6 Data Definition Language (DDL)

ЗАДАЧА 1. Да се дефинираат правилата на интегритет за примарните и надворешните клучеви во следниве релации:

SNABDUVACI (s#, ime_s, saldo, grad)
PROIZVODI (proiz#, vid#, ime_p, boja, tezina, grad_p)
PONUDI (p#, s#*, pr#*, v#*, kolicina_nar, datum_nar, kolicina_isp, datum_isp)

Решение:

Правилата за интегритет на примарните клучеви наложуваат вредностите на примарните клучеви за секоја торка (запис) да не бидат празни (непополнети - **NULL**) и да не се повторуваат во рамките на таа релација (да нема дупликати).
Правилата за интегритет на надворешните клучеви наложуваат вредностите што ги добиваат надворешните клучеви да мора да постојат во релациите каде што се наоѓаат примарните клучеви или да бидат непополнети.
Првото правило е јасно бидејќи примарниот клуч ЕДИНСТВЕНО треба да го идентификува записот, и ако го оставиме непополнет примарниот клуч, нема да може да пристапуваме правилно до тој запис.
Второто правило ќе го илустрираме со една замислена ситуација. Замислете дека треба да направите програма во која на екран ќе треба да ги прикажете нарачките за даден производ, но притоа, наместо да го прикажувате бројот на испорачателот, клиентот од вас бара да се прикажува името на испорачателот во истиот ред со податоците за одредена нарачка:

IME_S	KOLICINA_NAR	DATUM_NAR	KOLICINA_ISP	DATUM_ISP

Се гледа дека KOLICINA_NAR, DATUM_NAR, KOLICINA_ISP, DATUM_ISP треба да се читаат од табелата PONUDI, а IME_S од табелата SNABDUVACI, така што ќе се пронајде записот со соодветниот s#.

Сега замислете дека во табелата PONUDI постои следниов запис:
(4, 7, 100, 10.5.2001, 100, 13.5.2001), што се однесува за снабдувачот број 4, а во табелата SNABDUVACI не постои запис за снабдувачот број 4.
Тогаш во моментот кога треба да се прикажува на екранот горе споменатиот запис, ќе се појави грешка во колоната за IME_S, бидејќи таков запис со S# = 4 во табелата SNABDUVACI **не постои**. За да се спречи таквата неконзистентност (неповрзаност) на податоците, се воведуваат правилата на интегритет за надворешните клучеви (**referential integrity**) кои што се применуваат секогаш кога настанува некаква промена на податоците.
Можните промени на записите се: додавање на нов запис, бришење на запис и ажурирање на запис. Вообичаено СУБП нудат можност за реагирање на одредени **НАСТАНИ** и тоа **пред**, **по**, или **за време на** самиот настан.

Оттука дефинициите кога едно правило се извршува (на англиски) се:

BEFORE INSERTING,	ON INSERTING,	AFTER INSERTING,
BEFORE DELETING,	ON DELETING,	AFTER DELETING,
BEFORE UPDATING,	ON UPDATING, или	AFTER UPDATING.

Процедурите (во кои се дефинирани правилата) кои се извршуваат на одредени настани се викаат **ТРИГЕРИ**. Покрај со тригери, интегритетот на податоците може да се дефинира со самата дефиниција на табелите (со CREATE TABLE наредбата), или подоцна (со ALTER TABLE) наредбата.

```
CREATE TABLE snabduvaci (
s# NUMBER(3) PRIMARY KEY,
ime_s VARCHAR(50),
saldo NUMBER,
grad VARCHAR(20)
);

CREATE TABLE proizvodi (
proiz# NUMBER(5),
vid# NUMBER(2),
ime_p VARCHAR(50),
boja CHAR(5),
tezina NUMBER(3),
grad_p VARCHAR(20),
CONSTRAINT proizvodi_PK PRIMARY KEY(proiz#, vid#)
);

CREATE TABLE ponudi (
p# NUMBER(5) PRIMARY KEY,
s# NUMBER(3) REFERENCES snabduvaci(s#),
pr# NUMBER(5),
v# NUMBER(2),
kolicina_nar NUMBER,
datum_nar DATE,
kolicina_isp NUMBER,
datum_isp DATE,
CONSTRAINT ponudi_FK FOREIGN KEY (pr#, v#)
REFERENCES proizvodi(proiz#,vid#)
);
```

Со вака дефинираните ограничувања, не може да се внесе запис во зависната табела (PONUDI во нашиот случај) ако не постои соодветен запис во главната табела (SNABDUVACI и PROIZVODI).
Ако не се наведе поинаку, бришењето и промената на запис од главната табела не се дозволени, во случај кога има записи во зависната табела кои се однесуваат на записот кој треба да се избрише. На пример во табелата понуди постои запис за снабдувачот број 17, а од табелата SNABDUVACI сакаме да го избришиме баш записот за снабдувач број 17.

Но со користење на дополнителни наредби може да се овозможи:
- полето кое е надворешен клуч да се постави на празна вредност (SET NULL)
- записите каскадно да се избришат или променат (ажурираат) (CASCADE)
- полето кое е надворешен клуч да се постави на предодредена вредност (SET DEFAULT)

Дали одредени надворешни клучеви ќе се поставуваат на NULL или предодредена вредност, или ќе се бришат каскадно зависи од дефиницијата на ограничувањето (CONSTRAINT).

