ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ
И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

ВОВЕД ВО SQL:

Дефиниција на шема
Data Definition Language
Едноставни SQL прашања над една табела

БАЗИ НА ПОДАТОЦИ - предавања

Вон. проф. д-р Кире Триводалиев

За SQL

- Името **SQL** означува структуриран прашалнички јазик (Structured Query Language). Првично SQL бил наречен SEQUEL (Structured English QUEry Language)
- SQL е стандарден јазик за комерцијални релациони СУБП
  - SQL (ANSI 1986), наречен SQL-86 или SQL1.
  - Ревидиран и подобрен наречен SQL2 (познат и како SQL-92)
  - SQL-99
- SQL е разбирлив јазик за бази на податоци:
  содржи изрази за дефинирање на податоци, за поставување прашања и за ажурирање на податоците

SQL: DDL и DML

- SQL е и
  - **DDL** (Data Definition Language)
    јазик за дефинирање на податоци и
  - **DML** (Data Manipulation Language)
    јазик за манипулирање со податоци

- Дополнително, има олеснување за:
  - дефинирање на погледи врз базата за специфицирање на заштита и овластување
  - дефинирање на интегритетни ограничувања и
  - специфицирање на трансакциски контроли
  - Исто така има правила за вметнување на SQL изрази во програмски јазици со општа намена како што се Java, Cobol, или C/C++.

Дефиниција на податоци - DDL

- Овие наредби се употребуваат за да ги:
  - создадат (CREATE),
  - отстранат (DROP)
  - и да ги променат (ALTER)

  описите на табелите (релациите) во базата на податоци

Секоја БП мора физички да биде креирана во физичката меморија за да може понатаму да се користи!

CREATE SCHEMA

- Одредува нова шема на базата на податоци со тоа што ѝ дава име
- Алтернативно, на шемата може да ѝ биде доделено име и овластувачки идентификатор, а елементите можат да бидат дефинирани подоцна.
- На пример, следната наредба создава шема наречена KOMPANIJA, која ја поседува корисник со овластувачки идентификатор 'Jsmith’.

```
CREATE SCHEMA KOMPANIJA AUTHORIZATION Jsmith;
```

CREATE TABLE

- Креира нова табела (релација) во базата со тоа што ѝ дава име и ги одредува сите нејзини атрибути заедно со нивните податочни типови

```
CREATE TABLE name
( Lista na atributi );
```

- Предефинирани податочни типови:
  - Нумерички:
    - INTEGER, INT, SMALLINT
    - FLOAT, REAL, DOUBLE PRECISION
    - DECIMAL(i,j), NUMERIC(i,j)
  - Карактери-стрингови:
    - CHAR(n), CHARACTER(n)
    - VARCHAR(n), CHAR VARYING(n)
  - Бит-стринг
    - BIT(n), BIT VARYING(n)
    - BLOB, CLOB
  - Логички
    - BOOLEAN (True, False, Unknown)
  - Датум

CREATE TABLE (2)

- Пример наредба за креирање на табелата која произлегува од релацијата **Department** од примерот КОМПАНИЈА:

```
CREATE TABLE DEPARTMENT (
   DNUMBER          INTEGER,
   DNAME            VARCHAR(10),
   MGRSSN           CHAR(9),
   MGRSTARTDATE     DATE   );
```

CREATE TABLE (3)

- CREATE TABLE наредбата може да се користи и за:
  - да се одредат атрибутите на примарниот клуч **PRIMARY KEY**
  - единственоста на некои атрибути **UNIQUE**
  - одредување на задолжителноста на некој атрибут **NOT NULL**
  - како и ограничувањата на референтниот интегритет (надворешните клучеви) **FOREIGN KEY** + **REFERENCES**
    - RESTRICT / CASCADE / SET NULL / SET DEFAULT

```
CREATE TABLE DEPT (
    DNAME         VARCHAR(10)        NOT NULL,
    DNUMBER       INTEGER            NOT NULL,
    MGRSSN        CHAR(9),
    MGRSTARTDATE  CHAR(9),
    PRIMARY KEY (DNUMBER),
    UNIQUE (DNAME),
    FOREIGN KEY (MGRSSN) REFERENCES EMP(SSN)   );
```

Опции за референцијален интегритет

- Може да одредиме RESTRICT, CASCADE, SET NULL или SET DEFAULT кај надворешните клучеви

```
CREATE TABLE DEPT (
   DNAME              VARCHAR(10)      NOT NULL,
   DNUMBER            INTEGER          NOT NULL,
   MGRSSN             CHAR(9),
   MGRSTARTDATE       CHAR(9),
   PRIMARY KEY (DNUMBER),
   UNIQUE (DNAME),
   FOREIGN KEY (MGRSSN) REFERENCES EMP(SSN)
      ON DELETE SET DEFAULT
      ON UPDATE CASCADE  );
```

