DRUŠTVO MATEMATIČARA SRBIJE

AKREDITOVAN SEMINAR:

345
DRŽAVNI SEMINAR O NASTAVI MATEMATIKE I RAČUNARSTVA DRUŠTVA MATEMATIČARA SRBIJE

Kompetencija: K1
Priotiteti: 3

TEMA:

RELACIONE BAZE PODATAKA I SQL

REALIZATORI SEMINARA:

dr BILJANA TEŠIĆ, vanredni profesor na Fakultetu zdravstvenih i poslovnih studija Valjevo, Univerziteta Singidunum u Beogradu
MILIJANA PETROVIĆ, profesor u Srednjoj školi „17. septembar“ u Lajkovcu

BEOGRAD
20 – 21. 02. 2021.

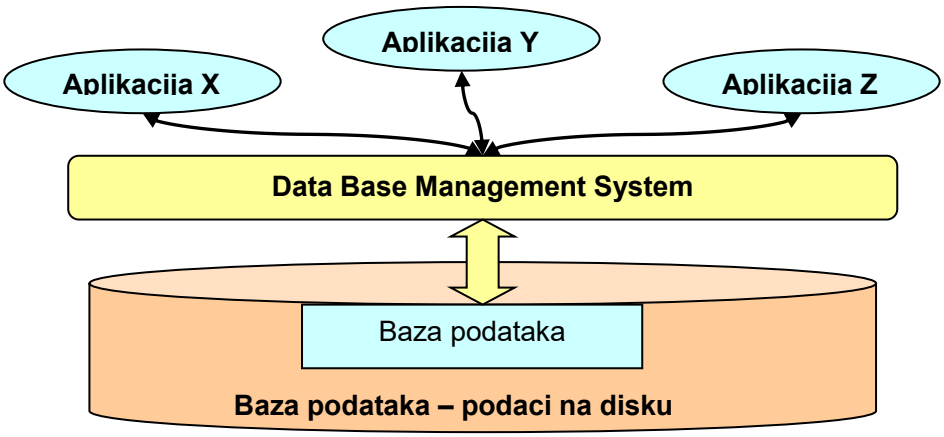
Kao i mnoge druge tehnologije u računarskoj industriji, koreni **relacionih baza podataka** potiču iz IBM-a i njihovog istraživanja automatizovanja kancelarijskih operacija u 60-tim i 70-tim godinama XX veka. 1970. godine, IBM-ov istraživač *Ted Codd* je prezentovao prvi rad o relacionim bazama podataka. Zbog same tehničke prirode rada i oslanjanja na matematički aparat, njegova važnost nije odmah shvaćena. Ipak, doveo je do formiranja IBM-ove istraživačke grupe System R. Od projekta System R se očekivalo da stvori sistem relacione baze podataka koji bi mogao postati proizvod. Prvi prototip prezentovan je 1974/75. godine i eksperimentalno je korišćen.

Većina prvobitnih baza podataka se odnosila na specifične programe napisane za specifične baze podataka. Za razliku od modernih sistema koji mogu biti primenjeni na potpuno različite baze podataka, ovi sistemi su bili usko povezani za bazu podataka da bi osigurali brzinu na uštrb fleksibilnosti. Sistemi upravljanja bazama podataka (**DBMS** - Database Management System) su se prvi put pojavili tokom 1960-tih godina i nastavili su da se razvijaju tokom sledećih decenija. U većini slučajeva, period uvođenja je dugo trajao, skoro deceniju pre navedene godine početka upotrebe. Na primer, relacioni model je prvi put definisan od strane E.F.Codd u tekstu objavljenom 1970. godine. Međutim, relacioni model nije bio široko prihvaćen sve do 1980-tih godina.
Relacioni model baza podataka zasnovan je na matematičkom pojmu relacije. Podaci i veze među podacima se prikazuju preko dvodimenzionalnih tabela.

