Jezik SQL

- Strukturirani jezik za upite (*Structured Query Language* ili SQL razvio je IBM u sklopu projekta *System R*.
- Poslije je jezik prihvaćen i od drugih većih tvrtki koje su razvijale i prodavale DBMSove. (Oracle, . . .). Danas je jezik ugrađen u sve vodeće DBMS proizvode.
- Tijekom historijskog razvoja, razni DBMSovi su počeli koristiti svoje specifične varijante SQLa.
- Uvodi se standard kako bi se jezik unificirao. 1992 god. uveden je *tz.* ANSI SQL ili SQL-92 ili SQL2. Zadnja verzija predložena je 1999.
- Razvoj SQL standarda traje još i danas.

- Zasnovan je na relacijskom računu: matematička notacija zamjenjena **ključnim riječima** nalik na govorni engleski jezik.

- Sadrži naredbe za:
  - postavljanje upita,
  - modificiranje podataka (upis, promjena, brisanje),
  - definiranje relacija (shema),
  - sortiranje i formatiranje ispisa,
  - aritmetičke operacije s podacima
  - definiranje *pogleda* (*view*) i virtualnih relacija,
  - utjecaj na fizičku građu baze podataka (indeksi)
  - kontrolu sigurnosti.
  - . . .

SQL-92 imena nemaju neka posebna ograničenja na dužinu ili upo-rabu znakova. Ipak, ograničenja postoje u specifičnim verzijama SQL jezika kod DBMSova.

Npr. U Oracle-u, imena tabela i kolona mogu biti dugačka do 30 slova. Moraju počinjati sa slovom, ali mogu zadržavati i brojke, podvučenu crtu (_), znak dolara ($) i povisilice (#). Imena ne bi smjela koristiti rezervirane riječi za druge objekte u bazi podataka.

Broj ograničenja obično se smanjuje s izdavanjem novijih verzija pro-grama.

Predefinirani tipovi atributa

- Tekstualni podaci
  - tekst zadane dužine (character)
  - tekst promjenjive dužine (character varying)
  - tekst proizvoljne dužine
- Numerički podaci
  - mali i veliki cijeli brojevi
  - realni brojevi i realni brojevi povećane preciznosti
- Binarni podaci
- Logički podaci (istina, laž)
- Datumi i vrijeme, vremenski intervali

(Postoji mogućnost definiranja složenih tipova podataka, kao i sasvim novih.)

Općenito SQL se sastoji od naredbi (instrukcija DBMSu) koje se mogu protezati preko nekoliko redaka a završavaju sa znakom **točka-zarez** (;).

Primjeri:

```
CREATE TABLE student (
        prezime VARCHAR(50),
        ime     VARCHAR(50),
        indeks  VARCHAR(10),
        godina  INT,
        smjer   VARCHAR(10),
        PRIMARY KEY (indeks)
);

INSERT INTO student (ime,prezime) VALUES ('imić','prezimić');

DROP TABLE student;

SELECT SQRT(5)+SQRT(3);
```

Izrazi

Naredbe mogu sadržavati izraze u kojima se pojavljuju:

- Logičke operacije: **AND**, **OR** i **NOT**.
- Operacije uspoređivanja: $=$, $<$, $>$, $\geq$, $\leq$, $<>$, te još **IN**, **ANY**, **ALL**, **BETWEEN**, **IS NULL**, **LIKE**, ...
- Skupovne operacije: unija (**UNION**), presjek (**INTERSECT**) i razlika (**EXCEPT**).
- Funkcije na skupovima podataka: broj članova (**COUNT**), zbroj članova (**SUM**), najmanji i najveći (**MIN** i **MAX**), srednja vrijednost (**AVG**).
- Ostale funkcije za rad s podacima.

Izrazi se mogu grupirati pomoću zagrada.

Izrazi mogu sadržavati zadane brojeve, tekstualne podatke i/ili ostale vrste podataka.

Tekstualni izrazi (podaci, konstante) se zatvaraju s jednostrukim navodnim znacima kako bi ih odvojili od samih SQL naredbi.

Primjer:

```
INSERT INTO institut (sifra,ime,adresa)
        VALUES (98,'Institut ''Ruđer Bošković''','Bijenička c. 54, Zagreb');
```

ili

```
INSERT INTO institut (sifra,ime,adresa)
        VALUES (98,'Institut \'Ruđer Bošković\'','Bijenička c. 54, Zagreb');
```

Da bi upisali jednostruki navodni znak možemo koristiti tz. *escape*-znakom što je kosa crta unazad, *backslash* (\), ili upisati jednostruki navodni znak dva puta zaredom.