Опции за референцијален интегритет (2)

```
CREATE TABLE EMP(
    ENAME          VARCHAR(30) NOT NULL,
    ESSN           CHAR(9),
    BDATE          DATE,
    DNO            INTEGER,
    SUPERSSN       CHAR(9),
    PRIMARY KEY (ESSN),
    FOREIGN KEY (DNO) REFERENCES DEPT(DNO)
          ON DELETE SET DEFAULT
          ON UPDATE  CASCADE,
    FOREIGN KEY (SUPERSSN) REFERENCES EMP(SSN)
          ON DELETE SET NULL
          ON UPDATE CASCADE    );
```

Напомена

- Внимавајте кои табели треба да бидат креирани, т.е. кои табели се дел од дадено **ентитетно множество**
  - Мултивредносен атрибут на ентитет или релација
- Внимавајте кога кај надворешен клуч не може да се оди со опцијата **on delete set null**:
  - Наследување
  - Слаб ентитет
  - Мултивредносен атрибут

т.е. не може за атрибут што е (дел од) ПК

- Ако не треба да ги чуваме нарачките за корисниците кои се избришани од базата на податоци
  - Пример: on delete cascade за korisnikId

Ограничување на атрибутите

- Доколку атрибутот има подразбирлива вредност, тогаш истата може да се специфицира со **DEFAULT**
- За секој атрибут или група на атрибути може да се дефинираат ограничувања со употреба на **CONSTRAINT**
- Ограничувањата и условите кои треба да ги исполнуваат вредностите врз доменот на атрибутите се дефинираат со соодветниот услов поставен по клучниот збор **CHECK**

Ограничување на атрибутите (2)

```
CREATE TABLE EMP(
    ENAME          VARCHAR(30) NOT NULL,
    ESSN           CHAR(9),
    BDATE          DATE,
    DNO            INTEGER  DEFAULT 1
       CHECK (DNO > 0 AND DNO < 21),
    SUPERSSN       CHAR(9),
    CONSTRAINT EMPSSN
       PRIMARY KEY (ESSN),
    CONSTRAINT EMPDEPT
       FOREIGN KEY (DNO) REFERENCES DEPT(DNumber)
          ON DELETE SET DEFAULT
          ON UPDATE CASCADE,
    CONSTRAINT EMPSUPSSN
       FOREIGN KEY (SUPERSSN) REFERENCES EMP(SSN)
          ON DELETE SET NULL
          ON UPDATE CASCADE   );
```

Дополнителни типови во SQL2 и SQL-99

Тоа се DATE, TIME и TIMESTAMP податочните типови

- **DATE:**
  - Составен од година-месец-ден во форматот yyyy-mm-dd
- **TIME:**
  - Составено од час:минути:секунди во форматот hh:mm:ss
- **TIME(i):**
  - Составено од час:минути:секунди плус i дополнителни цифри кои што ги одредуваат деловите од секундата
  - форматот е hh:mm:ss:ii...i

Дополнителни типови во SQL2 и SQL-99

- **TIMESTAMP:**
  - Ги има и DATE и TIME како составни делови
- **INTERVAL:**
  - Одредува релативна вредност наместо апсолутна вредност
  - Може да бидат DAY/TIME интервали или YEAR/MONTH интервали
  - Може да бидат позитивни или негативни
  - Кога ќе се додадат или одземат од апсолутната вредност, резултатот е апсолутна вредност

DROP TABLE

- Се употребува за отстранување на некоја табела (релација) од шема на БП заедно со нејзината дефиниција

default

```
DROP TABLE Table_name [RESTRICT|CASCADE];
```

- По отстранувањето, табелата не може да се употребува во прашања, ниту во други наредби бидејќи нејзиниот опис веќе не постои
- Ако табелата е референцирана од други табели, тогаш треба да се користи опцијата CASCADE по името на табелата
- Пример:

```
DROP TABLE DEPENDENT;
```

ALTER TABLE

- Се користи за да ја промени дефиницијата на некоја табела во базата на податоци
  - Ако се додава нова колона, штом се изврши наредбата, новиот атрибут ќе има вредности NULL во сите торки на релацијата; заради тоа, ограничувањето NOT NULL не е дозволено при додавање на нови атрибути
  - Пример:

```
ALTER TABLE EMPLOYEE
ADD JOB VARCHAR(12);
```

  - Останати варијанти се промена на колона:

```
ALTER TABLE EMPLOYEE
ALTER COLUMN JOB VARCHAR(25);
```