U srcu relacionog modela nalazi se koncept tabele (koja se naziva i relacija) u kojoj su smešteni svi podaci. Svaka tabela je načinjena od slogova (redova u tabeli), a svaki slog ima svoja polja (atribute). Osnovne karakteristike relacionog modela podataka su sledeće:

- Sve se predstavlja relacijama (tabelama)
- Zasniva se na strogoj matematičkoj teoriji
- Minimalna redudansa podataka
- Jednostavno ažuriranje podataka
- Izbegnute su anomalije ažuriranja
- Redosled kolona i redova ne utiče na informacioni sadržaj tabele
- Ne mogu da egzistiraju dva identična reda (rekorda) u jednoj tabeli
- Svaki red se može jednoznačno odrediti (postoji primarni ključ)
- ...

DBMS je uveden kao interfejs između korisnika (korisničkih programa, aplikacija) i zapisa baze podataka na disku. Korisnički programi ne pristupaju podacima direktno, već komuniciraju sa ovim softverom (programom). DBMS upravlja strukturom baze podataka: definiše objekte baze, njihova svojstva (atribute), dozvoljene vrednosti atributa, veze između objekata, ograničenja nad objektima i međusobnim vezama. Omogućava manipulaciju podacima u bazi: unošenje, brisanje i izmene, tj. omogućava njeno održavanje. Kontroliše pristup podacima: ko može da pristupi podacima, kojim podacima i šta može sa njima da radi.. DBMS dozvoljava deljenje BP između više aplikacija/korisnika i čini upravljanje podacima uspešnijim i delotvornijim Uobičajeno je da kada se govori o softveru za baze podataka, onda se misli upravo na DBMS. DBMS upravlja interakcijom između krajnjih korisnika (aplikacija) i baze podataka. Krajnji korisnici imaju bolji pristup većem broju bolje organizovanih podataka (Slika 1).



Slika 1. DBMS je interfejs između (aplikacija) korisnika i zapisa baze podataka na disku

MySQL je open source sistem za upravljanje relacionim bazama podataka (**RDBMS** - *Relational Database Management System*). Sa stanovišta korisnika, rangiran je kao drugi sistem za upravljanje bazama podataka (posle Oracle Database). MySQL je dostupan i za besplatno preuzimanje, a dostupno je i nekoliko plaćenih verzija koja nude dodatne funkcionalnosti. Deo "My" nazvan je po kćeri suosnivača Majkla Videnius-a, **My**. Standardizovan je 1986. godine pod imenom X3.135, a usvojen je 1987. godine kada ga je prihvatila Međunarodna organizacija za standarde **ISO** - *International Standardization Organization*, kao i **ANSI** - *American National Standards Institute*. Nakon što je SQL standardizovan, nastalo je nekoliko izvedenih i nadograđenih verzija, a razlike u njihovim implementacijama su zanemarljive.

Kao što ime govori, MySQL je zasnovan na SQL-u - standardnom jeziku za upis podataka unutar sistema za upravljanje relacionim bazama podataka. To znači da su podaci smešteni unutar strukture sposobni da prepoznaju odnose između sačuvanih informacija. Svaka baza podataka sadrži tabele (relacije), a svaka tabela sadrži jednu ili više kategorija podataka sačuvanih u kolonama, koji se nazivaju atributima. Redovi se nazivaju zapisi ili slogovi i sadrže jedinstvene podatke (ključeve) za kategorije definisane u kolonama.

MySQL se najviše koristi za web baze podataka. Može se koristiti za smeštanje bilo čega, od jednog zapisa podataka do celog popisa dostupnih proizvoda za Internet prodavnicu. U kombinaciji sa skriptnim jezikom kao što su PHP ili Perl, moguće je kreirati web stranice koje će u realnom vremenu komunicirati sa MySQL bazom podataka kako bi se korisniku web sajta brzo prikazale kategorisane i informacije koje se mogu pretraživati.