Brojevi se mogu i ne moraju pisati s navodnim znakovima.
Podaci se mogu pretvarati iz jednog tipa u drugi koristeći **CAST** funkciju.

Definiranje shema

U SQL jeziku relacija nazivamo eng. izrazom *table*.

Relacija se definira s naredbom **create table** koja općenito ima slijedeću strukturu:

CREATE TABLE ime_tabele (
    <defi nicija_ prvog_ atributa> ,
    <defi nicija_ drugog_ atributa> ,
    . . .
    <defi nicija_ zadnjeg_ atributa>,
    <dodatne_defi nicije_ relacije>
);

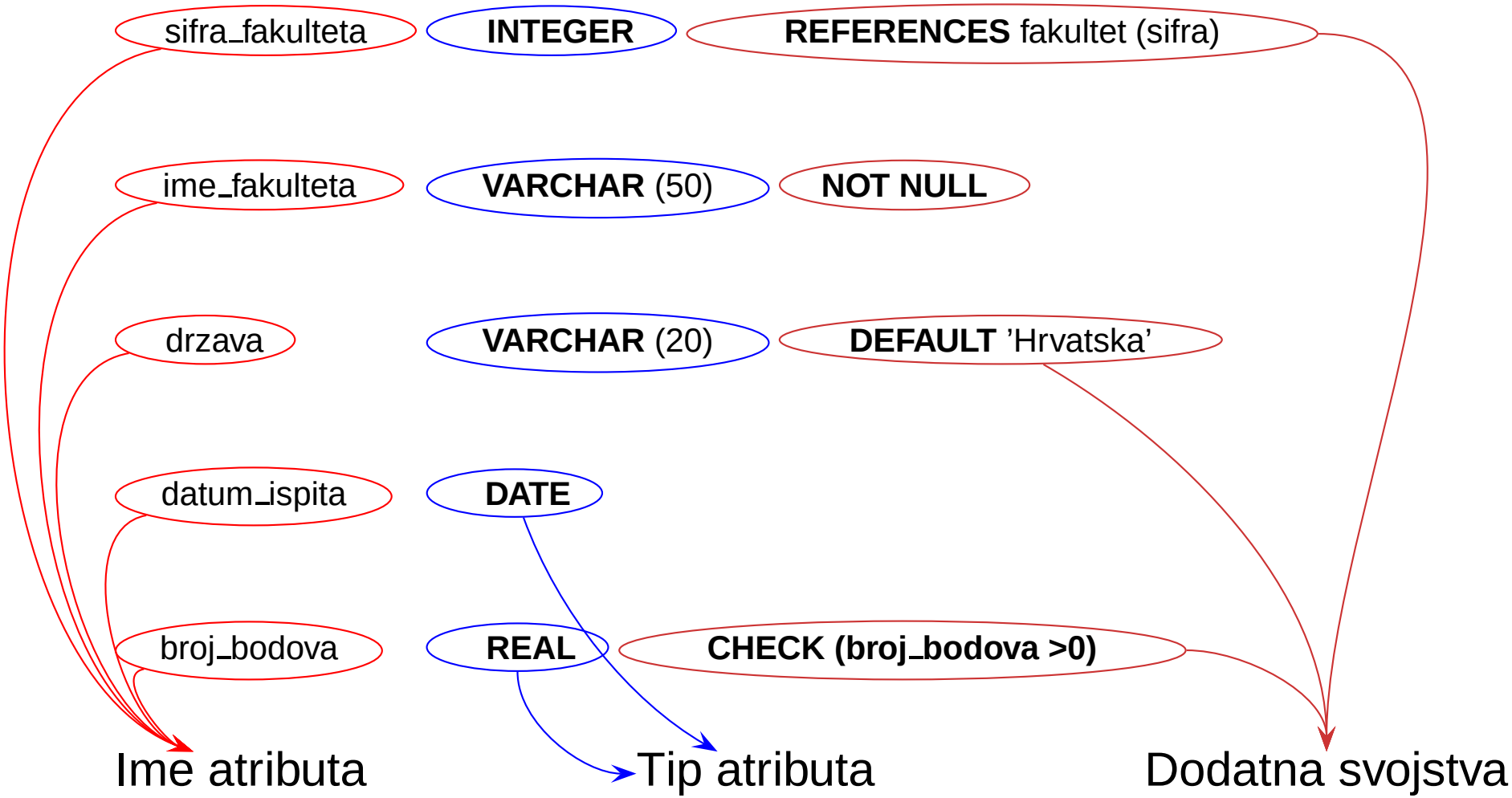
Definicije pojedinih atributa unutar zagrada odvojeni su **zarezima**.

