Дефиниране на схеми на релации. Ограничения.

Дефиниране на релационна схема

- Атрибутите имат типове:
  - CHAR(n) – низ с фиксирана дължина
  - VARCHAR(n) – низ с дължина до n символа
  - INT / INTEGER – цяло число, 32-bit, със знак
  - FLOAT / REAL, DOUBLE
  - DECIMAL(n,d) – дробно число с n цифри, d от тях след десетичната запетая (има разлика от FLOAT)
  - DATE, TIME
  - И други, някои са специфични за конкретната СУБД

Създаване на релация

```
CREATE TABLE MovieStar (
    name CHAR(30),
    address VARCHAR(255),
    gender CHAR(1),
    birthdate DATE
  );
```

- Стойности по подразбиране

```
CREATE TABLE MovieStar (
    ...
gender CHAR(1) DEFAULT '?',
regdate DATE DEFAULT CURRENT_DATE);
```

Изтриване на релация

- Изтриване на релация:

```
DROP TABLE R;
```

- Добавяне на атрибут в релация:

```
ALTER TABLE Moviestar
ADD phone CHAR(16);
```

- Изтриване на атрибут:

```
ALTER TABLE Moviestar
DROP birthdate;
```

Ограничения

- Първичен ключ
- UNIQUE ограничения
- Външен ключ
- Други ограничения
  - NOT NULL ограничения
  - CHECK ограничения на ниво атрибут
  - CHECK ограничения на ниво таблица
- Извършване на проверката
  -
- Модификация на ограничения

Първичен ключ (PRIMARY KEY)

- Ако множеството от атрибути S е първичен ключ на релацията R, то всеки два кортежа от R, трябва да се различават по отношение на стойността на поне един атрибут от множеството S

- Стойностите в една колона, декларирана като първичен ключ са уникални и не могат да бъдат NULL

- Една релация може да има само едни първичен ключ.

Деклариране на първичен ключ върху един атрибут

- Декларация на ниво атрибут

```
CREATE TABLE MovieStar(
  name CHAR(30) PRIMARY KEY,
  address VARCHAR(255),
  gender CHAR(1),
  birthdate DATE
);
```

На ограничението се поставя служебно име.

Деклариране на първичен ключ върху един атрибут (cont.)

- Декларация на ниво таблица

```
CREATE TABLE MovieStar(
  name CHAR(30),
  address VARCHAR(255),
  gender CHAR(1),
  birthdate DATE ,
  PRIMARY KEY (name)
);
```

!!! На ограничението се поставя служебно име.

Деклариране и именуване на първичен ключ върху един атрибут

- Декларация на ниво атрибут

```
CREATE TABLE MovieStar(
  name CHAR(30) CONSTRAINT pk_ms
  PRIMARY KEY,
  address VARCHAR(255),
  gender CHAR(1),
  birthdate DATE
);
```

Деклариране и именуване на първичен ключ върху един атрибут

- Декларация на ниво таблица

```
CREATE TABLE MovieStar(
  name CHAR(30),
  address VARCHAR(255),
  gender CHAR(1),
  birthdate DATE,
  CONSTRAINT pk_ms PRIMARY
  KEY(name)
);
```

Деклариране на съставен първичен ключ

- Възможно е САМО на ниво таблица.

```
CREATE TABLE Movie(
  title varchar(50),
 year  integer,
  length integer,
  inColor char(1),
  studioName varchar(50),
  producerC# integer,
                PRIMARY KEY (title,year)
);
```

Деклариране и именуване на съставен първичен ключ

```
CREATE TABLE Movie(
  title varchar(50),
  year  integer,
  length integer,
  inColor char(1),
  studioName varchar(50),
  producerC# integer ,
    CONSTRAINT pk_m PRIMARY KEY
  (title,year));
```

Деклариране на първичен ключ след създаване на таблицата

```
ALTER TABLE Movie
ADD [ CONSTRAINT Movie_pk ]
  PRIMARY KEY (title,year)

ALTER TABLE MovieStar
ADD [ CONSTRAINT MovieStar_pk ]
  PRIMARY KEY (name)
```

Сурогатен ключ

- Изкуствено създаден първичен ключ, когато нямаме подходяща колона с уникални стойности.
- Пример: автоматично номериране на кортежите
- Синтаксис в DB2

```
GENERATED (ALWAYS|BY DEFAULT) AS
  IDENTITY
       (START WITH -1,
        INCREMENT BY 1,
        CYCLE,
        MINVALUE -3,
        MAXVALUE 3)
```

Ограничение за уникалност (UNIQUE)

- Ограничението UNIQUE също като PRIMARY KEY гарантира уникалност на стойностите в колоните, върху които е поставено.