SQL (Structured Query Language) predstavlja jezik za rad sa relacionim modelom baza podataka koji su organizovani u logičke skupove i relacije, odnosno veze između njih. Osim relacionog modela, koji se koristi u SQL jeziku, postoje još dva bitna modela baze podataka, a to su model objekti-veze i ER (Entity Relationship) model.

Važno je istaći da SQL spada među jezike koji su najpogodniji i najednostavniji za rad. Za ovu osobinu SQL jezika najzaslužnija je činjenica da se relacije u njemu kreiraju pomoću samo jedne naredbe i, što je još važnije, odmah su dostupne za korišćenje. Svi podaci i rezultati operacija u SQL prikazuju se u vidu tabela koje omogućavaju korisnicima interaktivno i klasično programiranje.

Sastavni deo svakog sistema za upravljanje bazama podataka čine i specijalni jezici za opis i korišćenje baza podataka koji sadrže sledeće sastavne delove: jezik za opis podataka ***(DDL – Data Description Language***), jezik za opis fizičke strukture podataka (***DMCL – Device Media Control Language***), jezik za rad sa podacima (***DML – Data Manipulation Language***) i jezik upita (***QL – Query Languages***). Jedna od najvažnijih osobina baza podataka jeste da je opis podataka potpuno odvojen od programa za obradu podataka.

Jezik za opis podataka (**DDL**) može biti poseban neproceduralni jezik ili proširenje postojećeg programskog jezika (Cobol, C, Pascala ili C++). Njegova osnovna funkcija je specifikacija logičke strukture podataka time što se definišu: objekti i/ili tabele, logičke veze između objekata, atributi i dozvoljeni interval vrednosti za svaki atribut. Jezik za opis fizičke strukture podataka (**DMCL**) definiše: način memorisanja i dodele memorijskog prostora na disku, redosled i fizičku organizaciju podataka, dodelu privremene memorije i načine adresiranja i pretraživanja podataka. Jezik za rad sa podacima (**DML**) obezbeđuje vezu između podataka i aplikacije (programa za rad sa podacima: unos, obradu, prikaz i sl.), a njegove osnovne funkcije su: otvaranje i zatvaranje datoteka (naredbe OPEN i CLOSE), pronalaženje željenog sloga (FIND), izmena sadržaja nekog polja (MODIFY), dodavanje sloga (INSERT), brisanje sloga (DELETE, REMOVE) itd.

Osnovna ili standardna verzija SQL programskog paketa predviđena je za manipulisanje podacima u relacionim bazama podataka i implementirana je u svim programskim paketima za obradu podataka kao njihov sastavni deo. Sintaksa jezika zasniva se na osnovnim relacionim operatorima: spajanje, razlika, množenje, restrikcija i projekcija za ekvivalentne SQL naredbe (Slika 2). Tri preostala relaciona operatora (deljenje, presek i unija) ne smatraju se osnovnim, jer se te operacije mogu izvesti kombinovanjem osnovnih. Jezik SQL je skup naredbi pomoću kojih korisnik određuje šta želi da uradi i ne razmišlja o tome kako će to biti ostvareno. Pri izvršavanju naredbi najbitnija je uloga optimizatora naredbi. Pre izvršenja svake naredbe on određuje optimalan način izvršavanja sa stanovišta brzine i potrebnih resursa. Jednom naredbom moguće

je obraditi grupu slogova, a optimalan pristup grupi bolji je od grupe optimalnih pristupa pojedinačnim slogovima.