  - И бришење на колона:

```
ALTER TABLE EMPLOYEE
DROP COLUMN JOB;
```

ALTER TABLE

- Се користи за да ја промени дефиницијата на некоја табела во базата на податоци
  - Може да се искористи и за додавање и бришење на ограничувања
  - Додавање на ограничување:

```
ALTER TABLE Table_name
ADD CONSTRAINT Constr_name PRIMARY KEY(Atr_name);
```

  - Бришење на ограничување:

```
ALTER TABLE Table_name
DROP CONSTRAINT Constr_name;
```

Прашања за пребарување во SQL

- SQL има една основна наредба за извлекување на податоци од базата – тоа е наредбата **SELECT**
- Битна разлика помеѓу SQL и формалниот релациски модел:
  - SQL дозволува некоја табела (релација) да има два или повеќе реда (торки) кои што се идентични за сите вредности на атрибутите
  - Оттука, SQL табела (релација) е **повеќекратно множество** (**multi-set**), наречен и **вреќа** (**bag**) од торки; НЕ е множество од торки

SQL релациите може да бидат ограничени да станат множества со одредување на PRIMARY KEY или UNIQUE за некои атрибути, или со употреба на опцијата DISTINCT во прашањата

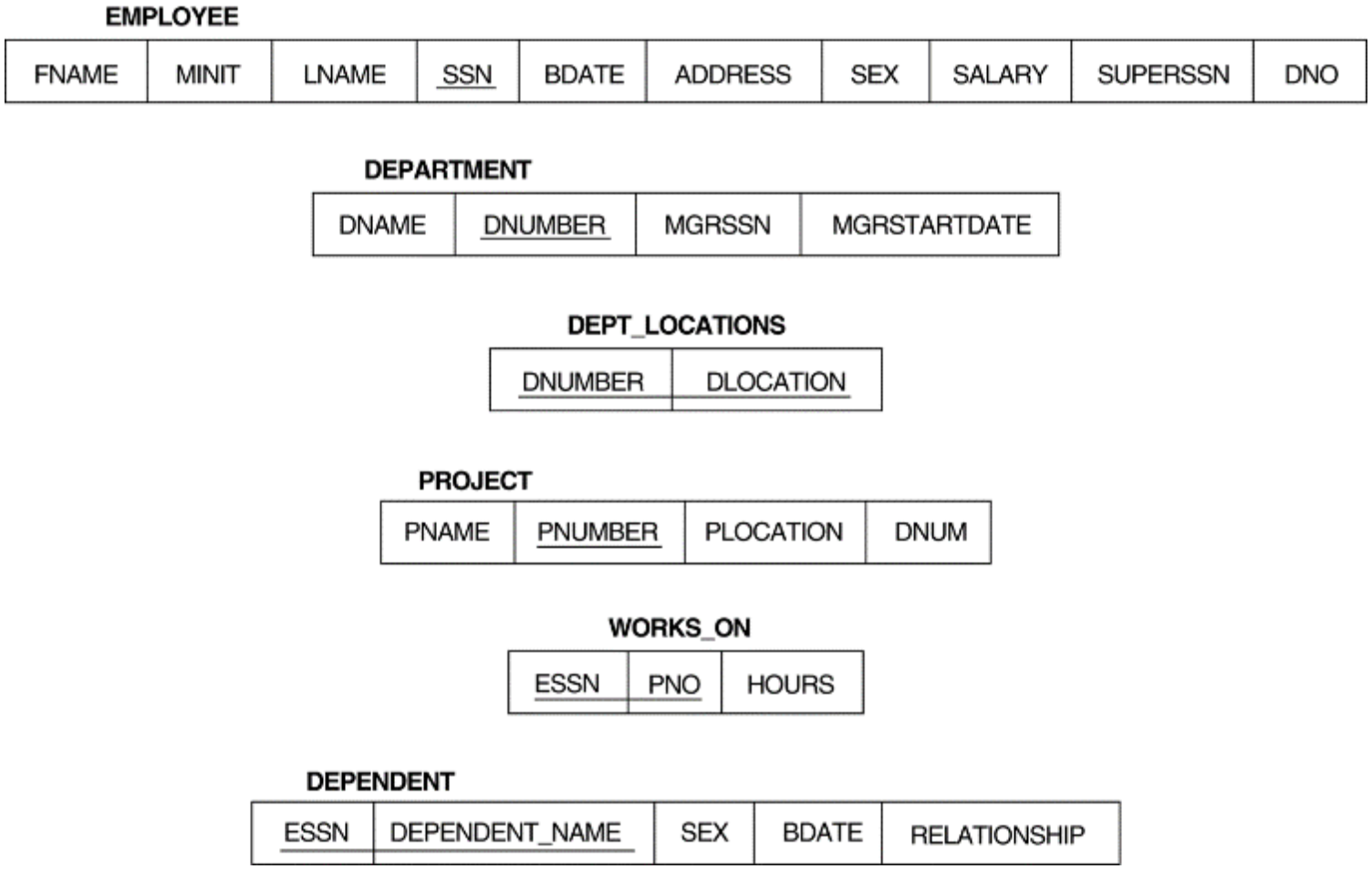
Прашања за пребарување во SQL (2)

