или повече декларации, указващи че множество от атрибути е външен ключ:

FOREIGN KEY (<child_table_attributes>) REFERENCES < parent_table> (<parent_table_attributes>)

Забележка: В MS SQL Server реферираните атрибути трябва да са PRIMARY KEY. UNIQUE колони не се допуска да бъдат реферирани

Пример 5

```
CREATE TABLE Studio (
        name CHAR(30) PRIMARY KEY,
        address VARCHAR(255),
        presC# INT  REFERENCES MovieExec(cert#)
);

CREATE TABLE Studio (
        name CHAR(30) PRIMARY KEY,
        address VARCHAR(255),
        presC# INT  CONSTRAINT fk_me REFERENCES MovieExec(cert#)
);

CREATE TABLE Studio (
        Name CHAR(30) PRIMARY KEY,
```

```
        Address VARCHAR(255),
        PresC# INT,
        FOREIGN KEY (PresC#)REFERENCES MovieExec(cert#)
);
CREATE TABLE Studio (
        name CHAR(30) PRIMARY KEY,
        address VARCHAR(255),
        presC# INT,
        CONSTRAINT fk_me  FOREIGN KEY (presC#)REFERENCES MovieExec(cert#)
);
```

Тъй като полето presC# допуска NULL стойности, разрешените стойности за него са NULL или реално съществуващ (регистриран в MovieExec) директор.

Политики за налагане на FK ограниченията
По подразбиране
Всяко действие, което нарушава референтна цялостност не се допуска от СУБД.
Възможни такива действия са:
Вмъкване на записи в child таблицата, което нарушава референтна цялостност;
Промяна на записи в child таблицата, което нарушава референтна цялостност;
Изтриване на записи в parent таблицата, което нарушава референтна цялостност;
Промяна на записи в parent таблицата, което нарушава референтна цялостност;

Каскадана
Това е политика по отношение на операциите по изтриване и промяна на записи в parent таблицата при изтриване на записи от parent таблицата, съответните "детайлни" записи от child таблицата, които биха нарушили ограничението за референта цялостност също се изтриват; при промяна на реферирано поле от parent таблицата, съответните полета на записи от child таблицата, също се променя;

Установяване на NULL стойности
Това е политика по отношение на операциите по изтриване и промяна на записи в parent таблицата, при условие че FK колоните от child таблицата допускат NULL стойности.
- при изтриване или промяна на записи от parent таблицата, които водят до нарушават референтната цялостност, съответните колони на записите от child таблицата се установяват на NULL.

Пример 6.

```
CREATE TABLE Studio (
        Name CHAR(30) PRIMARY KEY,
        Address VARCHAR(255),
        PresC# INT,
       CONSTRAINT fk_me   FOREIGN KEY (PresC#)REFERENCES MovieExec(cert#)
                          ON DELETE CASCADE
                          ON UPDATE SET NULL
```

);

Забележка: В MS SQL Server политиката “Установяване на NULL стойности” не се поддържа!

[ON DELETE { CASCADE | NO ACTION }]
[ON UPDATE { CASCADE | NO ACTION }]

Ограничения върху атрибутите и кортежите
Not null ограничения
Едно просто ограничение е недопускането даден атрибут да има NULL стойности. Ограничението се декларира като се постави ключовата дума NOT NULL след декларацията на артрибута в CREATE TABLE.

Пример 7
....
PresC# INT REFERENCES MovieExec(cert#) NOT NULL

CHECK ограничения на ниво атрибут
По сложни ограничения могат да бъдат добавени към декларацията на атрибут чрез кл.дума CHECK, последвана от оградено в скоби условие, което трябва да бъде изпълнено за всяка стойност на арибута. Обиковено с атрибут-базирано CHECK ограничение се задават допустимите стойности (домейна) за атрибута. Принципно, условето може да всяко условие, което може да се постави в WHERE клаузата на една SQL заявка. Все пак различните СУБД обикновено имат допълнителни рестрикции върху това, какво може да включва CHECK условието.
Атрибут-базираното CHECK ограничение се проверява всеки път, когато коретеж получава нова стойност за съответния атрибут (в резултат на insert или update).Ако ограничението се нарушава от новата стойност, то действието се отхвърля (завършва с неуспех).

Пример 8
....
PresC# INT REFERENCES MovieExec(cert#) CHECK (presC#>=100000)

Пример 9
…
gender CHAR(1) CHECK (gender in (‘F’,’M’))

CHECK ограничения на ниво таблица
За да декларираме ограничение върху кортежите на една релация R, когато я дефинираме със CREATE TABLE, трябва да добавим към списъка с атрибути и декларации на ключове, кл. дума CHECK, последвана от оградено в скоби условие. То се интерпретира като условие върху записите на таблицата R и в него могат да участват колони от R (Принципно може да е произволно WHERE условие).