- За разлика от PRIMARY KEY:
  - Можем да имаме повече от 1 UNIQUE ограничение в една таблица.
  - Ограничението UNIQUE допуска NULL стойности в ключовите атрибути (включително и NULL във всички атрибути на ключа)

Деклариране на UNIQUE ограничение

- Аналогично на PRIMARY KEY

```
CREATE TABLE Movie(
  title varchar(50),
  year  integer,
  length integer,
  inColor char(1),
  studioName varchar(50),
  producerC# integer,
    CONSTRAINT movie_uk
  UNIQUE(title,year));
```

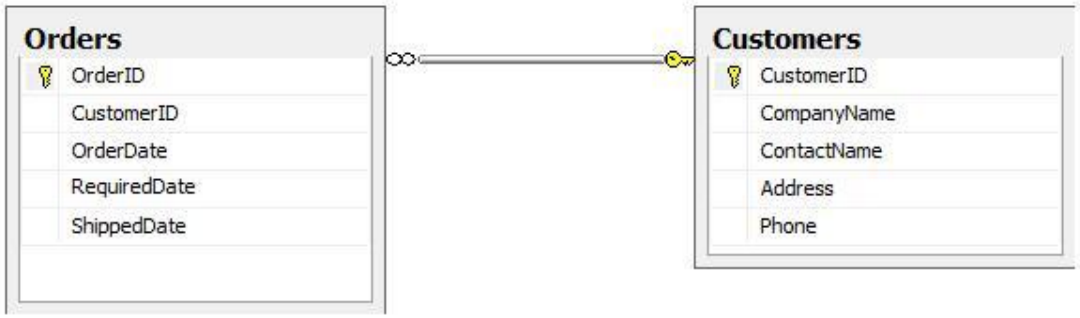
Външен ключ (FOREIGN KEY)

- Ограничение за референтна цялостност
- Гарантира, че стойностите на атрибутите от релацията, в която е дефиниран външния ключ, се срещат в съответните атрибути от релацията, която се реферира от външния ключ.
- Реферираните атрибути от втората релация трябва да бъдат декларирани като UNIQUE или PRIMARY KEY.

Външен ключ - пример

- Връзка Master – Detail (Parent – Child)
- FOREIGN KEY се декларира в таблицата дете/detail
- Съответните атрибути от Master и Detail таблицата трябва да са от един и същи тип

Detail / Child Master / Parent



Деклариране на външен ключ

- Декларация на ниво атрибут

```
REFERENCES  <parent_table>
  (<parent_table_attribute>)
```

- На ниво таблица

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

Деклариране на външен ключ върху един атрибут

- Декларация на ниво атрибут

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

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

Деклариране на външен ключ върху един атрибут (cont.)

- Декларация на ниво таблица

```
CREATE TABLE Studio (
  Name CHAR(30) PRIMARY KEY,
  Address VARCHAR(255),
  PresC# INTEGER  ,
            FOREIGN KEY (PresC#)
  REFERENCES MovieExec(cert#)
);
```

Деклариране и именуване на външен ключ върху един атрибут

- Декларация на ниво атрибут

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

Деклариране и именуване на външен ключ върху един атрибут (cont.)