- Основниот облик на SQL-SELECT наредбата е:

```
SELECT   <attribute list>
FROM     <table list>
WHERE    <condition>
```

- <attribute list> е список на имиња на атрибути чиишто вредности треба да бидат извлечени со прашањето од базата
- <table list> е список на имиња на релациите од кои ќе се влечат податоците потребни за обработка на прашањето
- <condition> е некој логички израз кој треба да го задоволат торките што треба да бидат извлечени со прашањето (филтрирање)
  - =, <, <=, >, >=,<>
  - AND, OR, NOT

Пример на шема на БП Копманија



Состојба на базата на податоци

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

Едноставни SQL прашања

- Основните SQL прашања соодветствуваат на примената на следниве операции во релационата алгебра:
  - SELECT
  - PROJECT
  - JOIN

Едноставни SQL прашања

- **Прашање а.** Да се врати (добие) датумот на раѓање и адреса на вработениот (или вработените) чие име е 'John B. Smith'

$$\pi_{\text{Bdate,Address}}(\sigma_{\text{Fname='John' AND Minit='B' AND Lname='Smith'}}(\text{EMPLOYEE}))$$

```
Qa:SELECT Bdate, Address
   FROM EMPLOYEE
   WHERE Fname='John' AND Minit='B'
         AND Lname='Smith'
```

- Слично на парот SELECT-PROJECT кај операциите во релациона алгебра:
  - SELECT-делот ги одредува атрибутите што се проектираат
    WHERE-делот ги одредува условите за избор (селекција)
- Сепак, резултатот од прашањето **може да содржи дупликат торки**

Едноставни SQL прашања

- **Прашање b.** Да се вратат сите информации за вработените во одделот број 5.

$$\sigma_{DNO=5}(EMPLOYEE)$$

```
Qb: SELECT *
    FROM EMPLOYEE
    WHERE DNO = 5
```

- За да се извлечат сите вредности на атрибутите на избраните торки, се става *, која што значи **сите атрибути**

Едноставни SQL прашања

- **Прашање c.** Да се вратат матичните броеви на вработените.

π_{SSN} (EMPLOYEE))

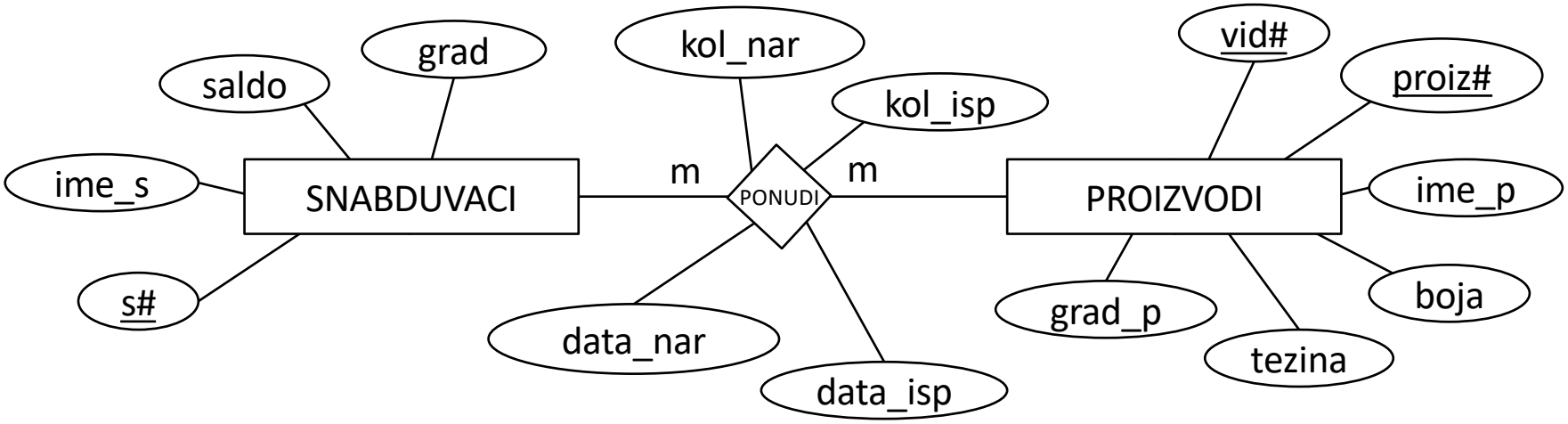
```
Qc: SELECT SSN
    FROM EMPLOYEE
```