CHECK ограничение дефинирано на ниво таблица се провярява всеки път, когато се въвежда или променя запис в таблицата.

Пример 10

```
CREATE TABLE MovieStar (
        name CHAR(30),
        address VARCHAR(255),
        gender CHAR(1),
        birthdate DATETIME,
CHECK (gender =’F’ OR name NOT LIKE ‘Ms.%’)
);
```

или

```
...
CONSTRAINT ch_ms CHECK (gender ='F' OR name NOT LIKE 'Ms.%');
```

МОДИФИКАЦИИ НА ОГРАНИЧЕНИЯ

SQL предоставя възможности за изтриване и добавяне на ограничения върху вече дефинирана релационна схема. За да изтрием дадено ограничение е необходимо да занем неговото име.

Изтриване на ограничение

```
ALTER TABLE table_name
DROP CONSTRAINT constriant_name;

ALTER TABLE table_name
DROP CONSTRAINT c1,c2,...;
```

Пример 11

```
 ALTER TABLE Studio
 DROP CONSTRAINT fk_me;
```

Добавяне на ограничение

```
ALTER TABLE table_name
ADD  CONSTRAINT constriant_name <table constraint declaration>;
```

Пример 12

```
ALTER TABLE Ships
ADD CONSTRAINT pk PRIMARY KEY (name);
```

В MS SQL Server:

```
ALTER TABLE table
ADD { < table_constraint > } [ ,...n ] ,  където
```

```
table_constraint > ::=
   [ CONSTRAINT constraint_name ]
   { [ { PRIMARY KEY | UNIQUE }
      { ( column [ ,...n ] ) }
   ]
      |   FOREIGN KEY
         [ ( column [ ,...n ] ) ]
         REFERENCES ref_table [ ( ref_column [ ,...n ] ) ]
         [ ON DELETE { CASCADE | NO ACTION } ]
         [ ON UPDATE { CASCADE | NO ACTION } ]
      | DEFAULT constant_expression
         [ FOR column ]
      | CHECK  ( search_conditions )
   }
```

Промяна на NOT NULL ограничение

Тъй като NOT NULL ограничение се декларира само на ниво колона, промяната му в MS SQL Server може се осъществи чрез промяна на дефиницята на съответната колона:

```
ALTER TABLE table
 ALTER COLUMN column_name
   { new_data_type [ ( precision [ , scale ] ) ]
     [ NULL | NOT NULL ] }
```

Пример 13

```
ALTER TABLE Ships
ALTER COLUMN  class nvarchar(40) NULL;

ALTER TABLE Ships
ALTER COLUMN  class nvarchar(40) NOT NULL;
```

Обобщение

Ограниченията могат да бъдат създавани:

- По време на създаването на таблица;
- След като таблицата бъде създадена;
- Ограничение може да се дефинира на ниво колона или на ниво таблица;

Всички ограничения се съхраняват в речника на данните (служебните таблици на СУБД) СУБД използват ограничения, за да се предпазят от наличието на невалидни данни в базата. Ако не се зададе име, СУБД поставя служебно име на ограничението

Следните типове ограничения са валидни:

- NOT NULL – непозволяма колона да съдържа NULL стойности
- UNIQUE – определя колона или комбинация от колони, чиито стойности трябва да бъдат уникални за всички редове в таблицата
- PRIMARY KEY – уникално идентифицира всеки ред в таблицата

- FOREIGN KEY –установява и налага връзка между колони от две таблици
- CHECK – определя условие, което трябва да бъде винаги истина
- На ниво таблица не може да се задава NOT NULL ограничение

