2024/2025

Računalništvo04
IS, UML, PB, SQL, ER

predavatelj: Aleksandar Lazarević

Agenda

- **Podatkovne baze**
- **SQL:**
  - **SQL DDL**
  - **SQL DDL**
  - **SQL DCL**

2024/2025

Podatkovne baze

Motivacija za uporabo podatkovnih baz (PB); definicija PB; shema in stanje PB; relacijske PB

Motivacija – študij primera

- Od naročnika 'Virtualna šola Miki Miška' smo dobili nalogo narediti sistem, ki bo omogočal hranjenje podatkov o študentih, tečajih, profesorjih, voditeljih in udeležencih posameznih tečajev. Aplikacija naj omogoča:
  1. Hranjenje podatkov za daljše časovno obdobje ➔ lahko sklepamo, da se bo s časom nabrala velika količina podatkov (tudi več 100MB)
  2. Zaščito pred nesrečami in zaščito pred nepooblaščeno uporabo podatkov.
  3. Izvajanje različnih poizvedb (primer: Kateri učitelj poučuje predmet MAT4).
  4. Dodajanje, spreminjanje in brisanje podatkov.
  5. Sočasen dostop do podatkov več 10 oz. 100 uporabnikom.
  6. Skrbnik sistema mora imeti možnost za dodajanje in spreminjanja tipov podatkov, uporabnikov aplikacije, ….

1. premislek

- Za programiranje izberemo C++/Javo
- Podatke bomo shranjevali v datoteke
- Datotečno strukturo bomo zapisali v kodo programa

- Problemi:
  1. Nadzor sočasnosti dostopov
  2. Izbira algoritmov
  3. Veliko različnih poizvedb
  4. Zagotavljanje nedeljivosti transakcij
  5. Izvoz podatkov v druge formate
  6. Spremembe strukture podatkov

1. premislek

Rešitev:

1. Vse bomo sprogramirali
2. V rezervi bomo vedno imeli vsaj 2 programerja, ki bodo pisali nove funkcije, za izvedbo različnih poizvedb nad podatki.
3. Za letno vzdrževanje aplikacije bomo od uporabnikov zahtevali vsaj toliko denarja, koliko stane prvotni razvoj aplikacije.

Težava: Uporabniki niso pripravljeni plačati tako visoke cene vzdrževanja!

2. premislek

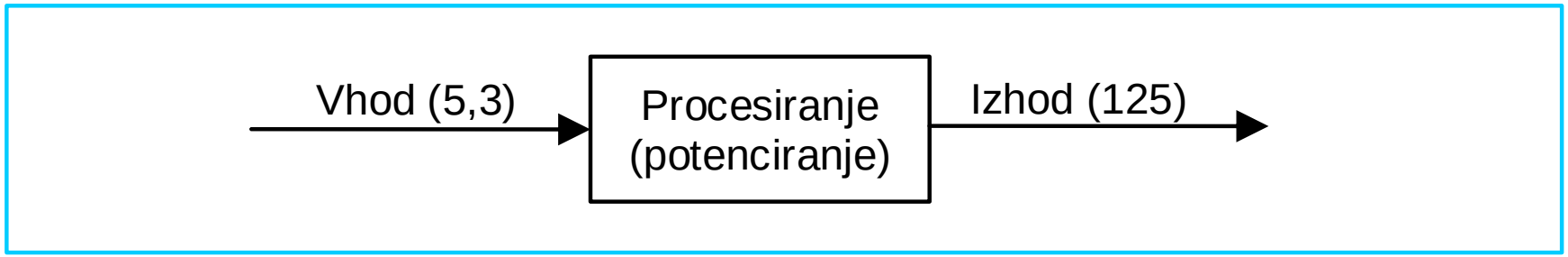
- Slišali smo za nek nov način shranjevanja podatkov v podatkovne baze, pri katerem večina prejšnjih težav odpade ali se bistveno zmanjša.
- Uporabnikom predlagamo drugačen način implementacije sistema z bistveno nižjo ceno vzdrževanja.

Uporabniki se strinjajo!

PB – trajno hranjenje podatkov?

1. primer

Danes			Čez 14 dni	
Koliko je 5^3?	Odgovor = 125		Koliko je 5^3?	Odgovor = 125

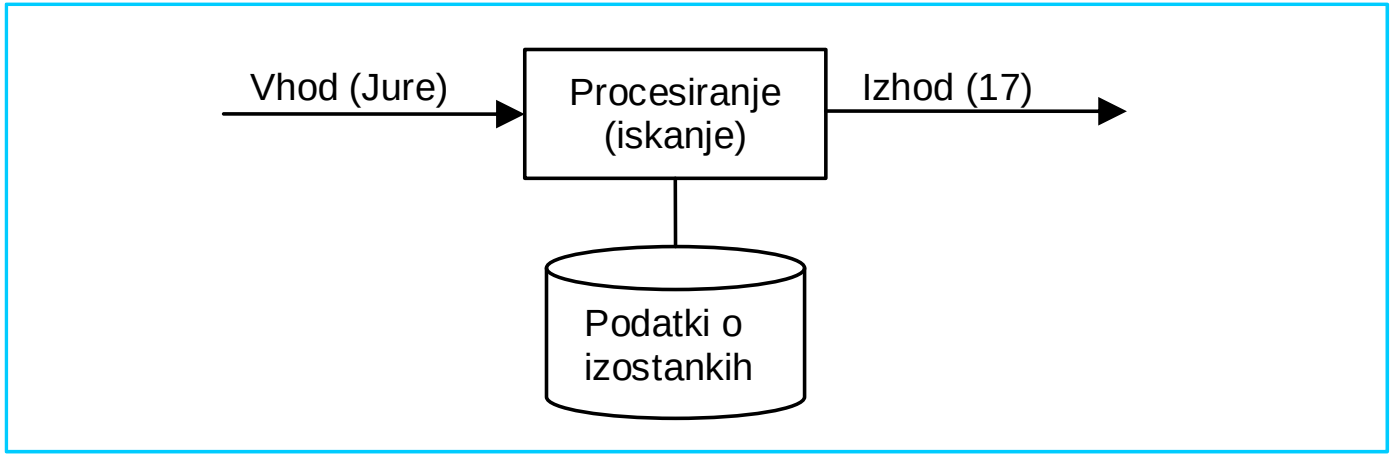


Trajno hranjenje podatkov v tem primeru ni potrebno!