- Испуштен WHERE-дел покажува дека не постои услов; оттука, сите торки во релациите од FROM-делот се избрани
  - Еквивалентно е со условот WHERE TRUE

Пример 2

- Да се напишат соодветните наредби за креирање на БП претставена со следниот релациски модел:

- 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)



Пример 2

- Да се напишат дефинициите за примарните клучеви и правилата на интегритет за надворешните клучеви:

```
CREATE TABLE snabduvaci (
    s# NUMBER(3),
    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),
);
```

```
CREATE TABLE ponudi (
    p# NUMBER(5),
    s# NUMBER(3),
    pr# NUMBER(5),
    v# NUMBER(2),
    kolicina_nar NUMBER,
    datum_nar DATE,
    kolicina_isp NUMBER,
    datum_isp DATE,
);
```

Пример 2

- Да се напишат дефинициите за примарните клучеви и правилата на интегритет за надворешните клучеви:

- 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)

```
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#)
);
```

Пример 2

- Да се дефинираат правила на интегритет што ќе ги задоволат следните услови):
  a. имињата на снабдувачите и производителите не смеат да бидат празни;
  b. имињата на снабдувачите се единствени (различни);
  c. салдото да има вредност поголема од нула;
  d. доколку не се внесе градот на снабдувачот, тогаш да се додели предодредена вредност Скопје;
  e. градот каде што се прават производите да биде еден од Лондон, Париз, Рим;
  f. испорачаната количина да не е поголема од нарачаната;
  g. не сакаме да ги чуваме понудите од снабдувачите кои се избришани од БП
  h. сакаме да ги чуваме понудите за продуктите кои се избришани од БП

Пример 2

- Да се напишат дефинициите за примарните клучеви и правилата на интегритет за надворешните клучеви:

```
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 ponudi_FK FOREIGN KEY (pr#, v#)
    REFERENCES proizvodi(proiz#,vid#)
      ON DELETE SET NULL  ON UPDATE CASCADE
);
```

a, c, d, b, a, e, g, f, h

Ако се избрише даден снабдувач се бришат сите негови понуди

Ако се избрише даден производ се чуваат сите понуди за тој производ, кај нив се поставува вредност NULL за надворешните клучеви

Пример 2

- 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)
);
```

наместо CHECK (saldo > 0)

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ
И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

АЖУРИРАЊЕ НА БАЗИ НА ПОДАТОЦИ

БАЗИ НА ПОДАТОЦИ - предавања

Вон. проф. д-р Кире Триводалиев

Наредби за ажурирање кај SQL

- Постојат три SQL наредби за промени на содржината на базата на податоци:
  - **INSERT**
  - **DELETE** и
  - **UPDATE**

INSERT

INSERT INTO <table name> **VALUES** (<atributes>)

- Се користи за да додаде **една** или **повеќе** торки во табелата
  - Може да се додаваат целосни торки (со сите вредности на атрибутите), или
  - Делумни торки (со одредени вредности на атрибутите)

INSERT

- Вредностите на атрибутите **мора** да се излистани по истиот редослед како и атрибутите во дефиницијата на табелата (пример, при специфицирањето во **CREATE TABLE** наредбата)

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

```
U1: INSERT INTO EMPLOYEE VALUES
    ('Richard','K','Marini',
     '653298653', '30-DEC-52',
     '98 Oak Forest,Katy,TX',
     'M', 37000,'987654321', 4)
```

INSERT

- Исто така може експлицитно да се специфицираат имињата на атрибутите за кои се наведени вредностите во делот VALUES
  - Така, атрибутите со NULL вредности може и да се изостават

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

```
U1A: INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)
     VALUES ('Richard', 'Marini', '653298653')
```

Останатите атрибути ќе се пополнат или со предефинираните вредности (доколку ги има) или со NULL вредности!

INSERT

- Друга варијанта на INSERT дозволува додавање на повеќе торки одеднаш, кои се резултат на некое прашање
- **Пример**: Сакаме да создадеме привремена табела која ќе ги содржи називите на одделите заедно со бројот на вработени и вкупните плати за секој оддел.