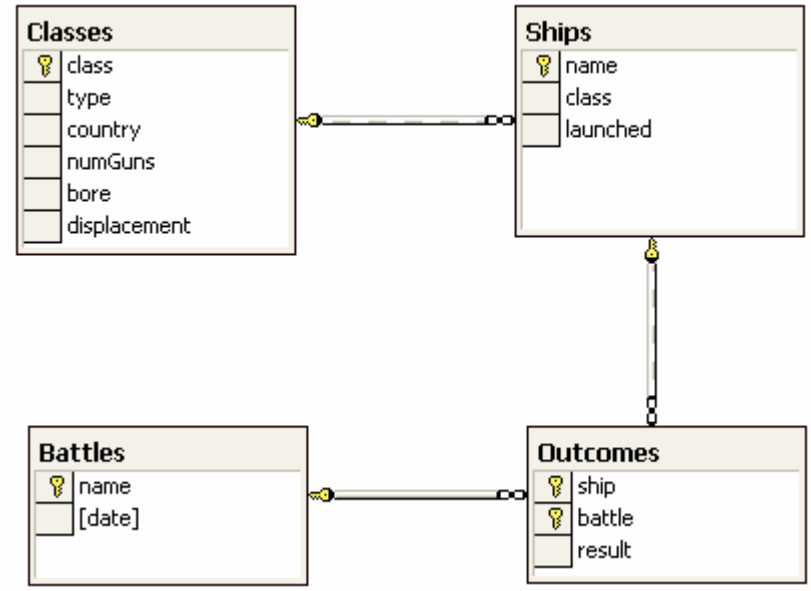
CREATE TABLE table
(column datatype [DEFAULT expr]
[column_constraint],
…
[table_constraint] [,…]);

column_constraint
column [CONSTRAINT constraint_name] constraint_type,

table_constraint
column,…
[CONSTRAINT constraint_name] constraint_type (column,…),

Задачи

1. Създайте базата от данни Ships, като следвате диаграмата:



2. Предложете и създайте подходящи първични ключове за базата от данни Ships.

Classes(class,type,country, numGuns,bore, displacement)
Ships(name,class,launched)
Battles(name, date)
Outcomes(ship,battle,result)

Добавете първични ключове чрез оператора ALTER TABLE.
Напишете следните ограничения за референтна цялостност върху базата Ships
Всеки клас споменат в Ships, трябва да бъде споменат и в Classes.
Всяка битка споменат в Outcomes, трябва да бъде споменат и в Battles.

Всеки кораб споменат в Outcomes, трябва да бъде споменат и в Ships.

3. Вмъкнете следните данни в таблиците:

Classes

```
insert into classes values('Bismarck','bb','Germany',8,15,42000)
insert into classes values('Iowa','bb','USA',9,16,46000)
insert into classes values('Kongo','bc','Japan',8,14,32000)
insert into classes values('North Carolina','bb','USA',12,16,37000)
insert into classes values('Renown','bc','Gt.Britain',6,15,32000)
insert into classes values('Revenge','bb','Gt.Britain',8,15,29000)
insert into classes values('Tennessee','bb','USA',12,14,32000)
insert into classes values('Yamato','bb','Japan',9,18,65000)
GO
```

Battles

```
insert into battles values('Guadalcanal','19421115 00:00:00.000')
insert into battles values('North Atlantic','19410525 00:00:00.000')
insert into battles values('North Cape','19431226 00:00:00.000')
insert into battles values('Surigao Strait','19441025 00:00:00.000')
GO
```

Ships

```
insert into ships values('California','Tennessee',1921)
insert into ships values('Haruna','Kongo',1916)
insert into ships values('Hiei','Kongo',1914)
insert into ships values('Iowa','Iowa',1943)
insert into ships values('Kirishima','Kongo',1915)
insert into ships values('Kongo','Kongo',1913)
insert into ships values('Missouri','Iowa',1944)
insert into ships values('Musashi','Yamato',1942)
insert into ships values('New Jersey','Iowa',1943)
insert into ships values('North Carolina','North Carolina',1941)
insert into ships values('Ramillies','Revenge',1917)
insert into ships values('Renown','Renown',1916)
insert into ships values('Repulse','Renown',1916)
insert into ships values('Resolution','Renown',1916)
insert into ships values('Revenge','Revenge',1916)
insert into ships values('Royal Oak','Revenge',1916)
insert into ships values('Royal Sovereign','Revenge',1916)
insert into ships values('Tennessee','Tennessee',1920)
insert into ships values('Washington','North Carolina',1941)
insert into ships values('Wisconsin','Iowa',1944)
insert into ships values('Yamato','Yamato',1941)
insert into ships values('South Dakota','North Carolina',1941)
GO
```

Outcomes

```
insert into outcomes values('California','Surigao Strait','ok')
insert into outcomes values('King George V','North Atlantic','ok')
```

```
insert into outcomes values('Kirishima','Guadalcanal','sunk')
insert into outcomes values('South Dakota','Guadalcanal','damaged')
insert into outcomes values('Tennessee','Surigao Strait','ok')
insert into outcomes values('Washington','Guadalcanal','ok')
insert into outcomes values('California','Guadalcanal','damaged')
```

4. Извършете следните модификации в базата от данни Ships:
   - За корабите от класа Kongo въведете следните факти в базата от данни: пуснати на вода през 1927 година, имат девет 16-инчови оръдия и водоизместимост от 34 000 тона.