PB – trajno hranjenje podatkov

2. primer

Danes			Čez 14 dni	
Koliko (ne)opravičenih izostankov ima moj sin Jure?	Odgovor = 2		Koliko (ne)opravičenih izostankov ima moj sin Jure?	Odgovor = 17

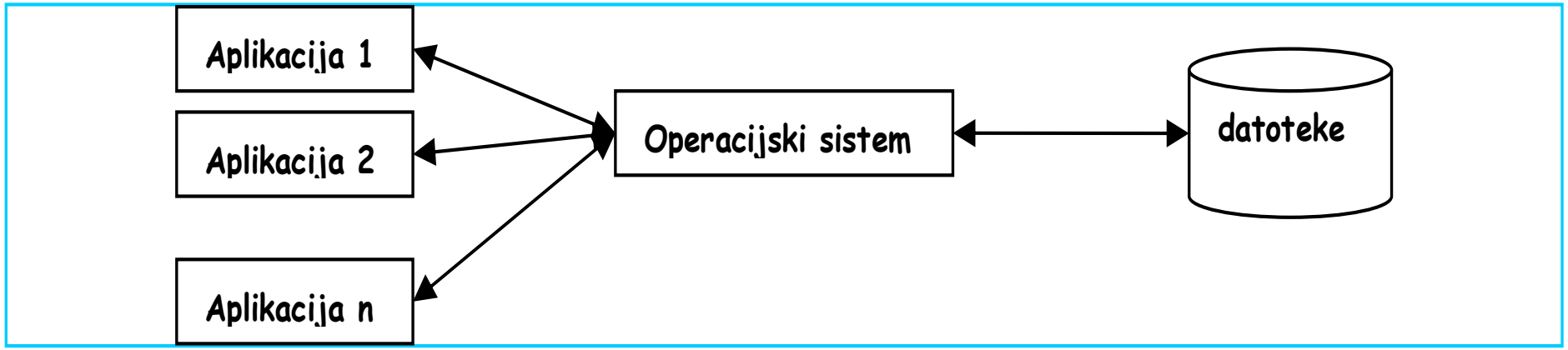


Trajno hranjenje informacije / podatka oziroma trajna informacija = informacija, katere življenjski čas je daljši od življenjskega časa enega procesa (poenostavljeno gledano izvedbe programa)

Načini implementacije trajnosti podatkov

- ‘klasični’ (datoteke)
- uporaba PB (podatkovnih baz) in SUPB (sistemov za upravljanje s podatkovnimi bazami)

Klasični načini implementacije trajnosti

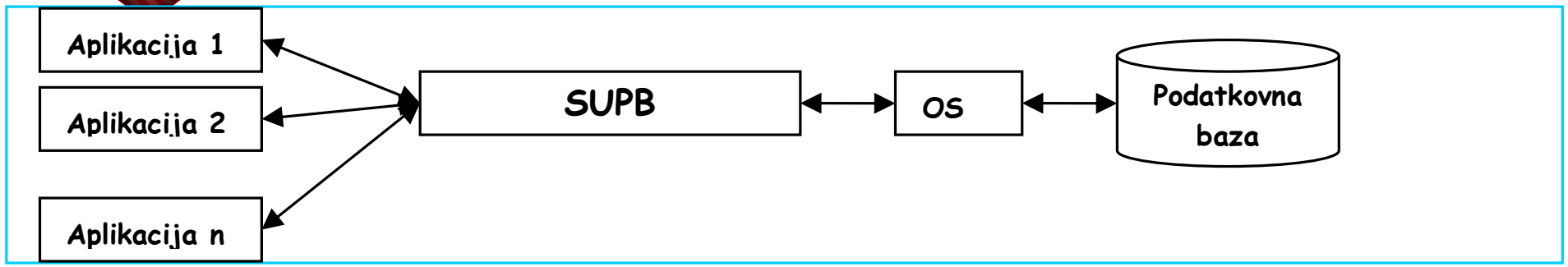


- Datoteka na nivoju OS = zaporedje zlogov
- Programi za obdelavo podatkov morajo poznati strukturo podatkov - zapisana je v kodi programa
- OS ne more preprečiti napak, ker ne pozna strukture datotek.
- Vse zaščite in druge omejitve morajo biti realizirane znotraj programa!
- Če več programov oziroma uporabnikov sočasno uporablja eno datoteko -> programsko zagotavljanje zaščite podatkov (v vseh programih)

Slabosti ‘klasičnih’ sistemov

- ☹ časovno požrešna metoda razvoja programske opreme (veliko programiranja)
- ☹ ni podpore 'ad hoc' poizvedbam
- ☹ izolirani ‘otoki’ informacij (ločene datoteke po različnih oddelkih podjetja)
- ☹ močna odvisnost med podatki in programi (sprememba strukture datoteke zahteva tudi spremembo vseh programov, ki jo uporabljajo)
- ☹ večkratno zapisovanje istih podatkov ➔ nenadzorovana redundanca podatkov ➔pri dodajanju, spreminjanju in brisanju podatkov pogosto prihaja do anomalij med podatki
- ☹ Pomanjkanje kontrole ➔ nekonsistenca/neskladnost podatkov
- ☹ Podatki, shranjeni s pomočjo ene aplikacije ne morejo biti obdelani s strani druge aplikacije (nekompatibilnosti formatov)

Uporaba PB in SUPB-ja za implementacijo trajnosti



- struktura shranjenih podatkov mora biti podana na način, ki ga SUPB razume; primer:

```
Create Table "Dijak"  (
    "DijakID" Integer NOT NULL,
    "Priimek" Char(20) NOT NULL,
    "Ime" Char(10) NOT NULL,
    "Razred" Char(3),
    Primary Key ("DijakID")
);
```

- SUPB sam odkriva in preprečuje napake
- Programiranje se poenostavi in se prestavi na višji nivo abstrakcije.
- SUPB - vmesnik med OS in aplikacijskimi programi / uporabniki

Uporaba PB in SUPB-ja za implementacijo trajnosti

- Koda SUPB-ja je testirana in dobro optimizirana
- SUPB-ji imajo vgrajene rutine za razvrščanje in iskanje podatkov, za upravljanje z izrabo datotečnih izravnalnikov (bufferjev), za shranjevanje datotek, statistične funkcije, …
- SUPB je optimiziran za delo z velikimi količinami podatkov, podpira večuporabniško okolje, varnostne mehanizme, ki preprečujejo dostop do podatkov nepooblaščenim osebam in mehanizme, ki zagotavljajo zaščito podatkov v primeru porušitve sistema ter obnavljanje sistema.
- PB je dostopna le s pomočjo SUPB ➔ omogočeno je skrivanje / prikrivanje internih sprememb znotraj podatkovne baze uporabnikom in uporabniškim programom. Vodilo abstraktnih podatkovnih tipov je možnost implementacije sprememb na nižjem nivoju in hkrati ohranjanje vmesnika proti višjem nivoju.