- Декларация на ниво таблица

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

Деклариране на съставен външен ключ

- САМО на ниво таблица

```
CREATE TABLE StarsIn(
  movietitle varchar(50),
  movieyear integer,
  name char(30),
   FOREIGN KEY (movietitle,
  movieyear) REFERENCES
  Movie(title, year)
);
```

Деклариране и именуване на съставен външен ключ

```
CREATE TABLE StarsIn(
  movietitle varchar(50),
  movieyear integer,
  starname char(30),
   CONSTRAINT Movie_FK
  FOREIGN KEY (movietitle,
  movieyear) REFERENCES
  Movie(title, year)
);
```

Деклариране на външен ключ след създаване на таблицата

```
ALATER TABLE StarsIn
ADD [ CONSTRAINT MovieStar_FK ]
  FOREIGN KEY (starname)
  REFERENCES MovieStar(name);

ALATER TABLE StarsIn
ADD [ CONSTRAINT MovieStar_FK ]
  FOREIGN KEY (movietitle,
  movieyear)
  REFERENCES Movie(title, year);
```

Политики за налагане на FK ограниченията

- NO ACTION/ RESTRICT
  - Всяко действие, което нарушава референтна цялостност не се допуска от СУБД
- CASCADE
  - Това е политика по отношение на операциите по изтриване и промяна на редове в parent таблицата
- SET NULL
  - Това е политика по отношение на операциите по изтриване и промяна на редове в parent таблицата, при условие че FK колоните от child таблицата допускат NULL стойности.

Политики за налагане на FK ограниченията

- Синтаксис

```
FOREIGN KEY
  (<child_table_attributes>)
  REFERENCES  <parent_table>
  (<parent_table_attributes>)
[ ON DELETE { CASCADE | NO
  ACTION | SET NULL} ]
[ ON UPDATE { CASCADE | NO
  ACTION | SET NULL } ]
```

Политики за налагане на FK ограниченията – пример

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

Други ограничения NOT NULL

- Не се допуска даден атрибут да приема NULL стойности.

- Декларира се САМО на ниво атрибут с ключовите думи NOT NULL

```
PresC# INTEGER
REFERENCES MovieExec(cert#) NOT NULL
```

- След като таблицата е създадена, се декларира като промяна на дефиницията на колоната с:

```
ALTER TABLE table1
ALTER COLUMN col1 col1_type NOT NULL
```

Други ограничения CHECK на ниво атрибут

- Синтаксис

```
attr1 attr1_type CHECK
  (condition_on_attr1)
```

- Може да се използва всяко условие, което може да се постави в WHERE клаузата на една SQL заявка.
- Проверява се всеки път, когато кортежът получава нова стойност за съответния атрибут (в резултат на insert или update)

```
gender CHAR(1) CHECK (gender in
  ('F','M'))
```

Други ограничения CHECK на ниво таблица

- ограничение върху кортежите на една релация
- Синтаксис:

CHECK (condition_on_some_attributes**)**

- Добавя се към списъка с атрибути и декларации на ограничения в CREATE.
- Проверява се всеки път, когато се въвежда или променя ред от таблицата.

Други ограничения
CHECK на ниво таблица – пример

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

ИЛИ

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

Извършване на проверката

- Спазването на ограниченията се проверява при всеки inset, update, delete или drop, т.е. за всеки израз.
- SQL предоставя възможности за отлагане на проверката на едно ограничение в края на всяка транзакция, а не при изпълнението на конкретен оператор.

Модификация на ограничения

- Изтриване на ограничение

```
ALTER TABLE table_name
DROP CONSTRAINT
  constraint_name;
```

- Ако искаме да изтрием ограничение, трябва да знаем неговото име (Името дадено му автоматично от СУБД или името, което е дадено изрично при създаването на ограничението)

Модификация на ограничения (cont.)

- Добавяне на ограничение (след като вече сме декларирали схемата на таблицата)

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

Обобщение

- Валидни са следните типове ограничения:
  - NOT NULL – непозволява колона да съдържа NULL стойности
  - UNIQUE – декларира, че стойностите в колона или група колони трябва да бъдат уникални за всички редове в таблицата
  - PRIMARY KEY – уникално идентифицира всеки ред в таблицата
  - FOREIGN KEY – установява и налага връзка за референтна цялостност между колони или група колони от две таблици
  - CHECK – определя условие, което трябва да бъде винаги истина