На пример третиот SQL израз може да изгледа вака:

```
CREATE TABLE ponudi (
p# NUMBER(5) PRIMARY KEY,
s# NUMBER(3) REFERENCES snabduvaci(s#),
        ON DELETE CASCADE      ON UPDATE CASCADE,
pr# NUMBER(5),
v# NUMBER(2),
kolicina_nar NUMBER,
datum_nar DATE,
kolicina_isp NUMBER,
datum_isp DATE,
CONSTRAINT proiz_FK FOREIGN KEY (pr#) REFERENCES proizvodi(proiz#)
        ON DELETE SET NULL      ON UPDATE CASCADE,
CONSTRAINT vid_FK FOREIGN KEY (v#) REFERENCES proizvodi(vid#)
        ON DELETE SET NULL      ON UPDATE CASCADE
);
```

Со ова при промена на шифрата на снабдувачот ќе се променат (ажурираат) сите места каде што се јавува таа шифра на снабдувач во табелата понуди. При бришење на некој снабдувач каскадно ќе се избришат сите понуди од тој снабдувач.
При промена на proiz# или vid# во табелата PROIZVODI каскадно се ажурираат вредностите на атрибутите pr# и v#, додека при бришење на запис од табелата PROIZVODI вредноста на атрибутите pr# и v# кои се однесуваат на записот кој се брише се поставуваат на вредност NULL.
Одлуката дали да се забрани бришење или промена, дали да се постават надворешните клучеви на NULL или предодредена вредност, или пак каскадно да се избришат или променат зависи од деловната политика на фирмата за која се прави базата на податоци. На пример во фирмата сакаат и старите понуди за производи кои веќе не постојат или кои веќе не се нудат, сепак да стојат во базата за евентуални споредби - и затоа е ставено ON DELETE SET NULL. Додека промената на proiz# и vid# на еден производ е дозволено да се пренесе во сите табели каскадно. Забележете дека и ова во некои случаи може да не е добро решение, бидејќи може во фирмата да имаат и многу печатени документи поврзани со шифрите на производот кои потоа би биле тешки за наоѓање или дури погрешно би биле интерпретирани како да се однесуваат за некој друг производ. Затоа во таков случај можеби позгодно решение би било да не се дозволи промена на вредноста.

ЗАДАЧА 2. За релациите од задача 1, да се дефинираат правила на интегритет што ќе ги задоволат следните услови):

- имињата на снабдувачите и производителите не смеат да бидат празни;
- имињатана снабдувачите се единствени (различни);
- салдото да има вредност поголема од нула;
- доколку не се внесе градот на снабдувачот, тогаш да се додели предодредена вредност Скопје;
- градот каде што се прават производите да биде еден од Лондон, Париз, Рим;
- испорачаната количина да не е поголема од нарачаната.

Решение:

```
CREATE TABLE snabduvaci (
s# NUMBER(3) PRIMARY KEY,
ime_s VARCHAR(50) NOT NULL,
saldo NUMBER CHECK (saldo > 0),
grad VARCHAR(20) DEFAULT ‘Skopje’,
CONSTRAINT ime_s_unique UNIQUE(ime_s)
);

CREATE TABLE proizvodi (
proiz# NUMBER(5),
vid# NUMBER(2),
ime_p VARCHAR(50) NOT NULL,
boja CHAR(5),
tezina NUMBER(3),
grad_p VARCHAR(20) CHECK (grad_p IN ( 'London', 'Pariz', 'Rim')),
CONSTRAINT proizvodi_PK PRIMARY KEY(proiz#, vid#)
);

CREATE TABLE ponudi (
p# NUMBER(5) PRIMARY KEY,
s# NUMBER(3) REFERENCES snabduvaci(s#),
        ON DELETE CASCADE      ON UPDATE CASCADE,
pr# NUMBER(5),
v# NUMBER(2),
kolicina_nar NUMBER,
datum_nar DATE,
kolicina_isp NUMBER,
datum_isp DATE,
CHECK (kolicina_isp <= kolicina_nar),
CONSTRAINT proiz_FK FOREIGN KEY (pr#) REFERENCES proizvodi(proiz#)
        ON DELETE SET NULL      ON UPDATE CASCADE,
CONSTRAINT vid_FK FOREIGN KEY (v#) REFERENCES proizvodi(vid#)
        ON DELETE SET NULL      ON UPDATE CASCADE
);
```

Во табелата PONUDI, проверката се однесува на ниво на запис, бидејќи во неа се употребуваат вредности од два атрибута, па затоа се пишува по дефинирањето на последниот атрибут.

Доколку дадено ограничување се повторува повеќе пати, може да се креира домен за даден тип на вредности, па потоа соодветните атрибути да се постават да бидат од тој домен. Така ќе дефинираме домен за салдото на снабдувачите pozitiven кој ќе ни го опфаќа множеството на позитивни броеви, и потоа при креирање на табелата SNABDUVACI атрибутот saldo ќе го поставиме да биде pozitiven.

```
CREATE DOMAIN pozitiven AS NUMBER CHECK (pozitiven>0);

CREATE TABLE snabduvaci (
s# NUMBER(3) PRIMARY KEY,
ime_s VARCHAR(50) NOT NULL,
saldo pozitiven,
grad VARCHAR(20) DEFAULT ‘Skopje’,
CONSTRAINT ime_s_unique UNIQUE(ime_s)
);
```