Definicija podatkovnih baz

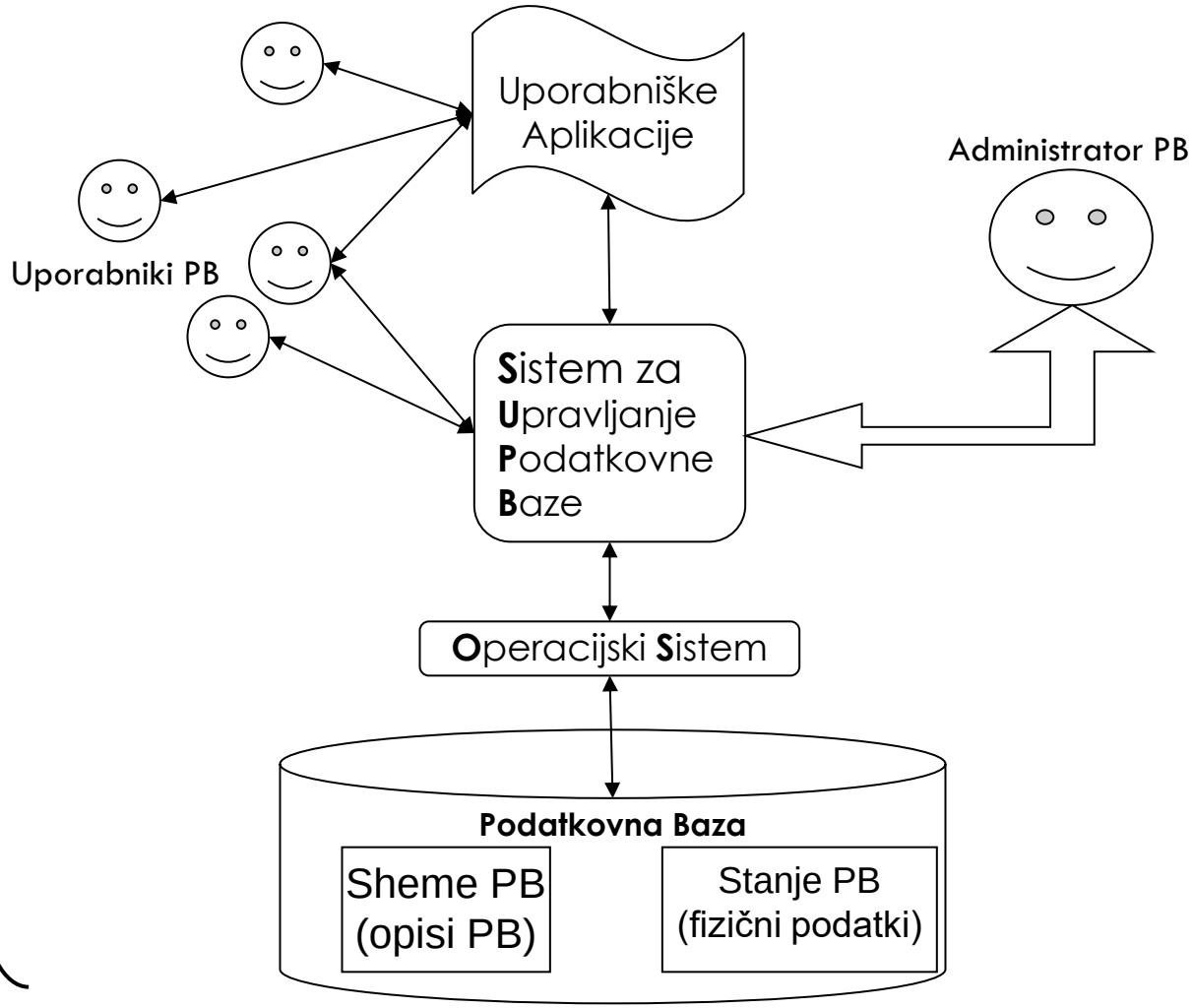
PB = zbirka povezanih podatkov.
Podatek - dejstva, ki so shranjena na nekem računalniškem trajnem pomnilniku in ki se jim lahko pripiše pomen. (ki implicitno imajo pomen). Elmasri and Navathe

PB = upravljana zbirka povezanih podatkov, shranjena na računalniškem sistemu, deljena med več uporabniki, zaščitena z varnostnimi mehanizmi in shranjena z nadzorovano redundantnostjo. Stamper and Price

PB = organizirana zbirka logično povezanih podatkov in opisov le teh, načrtovana tako, da zadovoljuje informacijske potrebe organizacije. Connolly and Begg

