SQL - STRUKTURIRANI JEZIK UPITA

SQL - " Structured Query Language" strukturirani jezik za upite.
- razvijen je 1970.g. u IBM Research Laboratory u San Jose-u, California.
- 1981. IBM je predstavio prvi komercijalni SQL proizvod SQL/DS
- SQL je razvijen za rad sa relacionim bazama podataka za koje dr. Codd 1970 godine iznosi 12 Codd-ovih pravila

SQL omogućava da kreiramo i menjamo strukturu baze podataka, dodamo prava korisniku za pristup bazama podataka ili tablelama, da dobijemo informacije od baze podataka i da menjamo sadržnaj baze podataka - odnosno imamo dve vrste funkcija:

- DDL (Data Definition Language) funkcije za definiciju podatka čiji je tipičan primer naredba *CREATE TABLE imeTabele ();*
- DML (Data Manipulation Language) funkcije za upravljanje podacima gdje kao primer možemo navesti osnovnu SQL naredbu *SELECT * FROM imeTabele.*.

Sam pristup podacima odvija se prema modelu klijent / server. To je po Bernardu H. Boar autoru knjige "Implementing Client/server Computing" [(1)] definsano kao :

Klijent/server model je baziran na distribuciji funkcija između dva tipa nezavisnih i autonomnih procesa: servera i klijenta. Klijent je bilo koji proces koji zahteva specifične usluge od server procesa. Server je proces koji obezbeđuje usluge za klijenta. Klijent i server mogu biti smešteni u istom računaru ili u različitim računarima povezanim preko mreže.
U slučaju da su klijent i server procesi smešteni u dva ili više nezavisnih i umreženih računara, server proces može da obezbedi usluge za više od jednog klijenta. Pored toga, klijent može zahtevati usluge i od više servera iz okruženja bez obzira na njihove lokacije ili fizičke karakteristike računara na kojima se nalaze server procesi. Mreža služi da poveže servere i klijente zajedno obezbeđujući medijum kroz koji klijenti i serveri komuniciraju.

SQL ne pravi razliku između malih i velikih slova, što znači da su sledeće dve naredbe jednake:

select prezime from osoba where ime = 'Pera'

ili

SELECT prezime FROM osoba WHERE ime = 'Pera'

Radi lakšeg čitanja koda, preporučuje se da se ključne riječi (naredbe) pišu velikim slovima, svi ostali elementi malim slovima. U nekim bazama niz znakova (string) mora biti napisan kao što je u bazi. Znači u gornjim naredbama nije isto ako piše 'Pera' ili 'PERA'.

Komentari su tekst koji upisujemo kao podsetnik, a koji se neće izvršavati.

Imamo ih dvije vrste:

za samo jedan red

- - ovo je komentar

tj. oznaka za komentar je - - ,a iza sledi tekst komentara ili

komentar kroz više redova

/

komentar kroz više

*redova */*

Za kreiranje baze koristi se naredba :

CREATE DATABASE imeBaze;

Primer pokazuje kreiranje (*CREATE*) i brisanje (*DROP*) novonastale baze podataka.

CREATE DATABASE proba;
DROP DATABASE proba;

isto važi i za tabele.

Pri kreiranju tabela određujemo nazive kolona, tip podatka koji će biti unet. Tipovi podataka su:

Celobrojni:

- **bit** podatak koji je 1 ili 0

- **int** (**integer**) koji iznosi od -2^{31} (-2,147,483,648) do 2^{31}-1

2,147,483,647) smešten u 4 byte-a

- **smallint** celi broj smešten u 2 byte-a; 2^{15} (-32,768) do 2^{15} - 1 (32,767)
- **tinyint** podatak od 0 - 255

Decimalni:

- **decimal** ili **numeric** $-(10)^{38}$ -1 do 10^{38} -1.

Primer je decimal(15, 3);. Prva cifra označava ukupan broj

cifri, a druga broj decimalnih mesta iza decimalne tačke.

Novac:

- **money** tip podatka je isti kao i decimal. Razlika je u zapisu.

-2^{63} (-922,337,203,685,477.5808) do 2^{63} -
1 (+922,337,203,685,477.5807)
- **smallmoney** 214,748.3648 do +214,748.3647

kod novčanog tipa podatka podaci se čuvaju sa četiri decimalna

mesta.

Pomerajući zarez:

- **float** Floating -1.79E + 308 do 1.79E + 308.

- **real** -3.40E + 38 do 3.40E + 38.

Datumi:

- **datetime** 1.avgust, 1753, do 31.decembar, 9999 uz tačnost od 3.33 milisekunde

- **smalldatetime** 1.avgust, 1753, do 31.decembar, 9999 uz tačnost od 1 minute

Nizovi znakova:

- **char (character)** znakovni niz npr. char (9) u bazi će podatak zauzimati 9 znakova bez obzira na unešenu dužinu što znači da može doći do skraćivanja ili dopunjavanja. Maksimalno 8000 znakova.
- **nchar (**national char**)** upisuju se znakovi koji spadaju u Unicode. Maksimalne dužine 4000 znakova.

- **text** unose se tekstualni podaci. Može sadržati 2,147,483,647 znakova.