```
U3A: CREATE TABLE DEPTS_INFO
     (DEPT_NAME VARCHAR(10),
      NO_OF_EMPS INTEGER,
      TOTAL_SAL INTEGER);

U3B: INSERT INTO
     DEPTS_INFO (DEPT_NAME, NO_OF_EMPS, TOTAL_SAL)
     SELECT DNAME, COUNT(*), SUM(SALARY)
     FROM DEPARTMENT, EMPLOYEE
     WHERE DNUMBER=DNO
     GROUP BY DNAME ;
```

Деталите за значењето на овие SQL прашања ќе бидат објаснети на следните предавања

INSERT

- **Забелешки за претходниот пример**:
  - Табелата DEPTS_INFO е создадена од прашањето U3A, и е наполнета со сумарната информација извлечена од базата на податоци со прашањето U3B
  - Табелата DEPTS_INFO може **да не содржи актуелни податоци**
    - ако по извршувањето на прашањето U3B се променат некои од записите било во табелата DEPARTMENT било во табелата EMPLOYEE, тогаш DEPTS_INFO ќе содржи невалидни податоци

Овој проблем се разрешува доколку се создаде т.н. **поглед** (**view**)
преку кој ќе се одржува ажурноста на ваквите сумарни прегледни табели

Ќе биде објаснето на следните предавања

DELETE

DELETE FROM <table> **WHERE** <conditions>

- Ги брише записите од табелата
  - Вклучува WHERE-дел за да се изберат торките што ќе бидат избришани
    - Ако не се наведе WHERE-делот, тогаш сите торки во табелата ќе бидат избришани – табелата ќе стане празна табела
    - Бројот на избришани торки зависи од бројот на торки во релацијата кои го задоволуваат условот во WHERE-делот
  - Мора да се применува референцијалниот интегритет
    - Торки се бришат само од една табела во даден момент (освен ако не е одредена и CASCADE опцијата)

DELETE

- Избриши ги сите вработени со презиме Brown.

```
U4A:  DELETE  FROM  EMPLOYEE
      WHERE  LNAME  =  'Brown'
```

- Избриши ги сите вработени со матичен број 123456789.

```
U4B:  DELETE  FROM  EMPLOYEE
      WHERE  SSN  =  '123456789'
```

DELETE

- Избриши ги сите вработени што работат во одделот Research.

```
U4C: DELETE FROM EMPLOYEE
     WHERE DNO IN
         ( SELECT DNUMBER
           FROM DEPARTMENT
           WHERE DNAME='Research')
```

- Избриши ги сите вработени.

```
U4D: DELETE FROM EMPLOYEE
```

UPDATE

UPDATE <table> **SET** <attributes>
WHERE <conditions>

- Се употребува за да ги промени вредностите на атрибутите на еден или повеќе избрани записи
  - Дополнителен SET-дел одредува кои атрибути ќе бидат променети со новите вредности
  - WHERE-делот одредува кои торки ќе бидат променети
  - Секоја наредба променува торка во истата релација
  - Референцијалниот интегритет се проверува и зачувува

UPDATE

- **Пример**: За проектот со реден број 10, промени ги местоположбата и управувачкиот оддел на 'Bellaire' и 5, соодветно.

```
U5: UPDATE PROJECT
    SET PLOCATION = 'Bellaire', DNUM = 5
    WHERE PNUMBER = 10
```

UPDATE

- **Пример**: Дај им на сите вработени во одделот 'Research' повишување на платата од 10%.