Shematski prikaz podatkovnega sistema

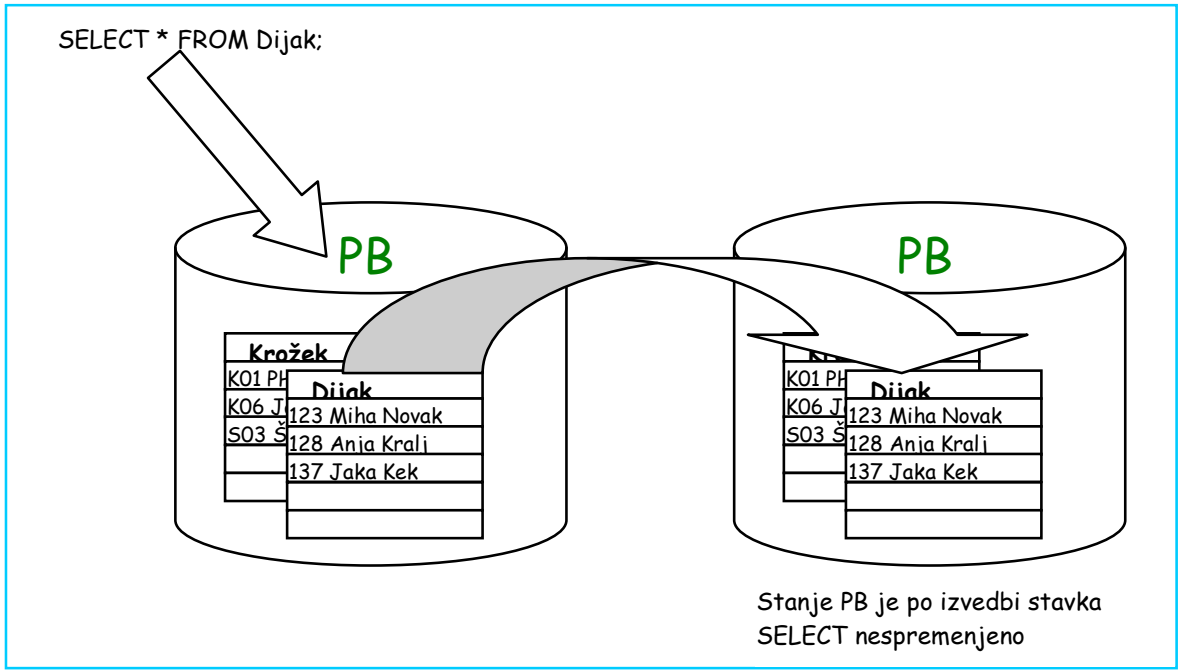
Podatkovni sistem



Stanje PB

Stanje PB = množica podatkov, ki so trenutno shranjeni v PB

Najpogostejša operacija nad podatki je branje podatkov (SQL = SELECT) – stanje PB se ne spremeni



Strukturiranost podatkov

Podatki v PB so ***strukturirani*** – ustrezajo vnaprej opredeljeni strukturi

Strukturiranost podatkov omogoča kompleksnejša vrednotenja stanja podatkovne baze

Kaj bi se zgodilo, če imamo v PB zapisane le stavke v naravnem jeziku (tekst)?

Podatkovni sistemi, ki več vedo o podatkih, ponujajo uporabnikom boljše storitve in s tem bolje podpirajo delo uporabnikov.

Shema PB

- formalna definicija strukture vsebine podatkovne baze
- opredeli vsa možna stanja podatkovne baze
- definirana je le enkrat (pri kreiranju PB)
- Spremembe sheme že kreirane PB ➔ včasih težave
- dobro (temeljito) načrtovanje ➔ manj sprememb sheme PB ➔ manj težav
- sinonim za shemo: metapodatkovna baza (to so podatki o podatkih)
- poenostavljena primerjava z višjimi programskimi jeziki:
  - Shema baze – deklaracija strukture
  - Stanje baze – trenutna vrednost v spremenljivki

Stanje vs. shema PB

Dijaki				
IDDijak	**Priimek**	**Ime**	**Razred**	**Telefon**
10205	Mlinar	Mateja	G2A	01-123-333
10301	Dolenc	Mitja	G2A	03-313-313
10305	Verk	Maria	G2C	NULL
10309	Žavbi	Jana	G2B	NULL
10310	Juh	Polona	G2C	01-111-111

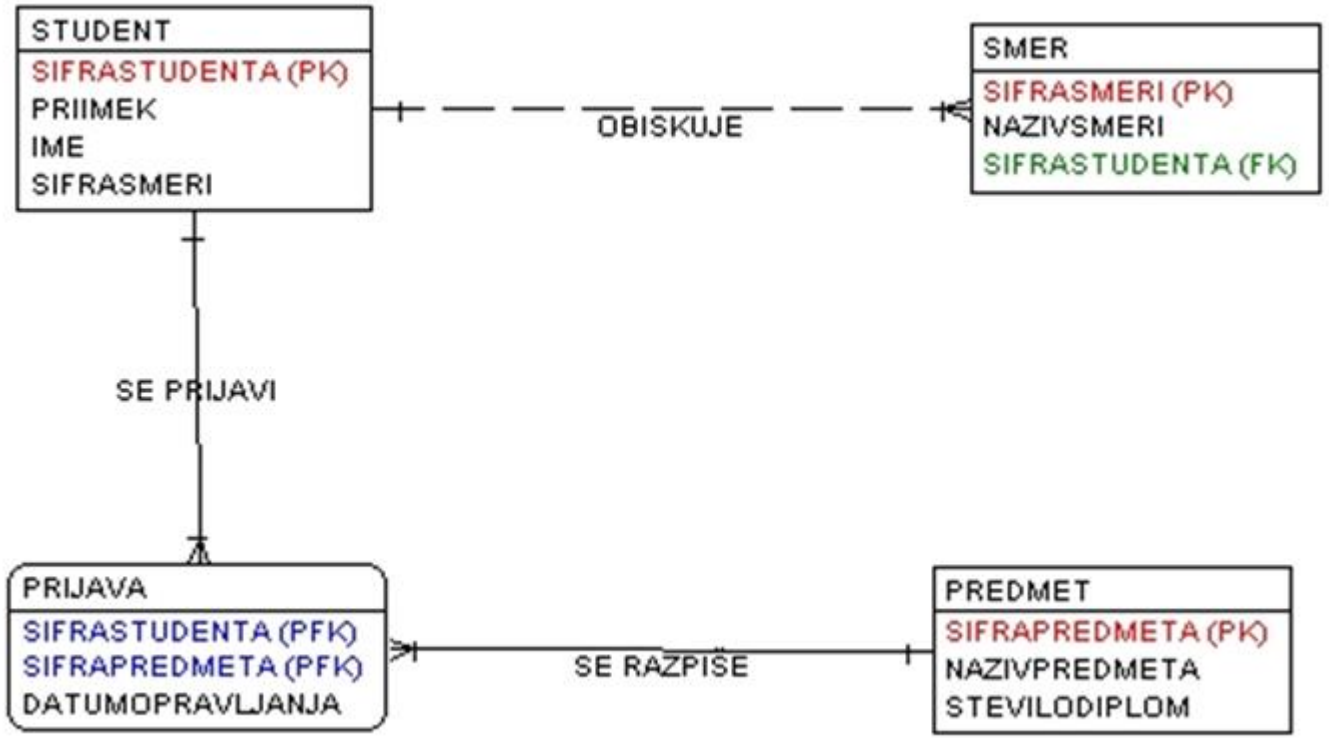
Shema PB // v resnici gre le za del sheme PB