Definicije atributa

Atribut definirano izrazom od dva ili tri dijela:

<ime_atributa> <tip_atributa> <dodana_svojstva_atributa>

Primjeri:



Najčešći tipovi atributa

- Tekstualni tipovi:
  - **CHAR(n)** - tekst dužine n slova
  - **VARCHAR(n)** - tekst dužine najviše n slova
  - **TEXT** - tekst proizvoljne dužine
- Numerički tipovi
  - **INT** - cijeli broj prikazan s 4 bytea
  - **BIGINT** - cijeli broj prikazan s 8 bytea
  - **SMALLINT** - cijeli broj prikazan s 2 bytea
  - **REAL** - realni broj prikazan s 4 bytea
  - **DOUBLE PRECISION** - realni broj prikazan s 8 bytea
  - **NUMERIC [(p,s)]** - realni broj zadane preciznosti
  - **MONEY** - novac
  - **SERIAL** - cijeli broj koji se samo-povećava (auto-inkrementira).

- Binarni podaci:
  - **BYTEA** -
- Logički podaci:
  - **BOOLEAN** - Istina/laž (*true/false, t/n, y/n, 1/0, . . .*)
- Datumi i vrijeme:
  - **DATE** - datum (’2002-10-21’)
  - **TIME** - vrijeme (’04:05:06’)
  - **TIMESTAMP** - datum+vrijeme (’2002-10-21 04:05:06’)
- Razni dodani tipovi izvedeni iz osnovnih tipova

Dodatna svojstva atributa

- **DEFAULT** - zadavanje predefinirane vrijednosti
- **NOT NULL** - vrijednost ne smije biti nepoznata ili nazadana
- **CHECK** - provjera da je vrijednost atributa u zadanim granicama.
- **UNIQUE** - jedinstvenost među *n*-torkama unutar relacije
- **PRIMARY KEY** - primarni ključ
- **REFERENCES** - vrijednost mora biti među vrijednostima atributa iz druge relacije - obično ključ iz druge relacije.
- . . .

Dodatna svojstva relacije

Među dodatnim svojstvima i definicijama jedne relacije mogu se malaziti

- Definiranje primarnog ključa
- Provjeravanje vrijednosti jednog ili više atributa
- ...

```
CREATE TABLE prijatelj (
        ime                VARCHAR(40) NOT NULL,
        adresa             VARCHAR(40),
        mobitel            VARCHAR(10),
        e_mail             VARCHAR(50),
        rodjendan          DATE,
        najdraza_pjesma    VARCHAR(40),
        najdraza_boja      VARCHAR(40),
        najdraze_jelo      VARCHAR(40),
        najdrazi_fi lm      VARCHAR(40)
        PRIMARY KEY (ime)
);
```

Brisanje i mijenjanje relacija

- Brisanje relacije:

```
DROP TABLE <ime_relacije>;
```

- Mijenjanje definicije relacije:
  - Dodavanje atributa:

```
ALTER TABLE prijatelj
    ADD COLUMN telefon VARCHAR(10);
```

  - Brisanje atributa:

```
ALTER TABLE prijatelj
    DROP COLUMN najdraze_jelo;
```

  - Promijeniti atribut:

```
ALTER TABLE prijatelj
    ALTER COLUMN rodjendan SET NOT NULL;
ALTER TABLE prijatelj
    ALTER COLUMN najdraza_boja SET DEFAULT 'plava';
ALTER TABLE prijatelj
    RENAME COLUMN mobitel TO moby;
```

Upisivanje podataka

Za upis podataka se koristi naredba **INSERT INTO**. Postoje dvije varijante:

- Bez navođenja imena atributa:

```
INSERT INTO prijatelj VALUES
    ('Buddy Frendić','Trg noćnih žurki bb.','091-1234567','buddy@frend.net.hr');
```

