Komponente jezika SQL

SQL jezik - delitev

- DDL - data definition language
  - opredelitev in spreminjanje podatkovnih struktur (meta-podatkov): pod. baza, tabele, pogledi, itn.
- DML - data manipulation language
  - upravljanje s podatki: dodajanje, spreminjanje, izbiranje, brisanje podatkov
- DCL - data control language
  - nadzor dostopa do posameznih objektov (dovoljenja)

SQL jezik - DDL

- DDL ukazi so:
  - CREATE TABLE – ustvarjanje nove tabele
  - ALTER TABLE – spreminjanje obstoječe tabele
  - DROP TABLE – brisanje obstoječe tabele

Ustvarjanje tabele - CREATE TABLE

Tabela se v podatkovni bazi ustvari takoj po uspešno izvedenem ukazu CREATE TABLE.

Splošna sintaksa za ustvarjanje tabele je:

```
CREATE TABLE NazivTabele (

        Stolpec1 TipPodatka [CONSTRAINT
        naziv_omejitve] VrstaOmejitve,

        (Stolpec2 TipPodatka [CONSTRAINT
        naziv_omejitve] VrstaOmejitve,...

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

Opredelitev stolpcev

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

Podatkovni tipi (primer)

TIP	OPIS
CHAR (<velikost>)	za shranjevanje alfanumeričnih znakov (min 0, max 255)
VARCHAR(<velikost>)	za shranjevanje spremenljivega števila znakov (min 0, max 65.535)
INTEGER(p) **FLOAT(p, s)** **DOUBLE(p, s)**	za shranjevanje števil (fiksna in plavajoča vejica). Max. natančnost (p) in/ali decimalni del (s) je 65.
DATE	za shranjevanje datuma in časa, veliko različnih formatov (privzeto LLLL-MM-DD)
RAW (<velikost>)	uporablja se za shranjevanje binarnih podatkov, kot so grafika, zvok in podobno.
CLOB	znakovni (character) veliki objekti, max velikost je 4 GB
BLOB	binarni (binary) veliki objekti, max velikost je 4 GB

Podatkovni tipi: VARCHAR

- VARCHAR je znakovni tip, ki omogoča shranjevanje nizov znakov (alfanumeričnih) ***spremenljive dolžine*** (do določenega maksimuma, ki je določen z velikostjo polja).
- Velikost je določena 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.

Podatkovni tipi: CHAR

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

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

Podatkovni tipi: NUMERIČNI

- INTEGER je celo število brez decimalnega dela.

- NUMBER (FLOAT, DOUBLE) se uporablja za shranjevanje negativnih, pozitivnih, celih števil ter decimalnih števil s fiksno in plavajočo vejico.

- Ko se uporablja tip NUMBER, je potrebno navesti velikost (*angl. precision*) in število decimalnih mest (*angl. scale*):
  - velikost je skupno število števk na levi in desni strani decimalne vejice.
  - število decimalnih mest je število števk na desni strani decimalne vejice.

Primer ustvarjanja tabele

- Primer ustvarjanja tabele STUDENTI:

```
CREATE TABLE STUDENTI
  (STUDENT_ID    CHAR (5),
   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 STUDENTI_STUDENT_ID_PK PRIMARY KEY
  (STUDENT_ID));
```

Standardni podatkovni tipi

- Standard: ISO/IEC 9075
- Podatkovni tipi se v določeni meri razlikujejo med posameznimi SUPB (RDBMS)
- Podatkovni tipi v MySQL:

https://dev.mysql.com/doc/refman/8.0/en/data-types.html

Vrste omejitev

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

1. Integritetne omejitve *(ang. integrity constraints)*: obsegajo primarni ključ in tuje ključe na tabeli ter primarne ključe na tabelah, na katero se tuji ključi sklicujejo.

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

Imenovanje omejitev

- Splošno pravilo za poimenovanje omejitev:

<naziv tabele>_<naziv stolpca>_<vrsta omejitve>

- naziv tabele, na kateri se opredeli omejitev,
- naziv stolpca, na katerega se omejitev nanaša,
- vrsta omejitve je (navadno) okrajšava, ki se uporablja za identificiranje vrste omejitve.

Imenovanje omejitev

- Na primer, omejitev `ZAPOSLENI_ODDELEK_ID_FK` pomeni:
  - omejitev v tabeli ZAPOSLENI na stolpcu ODDELEK_ID,
  - omejitev je vrste *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 vrste *primarni ključ* (PK).

Priljubljene okrajšave za omejitve

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

Določanje omejitev

- Omejitev (*angl. constraint*) se lahko določi:
  - ob ustvarjanju tabele (ukaz `CREATE TABLE` ...),
  - naknadno, na že obstoječi tabeli (ukaz `ALTER TABLE` ...).

- Lahko se je določi na dveh nivojih:
  - na nivoju stolpca (*angl. column level*),
  - na nivoju tabele (*ang. table level*).

Omejitve - nivo stolpca

- Omejitev na nivoju stolpca se nanaša samo na ustrezni stolpec, določi se jo skupaj s samim stolpcem.

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

Sintaksa:

Stolpec TipPodatka **[CONSTRAINT** ***naziv_omejitve*****]**
VrstaOmejitve

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 določijo ločeno od opredelitve stolpcev.
- Navadno se navedejo *po* opredelitvi stolpcev.
- Vse omejitve se lahko opredelijo na nivoju tabele razen omejitve za obvezno polje (NOT NULL constraint).

Sintaksa:

```
[CONSTRAINT naziv_omejitve] Vrsta_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*)
- Tabela lahko ima *samo en* primarni ključ.
- Če tabela uporablja **sestavljeni ključ** (več kot en stolpec), potem se takšen ključ lahko opredeli samo na nivoju tabele.

Primarni ključ – primera

- Na nivoju stolpca je opredelitev 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 s primarnim ključem 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 nimata nujno enakih imen, vrednost tujega ključa pa mora ustrezati vrednosti v nadrejeni (referencirani tabeli) ali biti NULL!

Tuji ključ – primer

- Tuji ključ se ustvari samo na nivoju tabele!

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

```
CONSTRAINT STUDENTI_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 opredeli 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 določena kot:

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

Na nivoju tabele se omejitev določi kot:

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

Kontrole (check constraint)

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

Na nivoju stolpca se omejitev določi kot:

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

Na nivoju tabele se omejitev določi kot:

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

Spreminjanje tabele - ALTER TABLE

- Z ukazom ALTER TABLE lahko spreminjamo zgradbo 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]
```

Dodajanje novega stolpca v tabelo

- Novi stolpec se v obstoječo tabelo doda z ukazom:

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

Primer:

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

Sprememba obstoječega stolpca

- Obstoječi stolpec se spremeni z ukazom:

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

Primer:

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

Brisanje obstoječega stolpca iz tabele

- Obstoječi stolpec se lahko odstrani iz tabele z ukazom:

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

Primer:

```
ALTER TABLE Studenti DROP COLUMN DAVCNA_STEVILKA;
```

Dodajanje omejitve

- Nova omejitev se v obstoječo tabelo doda z ukazom:

```
ALTER TABLE NazivTabele ADD [CONSTRAINT
Naziv_omejitve Vrsta_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;
```