Stanje PB

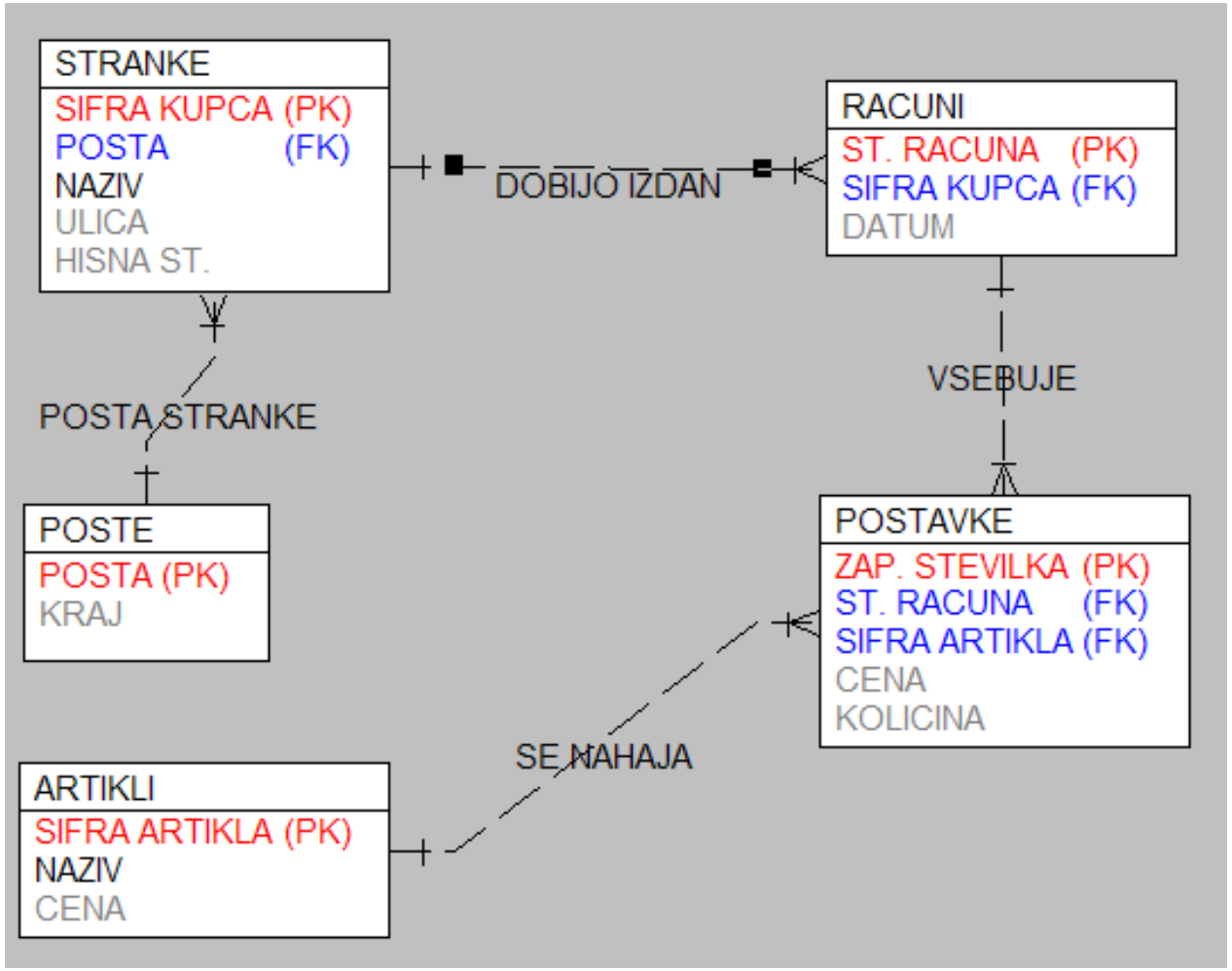
Relacijske podatkovne baze

- najbolj pogoste PB so t.i. relacijske podatkovne baze
- temeljijo na ER-modelih
- osnovni gradniki:
  - entitete/entitetni tipi: predstava o množici podobnih entitet; entitete opisujem z atributi
    - entitetni identifikator - ključ
  - relacije/razmerja – opredeljujejo povezave med entitetnimi tipi
    - stopnja razmerja/številnost – število povezanih entitetnih tipov

Relacijske podatkovne baze - primer



Relacijske podatkovne baze – primer



Uvod v SQL...

- SQL – jezik za uporavljanje relacijskih PB
- SQL sestavljata dve skupini ukazov:
  - Skupina ukazov DDL (*Data Definition Language*) za opredelitev strukture podatkovne baze in
  - Skupina ukazov DML (*Data Manipulation Language*) za poizvedovanje in ažuriranje podatkov.
- SQL do izdaje SQL:1999 ne vključuje ukazov kontrolnega toka. Kontrolni tok je potrebno obvladati s programskim jezikom ali interaktivno z odločitvami uporabnikov.

Uvod v SQL...

- Lastnosti SQL:
  - enostaven
  - nepostopkoven (***kaj*** in ne ***kako***)
  - uporaben v okviru številnih vlog: skrbniki PB, vodstvo, razvijalci informacijskih rešitev, končni uporabniki
  - obstaja ISO standard za SQL
  - SQL de-facto in tudi uradno standardni jezik za delo z relacijskimi podatkovnimi bazami

Zgodovina SQL

- V 1970h IBM razvija sistem System R, ki bo temeljil na relacijskem modelu.
-
- 1974 – D. Chamberlin in F. Boyce (IBM San Jose Laboratory) definirata jezik 'Structured English Query Language' (SEQUEL).
  - SEQUEL se kasneje preimenuje v SQL

- Pozno v 1970h – Relational Software (danes Oracle) razvije svoj SUPB, ki temelji na relacijskem modelu in implementira SQL.