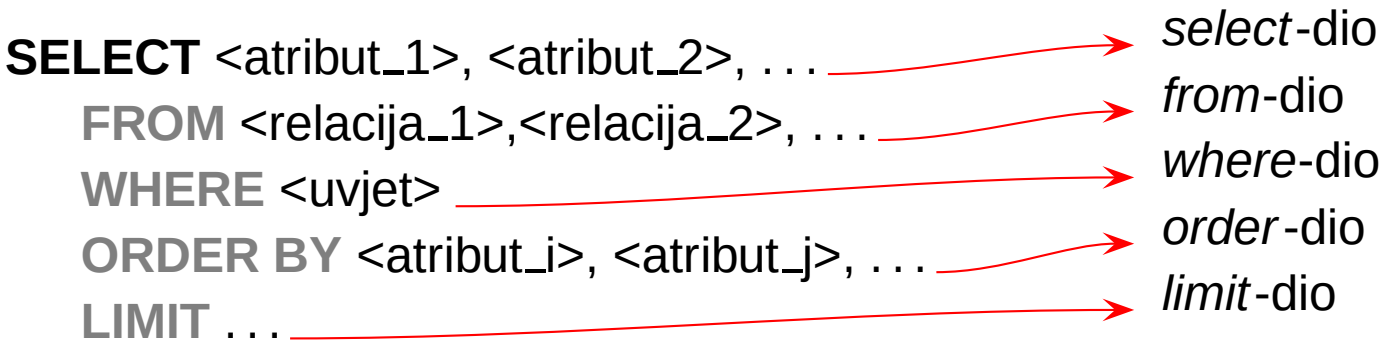
- S navođenjem imena atributa:

```
INSERT INTO prijatelj (adresa, e_mail, ime, najdraze_jelo) VALUES
    ('Bučna ulica 12.','frendi@cavrljanje.hr','Frendi Glasnić','hamburger');
```

Traženje i biranje podataka

Za izbor podataka koji zadovoljavaju određene uvjete koristi se naredba **SELECT**. Naredba **SELECT** ima prilično složenu strukturu.

Jednostavnija verzija ima slijedeću strukturu:



Jedini obavezni dio naredbe je *select*-dio. Ostali dijelovi mogu izostati.

select-dio

U *select*-dijelu se navodi lista atributa čije vrijednosti želimo dobiti. Ako se u *from*-dijelu navodi veći broj relacija, tada imena treba proširiti i s imenima samih relacija u kojima se ti atributi pojavljuju. Zvjezdica se može koristiti kao lista svih atributa iz navedenih relacija.

Primjeri:

SELECT * **FROM** prijatelj . . . ...

SELECT ime, mobitel **FROM** prijatelj . . . ...

SELECT student.ime, prijatelj.mobitel **FROM** student, prijatelj . . . ...

SELECT SQRT(2);

SELECT DISTINCT grad **FROM** student . . .

Rezultat pretraživanja neće sam po sebi izdvojiti različite *n*-torke. Ako se želi dobiti **samo različite** podatke potrebno je to *naglasiti* koristeći **DISTINCT** instrukciju.

from-dio

U *from*-dijelu navodi se lista relacije iz kojih uzimamo podatke.

Primjeri:

```
SELECT student.ime, prijatelj.mobitel FROM student, prijatelj . . . ...

SELECT student,ocjena FROM (student NATURAL JOIN ispit)
    WHERE godina=3 AND kolegij=. . . ;

SELECT a.student, b.student FROM student AS a, student AS b
    WHERE a.rodjendan=b.rodjendan AND a.indeks>b.indeks;
```

Pomoću riječi **AS** može se definirati drugo ime (alias) za relaciju. Uvođenje aliasa je potrebno u pretraživanjima u kojima se uspoređuju atributi *n*-torki jedne te iste relacije. Alias je potrebno koristiti i u situacijama kada je potrebno navesti ime za *virtuelne* relacije (npr. nastale spajanjem realnih relacija ili onih koje se nastale kao rezultat nekog pod-pretraživanja.)

```
SELECT a.ime FROM (SELECT indeks,ime,prezime FROM student) AS a . . .
```

where-dio

U *where*-dijelu treba definirati uvjete koje moraju zadovoljavati *n*-torke relacija čije podatke tražimo. Struktura je:

. . . **WHERE** <uvjet> . . .

Uvjet je logički izraz koji izračunom treba dati vrijednost *Istina* ili *Laž* (*True* ili *False*). Podaci će biti prikazani samo za one *n*-torke koji taj uvjet zadovoljavaju.