```
U6: UPDATE EMPLOYEE
    SET SALARY = SALARY * 1.1
    WHERE DNO IN (SELECT DNUMBER
                  FROM DEPARTMENT
                  WHERE DNAME='Research')
```

- Во ова барање, променетата вредност за SALARY зависи од оригиналната вредност на SALARY за секоја торка
  - Повикувањето на атрибутот SALARY на **десно** од знакот = се однесува на **старата** вредност на SALARY пред промената
  - Повикувањето на атрибутот SALARY на **лево** од знакот = се однесува на **новата** вредност на SALARY што ќе ја добие по промената

Прашање 1

- Најважни наредби во DDL делот од јазикот SQL се:

A) CREATE, DROP и ALTER

B) INSERT, DELETE и UPDATE

C) CREATE, DELETE и ALTER

D) INSERT, DROP и UPDATE

Прашање 2

- Кои можни нарушувања на интегритетот може да настанат при операцијата DELETE?

A) ограничување на доменот

B) ограничување на клучот

C) референцијалниот интегритет

D) ентитетниот интегритет

Прашање 3

- Дадени се SQL декларациите за креирање на две табели S и T:

  CREATE TABLE S(c INT PRIMARY KEY, d INT);

  CREATE TABLE T(a INT PRIMARY KEY, b INT REFERENCES S(c));