- Poleti 1979 – Oracle izda prvo komercialno različico SQL; nekaj tednov pred IBM-ovo implementacijo System/38

Standardizacija SQL

Vir: Wikipedia EN

Year	Name	Alias	Comments
1986	SQL-86	SQL-87	First formalized by ANSI.
1989	SQL-89	FIPS 127-1	Minor revision, adopted as FIPS 127-1.
1992	SQL-92	SQL2, FIPS 127-2	Major revision (ISO 9075), *Entry Level* SQL-92 adopted as FIPS 127-2.
1999	SQL:1999	SQL3	Added regular expression matching, recursive queries, triggers, support for procedural and control-of-flow statements, non-scalar types, and some object-oriented features.
2003	SQL:2003	SQL 2003	Introduced XML-related features, *window functions*, standardized sequences, and columns with auto-generated values (including identity-columns).
2006	SQL:2006	SQL 2006	ISO/IEC 9075-14:2006 defines ways in which SQL can be used in conjunction with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form. In addition, it enables applications to integrate into their SQL code the use of XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.[24]
2008	SQL:2008	SQL 2008	Legalizes ORDER BY outside cursor definitions. Adds INSTEAD OF triggers. Adds the TRUNCATE statement.[25]

Pomembnost jezika SQL...

- SQL do sedaj edini široko sprejet standardni podatkovni poizvedovalni jezik.
- SQL del aplikacijskih arhitektur (npr. v okviru IBM-ove arhitekture - Systems Application Architecture (SAA).
- Strateška odločitev več pomembnih združb
  - Konzorcij X/OPEN za UNIX
  - Federal Information Processing Standard (FIPS) – standard, kateremu morajo ustrezati vsi SUPB-ji prodani državnim organom v ZDA.
  - …

Pomembnost jezika SQL

- SQL uporabljen tudi v drugih standardih
  - ISO Information Resource Dictionary System (IRDS)
  - Remote Data Access (RDA),...
- Interes v akademskih krogih daje jeziku teoretično osnovo in tehnike za implementacijo
  - Optimizacija poizvedb
  - Distribucija podatkov
  - Varnost podatkov
- Pojavljajo se specializirane implementacije SQL, npr. za OLAP

Implementacije SQL

- Med standardi SQL-92, SQL:1999; SQL:2008,… razlike
- Implementacije ponudnikov SUPB različne (dialekti)

- Primerjava implementacij SQL
  - http://troels.arvin.dk/db/rdbms/

Uvod v SQL...

- SQL sestavljata tri skupine ukazov:
  - Skupina ukazov DDL (*Data Definition Language*) za opredelitev strukture podatkovne baze in
  - Skupina ukazov DML (*Data Manipulation Language*) za poizvedovanje in ažuriranje podatkov.
  - Skupina ukazov DCL (*Data Control Language*) za upravljanje s pravicami uporabnikov.

- SQL do izdaje SQL:1999 ne vključuje ukazov kontrolnega toka. Kontrolni tok je potrebno obvladati s programskim jezikom ali interaktivno z odločitvami uporabnikov.

SQL jezik - DDL

- DDL ukazi so: CREATE, ALTER, DROP <bazni objekt>
- Glavni objekti v PB so tabele:
  - CREATE TABLE – kreiranje nove tabele
  - ALTER TABLE – spreminjanje obstoječe tabele
  - DROP TABLE – brisanje tabele

Kreiranje tabele - CREATE TABLE

Tabela se v bazi kreira takoj po uspešno izvedenem CREATE ukazu.

Splošna sintaksa za kreiranje tabele je:

```
CREATE TABLE [shema.] NazivTabele

   (Stolpec1 TipPodatka [CONSTRAINT
naziv_omejitve] TipOmejitve,

   Stolpec2 TipPodatka [CONSTRAINT
naziv_omejitve] TipOmejitve,...

   [CONSTRAINT naziv_omejitve] TipOmejitve
(Stolpec, ... ), ... );
```

Definicija stolpcev/polj

- Kaj je potrebno definirati?
  - naziv stolpca (ang *column name*) – obvezno!
  - podatkovni tip (ang. *data type*) in velikost – obvezno!
  - omejitve (ang. *column constraints*) - neobvezno:
    - primarni ključ (*primary key*),
    - obvezna vrednost (*null/not null*),
    - privzeta vrednost (*default*),
    - enolična vrednost (*unique*),
    - vrednost iz zaloge vrednosti/kontrola (*check*)

Podatkovni tipi

2024/2025

TIP	OPIS
CHAR (<velikost>)	za shranjevanje alfanumeričnih znakov. Minimum je 1. max je 2000.
VARCHAR(<velikost>)	za shranjevanje variabilnega števila znakov – minimum je 1 in maksimum je 4000.
INTEGER(p) **FLOAT(p, s)** **DOUBLE(p, s)**	za shranjevanje števil (fiksna in plavajoča vejica). Max. natančnost (p) in/ali decimalni del (s) je 38.
DATE	za shranjevanje datuma in časa. Datum je fiksne velikosti (7-bytov). Možnost različnih formatov (privzeto DD-MON-YY). Možnost velikega števila različnih formatov.
LONG	za shranjevanje do 2GB znakov. Samo en stolpec v tabeli je lahko tega tipa. Ne more biti indeksiran.
RAW (<velikost>)	uporablja se za shranjevanje binarnih podatkov, kot so grafika, zvok in podobno. Max. velikost je 2000 bytov. Lahko se indeksira.
LONGRAW	podobna LONG tipu. Za shranjevanje binarnih podatkov.
CLOB	znakovni veliki objekti. Max. velikost je 4 GB.
BLOB	binarni veliki objekti. Max. velikost je 4 GB.
BFILE	binarne datoteke. Vsebuje lokator do velike binarne datoteke shranjene zunaj baze. Omogoča I/O tok za dostop do zunanje LOB, ki se nahaja na baznem strežniku. Max. velikost je 4GB.