- **varchar** promenjiva dužina (u bazi se čuva trenutna dužina podatka) ne Unicode znakova. Maksimalne dužine 8000 znakova.
- **nvarchar (**national char varying**)** promenjiva dužina Unicode znakova. Može sadržati 4000 znakova.

Binarni

- **binary** binarni podatak maksimalne dužine 8000 bajtova

- **varbinary** binarni podatak promenljive dužnine. Maksimalne dužine 8000 bajtova.
- **image** binarni podatak promenljive dužine , maksimalne dužine 2,147,483,647 bajtova.

KREIRANJE TABELA

Tabele predstavljaju dvodimenzionalne matrice čiji redovi predstavljaju naziv i svojstva objekata smeštenih u tablelu, a kolone svojstva objekata izranena odgovarajućim tipom podatka. Uz pomoć jedne n-torke opisali smo jedan objekt.

Maticni	Ime	prezime	ulica	mesto
0412974725028	Pera	Milić	Golsfortijeva 21	Novi Sad
2107979715020	Ivan	Matić	Cvijićeva 17	Niš
1503984725028	Marko	Jovanović	Vasina 5	Beograd

```
CREATE TABLE osoba
(
   maticni NVARCHAR(15),
   ime NVARCHAR(15) NOT NULL,
   prezime NVARCHAR(15) NOT
   NULL,
   ulica NVARCHAR(25),
   mesto NVARCHAR(15) DEFAULT
  'Beograd'
   PRIMARY KEY (maticni)
 );
```

U tabeli susrećemo *NOT NULL*. Iz samog naziva vidimo da su to kolone u kojima mora biti nešto upisano, jasno je da upotreba ove naredbe zavisi od dizajna naše tabele odnosno od strukture tabele koju smo mi zamislili tako da će se dalje zapravo objašanjavati dizajn tabele i zašto su nekim kolonama određene dodatne karakteristike.

Za kolone ime i prezime neophodno je da budu upisani neki podaci bez čega inače ta tabela ne bi imala smisla.
Kod adrese je podešeno da default vrednost bude Beograd radi lakšeg upisa podataka jer se očekuje da će najveći broj zapisa imati ovu vrednost.
Kolona maticni je određena kao primarni (osnovni) ključ (PRIMARY KEY (maticni)). Primarni ključ bi trebao da zadovolji kriterijum da

bude jedinstven u čitavoj tabeli (ne postoje dva ista) i da se ostali podaci iz tog reda ne ponavljaju u nekom drugom redu jer bi dovelo do ponavljanja podataka što treba izbeći.

Maticni	ime	prezime	ulica	mesto
0412974725028	Pera	Milić	Golsfortijeva 21	Novi Sad
2107979715020	Ivan	Matić	Cvijićeva 17	Niš
1503984725028	Marko	Jovanović	Vasina 5	Beograd

Vidimo da se broj u koloni matični ne ponavlja znači da je prvi uslov zadovoljen, a on jednoznačno određuje osobu tj. nema osobe koja ima dva matična broja pa smo zadovoljili i drugi uslov.

NAREDBA SELECT

Osnovna naredba u SQL-u je

SELECT izraz FROM imeTabele.

U prevodu *IZABERI izraz IZ imeTabele.*

U naredbi reč *izraz* zamenjujemo sa nazivima kolona koje želimo videti ili u slučaju da želimo videti sve sa *.

*SELECT * FROM imeTabele*

Primer 1

*SELECT * FROM osoba*

maticni	ime	prezime	ulica	mesto
0412974725028	Pera	Milić	Golsfortijeva 21	Novi Sad
2107979715020	Ivan	Matić	Cvijićeva 17	Niš
1503984725028	Marko	Jovanović	Vasina 5	Beograd

Primer 2

SELECT maticni, ime, prezime *FROM osoba*

maticni	ime	prezime
0412974725028	Pera	Milić
2107979715020	Ivan	Matić
1503984725028	Marko	Jovanović

Možemo pored naziva kolone napisati novo ime kolone pod kojim ga želimo videti u izveštaju, ali da u tabeli ostaje sve po starom.

Možemo upisati neki matematički izraz ili tekst unutar navodnika, a u slučaju da imamo više parametara odvajamo ih zarezima.

Primer 3

TabelaPrimanjaRadnika			
IDRadnika	**Plata**	**Prinadležnosti**	**Položaj**
010	75000	15000	rukovodilac
105	65000	15000	rukovodilac
152	60000	15000	rukovodilac
215	60000	12500	rukovodilac
244	50000	12000	činovnik
300	45000	10000	činovnik
335	40000	10000	činovnik
400	32000	7500	pripravnik
441	28000	7500	pripravnik

*SELECT 2 * plata, IDRadnika FROM TabelaPrimanjaRadnika*

Plata	**IDRadnika**
150000	010
130000	105
120000	152
120000	215
100000	244
90000	300
80000	335
64000	400
56000	441

Možemo još spomenuti i ključnu reč *TOP x* koja nam daje samo prvih x redova, što je korisno pri radu na velikim tabelama kad često moramo proveravati rezultate naših izraza.