- Нека S(c,d) е пополнета со торките: (2,10), (3,11), (4,12), (5,13), а T(a,b) со: (0,4), (1,5), (2,4), (3,5). Врз основа на ограничувањата во S и T некои внесувања, бришења и/или промени не се дозволени.

- Кои од следните операции **нема да ги нарушат** овие ограничувања?

  A) INSERT (0,3) во T

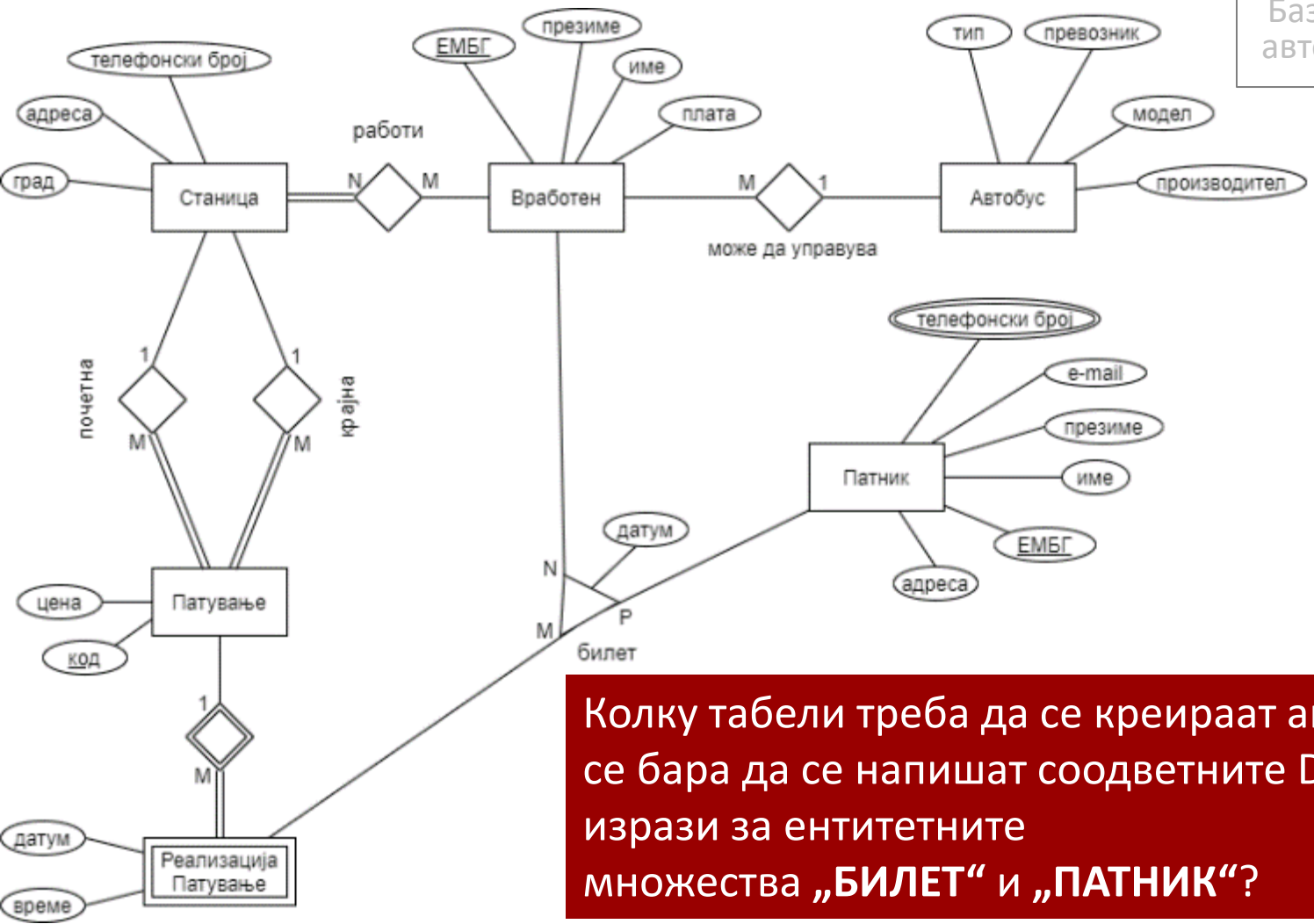
  B) INSERT (4,10) во S

  C) DELETE (5,13) од S

  D) INSERT (7,5) во T

Прашање 4

База на податоци за автобуска компанија



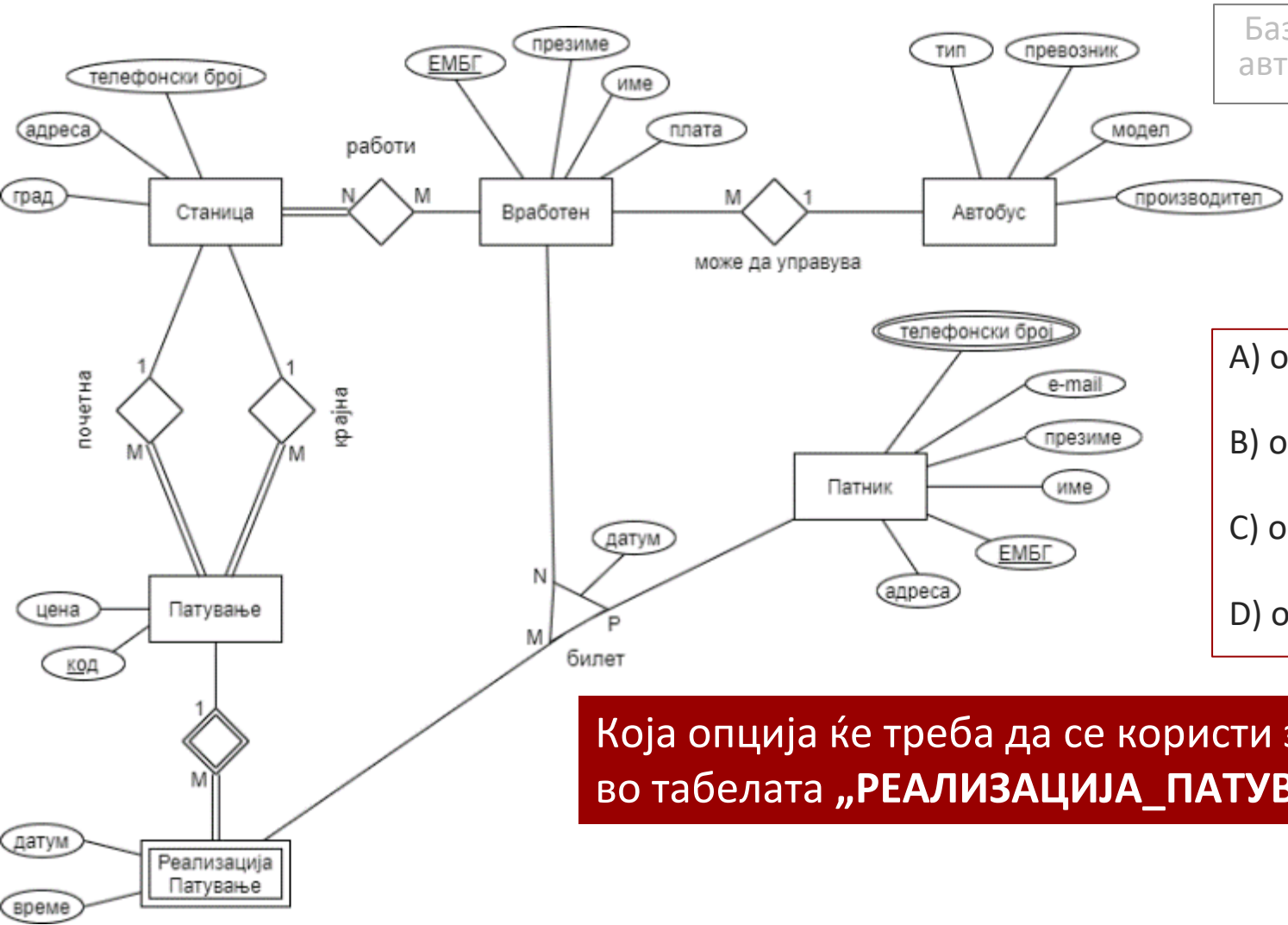
Колку табели треба да се креираат ако се бара да се напишат соодветните DDL изрази за ентитетните множества **„БИЛЕТ“** и **„ПАТНИК“**?

A) 1 B) 2

C) 3 D) 4

E) 5

Прашање 5



A) on delete set null

B) on delete cascade

C) on delete set default

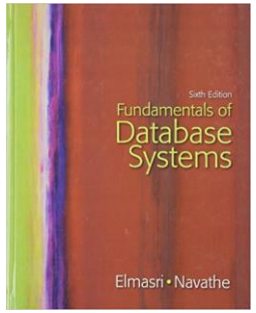
D) on delete restrict

Која опција ќе треба да се користи за PatuvanjeKod, во табелата **„РЕАЛИЗАЦИЈА_ПАТУВАЊЕ“**?

Користена литература

- **Глава 8** (241 - 296)

- **Глава 3** (59 - 85)
- **Глава 4** (87 - 114)



- **Глава 6** (287 - 358)
- **Глава 7** (359 - 392)