Tipi podatkov: VARCHAR

- VARCHAR je znakovni tip, ki omogoča shranjevanje nizov znakov (alfanumerični) ***spremenljive dolžine*** (do določenega maksimuma, ki je določen z velikostjo polja).
- Velikost je specificirana znotraj oklepajev, npr. VARCHAR(20).
- Če je podatek manjši od specificirane velikosti, se v stolpcu shranjuje samo do te velikosti – preostali prazni znaki se ne dodajajo originalnem podatku.
- VARCHAR je najbolj primeren tip za vrednosti, ki nimajo fiksne velikosti.

Tipi podatkov: CHAR

- CHAR je znakovni tip, ki omogoča shranjevanje nizov znakov (alfanumerični) *nespremenljive dolžine*.

- CHAR tip bolj učinkovito izkorišča prostor v RDBMS in obdeluje podatke hitreje, kot VARCHAR tip.

Tipi podatkov: NUMERIČNI

- INTEGER je celo število brez decimalnega dela.
- NUMBER (FLOAT, DOUBLE) se uporablja za shranjevanje negativnih, pozitivnih, celih števil ter decimalnih števil z fiksno in plavajočo vejico.
- Ko se uporablja tip NUMBER, je potrebno navesti velikost (*ang. precision*) in število decimalnih mest (*ang. scale*):
  - velikost je skupno število števk – skupaj na levi in desni strani decimalne vejice.
  - število decimalnih mest je skupno število števk na desni strani decimalne vejice.

Primer kreiranja tabele

- Primer kreiranja tabele STUDENTI:

```
CREATE TABLE STUDENTI
  (STUDENT_ID    CHAR (5) PRIMARY KEY,    // nivo stolpca
   PRIIMEK       VARCHAR (15) NOT NULL,
   IME           VARCHAR (15) NOT NULL,
   ULICA         VARCHAR (25),
   MESTO         VARCHAR (15),
   POSTA         CHAR (4) DEFAULT '1000',
   ZACETEK       CHAR (4),
   DAT_ROJ       DATE,
   PROGRAM_ID    INTEGER (3),
   SMER_ID       INTEGER (3),
   TELEFON       CHAR (10),
   CONSTRAINT STUDENT_STUDENT_ID_PK PRIMARY KEY (STUDENT_ID));    // nivo
  tabele
```

Standardni podatkovni tipi

- Standard: ISO/IEC 9075
- Podatkovni tipi se v določeni meri razlikujejo med posameznimi SUPB (RDBMS)
- Primera (reference za podatkovne tipe):
  - Access 2007:
    http://msdn.microsoft.com/en-us/library/bb208866.aspx
  - SQL Server 2005:
    http://msdn.microsoft.com/en-us/library/ms187752(SQL.90).aspx

Tipi omejitev

Obstajata dve vrsti omejitev (*ang. constraints*):

1. Integritetne omejitve *(ang. integrity constraints)*: obsegajo primarni ključ in tuje ključe na tabeli ter primarni ključ na tabeli na katero se tuji ključ referencira.

2. Vrednostne omejitve *(ang. value constraints)*: definirajo ali so dovoljene prazne vrednosti v stolpcih tabel, ali so potrebne edinstvene vrednosti in ali je neka množica vrednosti v stolpcu dovoljena.

Imenovanje omejitev

- Splošno pravilo za imenovanje omejitev je:

  <naziv tabele>_<stolpec>_<tip omejitve>

  - *table name* je naziv tabele na kateri se definira omejitev,
  - *column name* je naziv stolpca na katerega se nanaša omejitev,
  - in *constraint type* je okrajšava, ki se uporablja za identificiranje tipa omejitve.

Imenovanje omejitev

- Na primer, omejitev `ZAPOSLENI_ODDELEK_ID_FK` pomeni:
  - omejitev v tabeli ZAPOSLENI na stolpcu ODDELEK_ID.
  - omejitev je tipa tuji ključ (FK).
  - tuji ključ se povezuje na tabelo z oddelki preko polja ODDELEK_ID.
- Na primer, omejitev `ODDELKI_ODDELEK_ID_PK` pomeni:
  - omejitev na tabeli ODDELKI na stolpcu ODDELEK_ID.
  - omejitev je tipa primarni ključ (PK).

Popularne okrajšave za omejitve

Tip	Okrajšava	
primarni ključ	PK	*ang. primary key*
tuji ključ	FK	*ang. foreign key*
unikatni ključ	UK	*ang. unique key*
vrednost iz zaloge vrednosti - kontrola	CK	*ang. check*
obvezna vrednost	NN	*ang. not null*

Definiranje omejitev (constraints)

- Omejitev (*ang. constraint*) se lahko kreira v času:
  - kreiranja tabele (ukaz `CREATE TABLE` ...)
  - lahko se doda naknadno (ukaz `ALTER TABLE` ...)

- Lahko se definira na dveh nivojih:
  - nivo stolpca/polja (*ang. column level*)
  - nivo tabele (*ang. table level*)

Omejitve - nivo stolpca

- Omejitev na nivoju stolpca se nanaša samo na ustrezni stolpec in se definira skupaj s samim stolpcem.

- Vsaka omejitev se lahko definira na nivoju stolpca razen tujega ključa in sestavljenega primarnega ključa.

Sintaksa:

Stolpec TipPodatka **[CONSTRAINT** ***naziv_omejitve*****]** ***TipOmejitve***

Primer:

```
STAVBA VARCHAR(7) CONSTRAINT lokacija_stavbe_NN NOT NULL
```

Omejitve - nivo tabele

- Omejitve na nivoju tabele se nanašajo na enega ali več stolpcev in se definirajo ločeno od definicije stolpcev.
- Ponavadi se navedejo *po* definiciji stolpcev.
- Vse omejitve se lahko definirajo na nivoju tabele razen omejitve za obvezno polje (NOT NULL constraint).

Sintaksa:

```
[CONSTRAINT naziv_omejitve] Tip_omejitve (Stolpec,...),
```

Primer:

```
CONSTRAINT LOKACIJA_ID_PK PRIMARY KEY(LOKACIJA_ID);
```

Primarni ključ (primary key constraint)

- Primarni ključ je znan kot *entitetna integritetna omejitev* (*ang. entity integrity constraint*)
- Kreira primarni ključ za tabelo. Tabela lahko ima *samo en* primarni ključ.
- Če tabela uporablja **sestavljeni ključ** (več kot en stolpec), potem se takšen ključ lahko definira samo na nivoju tabele.