Primer 4

SELECT TOP 2 ime , prezime naziv FROM osoba

Pera Milić
Ivan Matić

Neka tabela **TabelaAdresaRadnika** sadrži jedinstveni matični broj građana, imena, prezimena i adrese zaposlenih:

TabelaAdresaRadnika					
JMBG	**Ime**	**Prezime**	**Adresa**	**Grad**	**Republika**
512687458	Đorđe	Petrović	Kralja Petra 9	Beograd	Srbija
758420012	Marija	Simić	Bul Nikole Tesle 22	Jagodina	Srbija
102254896	Savo	Jovanović	Njegoševa 17	Podgorica	Crna Gora
876512563	Svetlana	Aćimović	Laze Lazarevića 10	Subotica	Srbija

Pretpostavimo da želite, recimo, da vidite adrese svih zaposlenih. Da biste to postigli, koristite naredbu SELECT:

SELECT Ime, Prezime, Adresa, Grad, Republika FROM TabelaAdresaRadnika;

Rezultat ovog *upita* u bazu podataka je:

Ime	**Prezime**	**Adresa**	**Grad**	**Republika**
Đorđe	Petrović	Kralja Petra 9	Beograd	Srbija
Marija	Simić	Bul Nikole Tesle 22	Jagodina	Srbija
Savo	Jovanović	Njegoševa 17	Podgorica	Crna Gora
Svetlana	Aćimović	Laze Lazarevića 10	Subotica	Srbija

Da objasnimo sada šta ste upravo uradili: tražili ste sve podatke u tabeli TabelaAdresaRadnika – preciznije, tražili ste *kolone* pod nazivom Ime, Prezime, Adresa, Grad, Republika. Obratite pažnju da imena kolona i tabela ne sadrže razmake – ona se moraju navesti kao jedna reč, kao i da se naredba završava tačkom i zarezom (;). Opšti oblik naredbe SELECT kojom se dobijaju svi *redovi* u tabeli je:

```
SELECT ImeKolone, ImeKolone, ...
FROM ImeTabele;
```

Da biste dobili sve kolone neke tabele bez navođenja svih imena kolona, koristite:

```
SELECT * FROM ImeTabele;
```

Svaki sistem za upravljanje bazama podataka i softver za rad sa njima ima različite metode za prijavljivanje na bazu podataka i upisivanje SQL naredbi;

Uslovna selekcija podataka

Da bismo dalje razmotrili naredbu SELECT, pogledajmo drugi primer (hipotetičke) tabele:

TabelaPrimanjaRadnika			
IDRadnika	**Plata**	**Prinadležnosti**	**Položaj**
010	75000	15000	rukovodilac
105	65000	15000	rukovodilac
152	60000	15000	rukovodilac
215	60000	12500	rukovodilac
244	50000	12000	činovnik
300	45000	10000	činovnik
335	40000	10000	činovnik
400	32000	7500	pripravnik
441	28000	7500	pripravnik

Relacioni operatori

U SQL-u postoji šest relacionih operatora i posle njihovog predstavljanja videćemo kako se koriste:

=	Jednako
< ili !=	Različito
<	Manje
>	Veće
<=	Manje ili jednako
>=	Veće ili jednako

Da bi se prikazali samo oni redovi iz tabele koji zadovoljavaju određene kriterijume, koristi se *klauzula WHERE*. Ona se može najlakše razumeti ukoliko se pogleda nekoliko primera.

Ukoliko želite da dobijete ID brojeve onih zaposlenih koji zarađuju preko 50.000, koristite sledeću naredbu:

```
SELECT IDRADNIKA
FROM TABELAPRIMANJARADNIKA
WHERE PLATA >= 50000;
```

Obratite pažnju da se koristi znak >= (veće ili jednako), pošto smo želeli da izdvojimo one zaposlene koji zarađuju više od 50,000, ili jednako 50,000, i to prikazano zajedno. Kao rezultat dobijamo:

```
IDRADNIKA
----------
010
105
152
215
244
```

Opis klauzule WHERE, odnosno deo PLATA >= 50000, naziva se *uslov* (operacija koja kao rezultat daje vrednost True (tačno) ili False (netačno)). Isti tip operacije može se primeniti na tekstualne kolone:

```
SELECT IDRADNIKA
FROM TABELAPRIMANJARADNIKA
WHERE POLOŽAJ = 'rukovodilac';
```

Ova naredba prikazuje ID brojeve svih rukovodilaca. Generalno, u slučaju tekstualnih kolona, koristite operatore jednako ili različito, i obavezno ceo tekst koji se pojavljuje u naredbi navedite unutar apostrofa (').

LOGIČKI OPERATORI AND, OR i NOT

Dalji korak bi bio upotreba logičkih operatora tako da možemo još više proširiti uslove za pretraživanje. Logički operatori su *AND, OR* i *NOT*. Kod složenijih izraza bi trebalo voditi računa o prioritetima. Hijerarhija primjene operatora glasi:

zagrada (),

deljenje / i množenje * ,
sabiranje + i oduzimanje -
, *NOT* (ne),
AND (i
), *OR* (
ili).