- Logički izraz može biti kombinacija manje složenih izraza povezanih logičkim operacijama **AND**, **OR** i **NOT**.
- Logički izraz može sadržavati operacije uspoređivanja vrijednosti atributa s vrijednostima atributa druge relacije ili konstantama.

Primjeri:

. . . **WHERE** student.godina=2;

. . . **WHERE** student.indeks=’F-9876’

. . . **WHERE** (godina=2 **AND** ocjena>2)
OR (godina=4 **AND** ocjena>4);

. . . **WHERE** ime **LIKE** ’Last%’; → Ime koje počine sa slovima "Last"

. . . **WHERE** ime **SIMILAR TO** ’(A|B)%’; → Ime koje počine ili s "A" ili s "B" (SQL-99)

. . . **WHERE** datum > ’2003-09-31’

. . . **WHERE** rodjendan **IS NULL**;

. . . **WHERE** prijatelj.ime
IN (**SELECT** ime **FROM** student **WHERE** godina =2) ;

→ rezultat **samo jedan atribut** istog tipa kao i *prijatelj.ime*!

order-dio

U *order*-dijelu se definira poredak ispisa podataka. Struktura je:

. . . **ORDER BY** <atribut_1>, <atribut_2>, . . .

Ako se želi dobiti uzlazna ili silazna lista tada se to može definirati za svaki atribut posebno, koristeći riječi **ASC** ili **DESC** iza imena atributa.

Primjeri:

ORDER BY ime;

ORDER BY ime,godina DESC;

ORDER BY godina DESC, ime ASC;

limit-dio

- Pomoću instrukcije **LIMIT br** moguće je ograničiti broj pronađenih *n*-torki na manje ili jednako **br**.
- Instrukciju **LIMIT br** može se kombinirati s instrukcijom **OFFSET** s kojom se regulira koji skup od **br** *n*-torki u uređenom nizu će biti dobiven.

Primjer:

```
SELECT indeks, ime FROM student
    ORDER BY indeks DESC LIMIT 1;

SELECT indeks, ime FROM student WHERE godina=1
    ORDER BY ime LIMIT 10;

SELECT indeks, ime FROM student WHERE godina=1
    ORDER BY ime LIMIT 10 OFFSET 10;

SELECT indeks, ime FROM student WHERE godina=1
    ORDER BY ime LIMIT 10 OFFSET 20;
```

Kombiniranje više pretraga

Svako pretraživanje predstavlja samo po sebi jednu viruelnu relaciju. Više raznih međusobno **kompatibilnih** pretraživanja (isti stupanj i imena atributa) mogu se kombinirati koristeci **skupovne** operacije:
Primjeri:

(**SELECT** indeks,ime **FROM** student **WHERE** godina=1)
UNION
(**SELECT** indeks,ime **FROM** student **WHERE** godina=3)

(**SELECT** indeks **FROM** ispit **WHERE** kolegij=9876)
INTERSECT
(**SELECT** indeks **FROM** ispit **WHERE** kolegij=5432)

(**SELECT** indeks,ime **FROM** student)
EXCEPT
(**SELECT** indeks,ime **FROM** student **WHERE** godina=2)

Mjenjanje podataka

Za promjenu već upisanih podataka koristi se naredba **UPDATE**. Struktura onaredbe je općenito oblika:

UPDATE <ime_relacije>
SET <ime_atributa_1>=. . . ,
. . .
<ime_atributa_*n*>=. . .
WHERE . . . → *where*-dio isti kao i u SELECT neredbi

Pomoću *where*-dijela izdvajamo one *n*-torke iz relacije kojima želimo mijenjati podatke.

Primjeri:

UPDATE kolegij **SET** nastavnik='Krešimir Furić' **WHERE** kolegij=2362;

UPDATE ispit **SET** ocjena=5 **WHERE** indeks='F-9876';

UPDATE ispit **SET** ocjena=5, datum='2003-09-23'
WHERE indeks='F-9876' **AND** kolegij=2362;

Brisanje podataka

Za brisanje podataka koristi se naredba **DELETE FROM**. Struktura onaredbe je općenito oblika:

DELETE FROM <ime_relacije>
WHERE . . . → *where*-dio isti kao i u SELECT neredbi

Pomoću *where*-dijela izdvajamo one *n*-torke koje želimo izbaciti iz relacije (obrisati).

Primjeri:

DELETE FROM kolegij **WHERE** naslov='Najteži kolegij';

DELETE FROM ispit **WHERE** indeks='F-9876'

DELETE FROM ispit → OPASNO - briše sve!