Primarni ključ – primera

- Na nivoju stolpca je definicija naslednja:

```
STM_ID NUMBER (2) CONSTRAINT STM_STM_ID_PK PRIMARY KEY
```

- Na nivoju tabele je definicija naslednja:

```
CONSTRAINT STM_STM_ID_PK PRIMARY KEY(STM_ID),
```

Tuji ključ (foreign key constraint)

- **Tuji ključ** je znan tudi kot omejitev referenčne integritete (*ang. referential integrity constraint*).

- Uporablja stolpec ali stolpce, kot tuje ključe ter določa povezavo med primarnim ključem *iste* ali *druge* tabele.

Tuji ključ

- Za vzpostavitev tujega ključa na tabeli mora obstajati *druga* (referencirana) tabela in *njen primarni ključ*!

- Stolpca za tuji ključ in referencirani primarni ključ v drugi tabeli ni treba da imata enaki imeni vendar vrednost tujega ključa mora ustrezati vrednosti v nadrejeni (referencirani tabeli) ali biti NULL!

Tuji ključ – primer

- Tuji ključ se kreira samo na nivoju tabele!

- Primer: povezujemo tabelo STUDENT in tabelo PROGRAMI preko polja PROGRAM_ID (student je vpisan na program PROGRAM_ID):

```
CONSTRAINT STUDENT_PROGRAM_ID_FK FOREIGN KEY(PROGRAM_ID)
REFERENCES PROGRAMI (PROGRAM_ID),
```

Obvezno polje (NOT NULL constraint)

- Omejitev za obvezno polje zagotavlja, da bo ustrezni stolpec v tabeli vedno imel vrednost.
- Presledek ali 0 (kot numerična vrednost) nista null-vrednosti!
- Omejitev se definira samo na nivoju stolpca:

```
Naziv VARCHAR(15) CONSTRAINT sola_naziv_NN NOT NULL,
```

Unikatni ključ (unique key constraint)

- Unikatni ključ zahteva, da so vrednosti v ustreznem stolpcu ali skupini stolpcev, edinstvene (*ista vrednost se lahko pojavi samo enkrat*).

Na nivoju stolpca je omejitev definirana kot:

```
DeptName VARCHAR(12) CONSTRAINT dept_deptname_uk UNIQUE,
```

Na nivoju tabele se omejitev definira kot:

```
CONSTRAINT dept_deptname_uk UNIQUE(DeptName),
```

Kontrole (check constraint)

- Kontrola (*CHECK constraint*) definira pogoj, ki mora biti izpolnjen na vsakem zapisu v tabeli.

Na nivoju stolpca se omejitev definira kot:

```
STM_ID NUMBER(2) CONSTRAINT OE_STM_ID_CC
 CHECK((STM_ID >= 10) AND (STM_ID <= 99)),
```

Na nivoju tabele se omejitev definira kot:

```
CONSTRAINT OE_STM_ID_CC
 CHECK((STM_ID >= 10) AND (STM_ID <= 99)),
```

Spreminjanje tabele - ALTER TABLE

- Z ukazom **ALTER TABLE** lahko spreminjamo podatkovne tabele
- Spremembe vključujejo:
  - stolpci/polja (*columns*):
    - dodajanje (*add*),
    - spreminjanje (*modify*),
    - brisanje (*drop*)
  - omejitve (*constraints*):
    - dodajanje (*add*),
    - spreminjanje (*modify*),
    - brisanje (*drop*)

```
ALTER TABLE NazivTabele [SPREMEMBE]
```

- Novi stolpec se v obstoječo tabelo doda z ukazom:

```
ALTER TABLE NazivTabele ADD NazivStolpca TipPodatka;
```

Primer:

```
ALTER TABLE student ADD DAVCNA_STEVILKA VARCHAR(10);
```

- Obstoječi stolpec se v obstoječo tabelo doda z ukazom:

```
ALTER TABLE NazivTabele MODIFY NazivStolpca
NoviTipPodatka;
```

Primer:

```
ALTER TABLE student MODIFY DAVCNA_STEVILKA VARCHAR(8);
```

- Obstoječi stolpec se lahko odstrani iz tabele z ukazom:

```
ALTER TABLE NazivTabele DROP COLUMN NazivStolpca
NoviTipPodatka;
```

Primer:

```
ALTER TABLE student DROP COLUMN DAVCNA_STEVILKA;
```

- Obstoječi stolpec se lahko odstrani iz tabele z ukazom:

```
ALTER TABLE NazivTabele ADD [CONSTRAINT naziv_omejitve
            Tip_omejitve (stolpec, ...)
```

ALTER TABLE - primeri

- Primer:

dodajanje tujega ključa:

```
ALTER TABLE STUDENT
  ADD CONSTRAINT student_servis_ID_FK
  FOREIGN KEY(SERVIS_ID)
  REFERENCES SSERVISI(SERVIS_ID) ON UPDATE
 CASCADE;
```

dodajanje kontrole (*check constraint*):

```
ALTER TABLE EizStatusi
  AVCON_1307099697_STA_I_000
  CHECK (STA_ID IN ('AKT', 'PRE', 'NAP'));
```

Brisanje tabele - DROP TABLE

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

POZOR: DROP ukaz za vedno odstrani tabelo – strukturo in podatke!

Ima dve opciji:

- **CASCADE**: določa, da se pri vseh povezav preko tujih ključev, ki so povzročene z brisanjem tabele, bodo izbrisali ustrezni zapisi v povezani tabeli.
- **RESTRICT**: blokira brisanje tabele, če obstajajo zapisi v tabeli povezani preko tujih ključev.

Indeksi

- Bazni objekti namenjeni hitrem iskanju podatkov
- Za primarni ključ se naredi avtomatično
- Ostale naredimo po potrebi:

```
CREATE [UNIQUE] [ASC] | DESC] INDEX ime_indeksa ON ime_tabele (atribut1, atribut2, ..);
```

- Brisanje sekundarnih indeksov:

```
DROP INDEX ime_indeksa;
```

2024/2025

Dodatne prosojnice