Simbol	Naziv	Složenost	Br. operanada
σ	restrikcija	elementarna	unarna
π	projekcija	elementarna	unarna
$\cup$	unija	elementarna	binarna
-	razlika	elementarna	binarna
$\cap$	presek	izvedena	binarna
$\times$	Dekartov proizvod	elementarna	binarna
$\rtimes$	spajanje	izvedena	binarna
/	deljenje	izvedena	binarna

Slika 2. Relaciona algebra – osnovne operacije

Jezici upita (**QL**) omogućavaju realizaciju proizvoljnih funkcija – upita nad relacijama. U zavisnosti od toga da li se zasnivaju na relacionoj algebri ili na predikatskom (relacionom) računu, postoje dve klase ovih jezika. U relacionoj algebri definisane su operacije pomoću kojih je moguće dobiti željenu relaciju (tabelu) iz skupa datih relacija (tabela). Relacionim računom definišu se osobine relacija koje se žele dobiti. U relacionoj algebri definisane su sledeće operacije: unija, diferencija, presek, Kartezijev (Dekartov) proizvod, projekcija, selekcija (restrikcija), spajanje (join), deljenje. Postoje još neke izvedene operacije uslovljene **NULL** vrednostima. Relacioni račun je neproceduralni jezik, te programer umesto da definiše proceduru pomoću koje će se dobiti željeni rezultat, samo specificira željeni rezultat. SQL je programski paket koji se zasniva na relacionoj algebri i relacionom računu.

Osnovne karakteristike SQL-a su:

- **Jednostavnost i jednoobraznost pri korišćenju** - Ogleda se u tome da se Tabela (relacija) kreira jednom izvršnom naredbom i da je odmah dostupna za korišćenje. Svi podaci su memorisani u tabelama, i rezultat bilo koje operacije, logički se prikazuju u obliku tabele.

- **Mogućnost interaktivnog i klasičnog (aplikativnog) programiranja -** Koristeći SQL dobijaju se odgovori na trenutne, unapred nepredvidene zahteve ili se SQL blokovi "ugrađuju" u klasični viši programski jezik (FORTRAN, COBOL, PL/I, C), omogućujući klasičnu obradu gde korisnik same aplikacije najčešće uopšte nije ni svestan da koristi SQL.

- **Neproceduralnost** - Ni za jedan jezik se ne može reći da je potpuno neproceduralan, već da je neproceduralan u većem ili manjem stepenu. SQL je u velikoj meri neproceduralan jer definiše **ŠTA**, a ne **KAKO**: koji podaci se žele, koje tabele se referenciraju i koji uslovi treba da budu ispunjeni, bez precizne specifikacije procedure za dobijanje željenih podataka. Preciznije rečeno SQL je na višem nivou apstrakcije nego klasični viši programski jezici, odnosno mnogi podjezici relacionih sistema za upravljanje bazama podataka koji se često koriste.

Tipovi podataka koje podržava MySQL mogu se svrstati u tri kategorije, a dozvoljava proširenja podataka većeg obima. Dodatno se mogu definisati tip podataka i skup vrednosti. Osnovni tipovi podataka su:

- Numerički,
- Vremenski.
- Binarni i tekstualni.

Numeričke tipove podataka čine celobrojne vrednosti (INTEGER, SMALLINT, DECIMAL i NUMERIC) kao i aproksimativne vrednosti (FLOAT, REAL i DOUBLE PRECISION). Sinonim za INTEGER je INT dok je sinonim za DECIMAL DEC. Na **Slici 3.** prikazani su **Numerički** tipovi podataka.

TIP PODATKA	OPIS
TINYINT( )	-128 do 127 **označeni** 0 to 255 **neoznačeni**
SMALLINT( )	-32 768 do 32 767 **označeni** 0 do 65 535 **neoznačeni**
MEDIUMINT( )	-8 388 608 do 8 388 607 označeni 0 do 16 777 215 neoznačeni
INT( )	-2 147 483 648 do 2 147 483 647 označeni 0 do 4 294 967 295 neoznačeni
BIGINT( )	-9 223 372 036 854 775 808 do 9 223 372 036 854 775 807 označeni 0 do 18 446 744 073 709 551 615 neoznačeni
FLOAT	Mali broj u pokretnom zarezu
DOUBLE(,)	Veliki broj u pokretnom zarezu
DECIMAL(,)	Broj tipa DOUBLE sačuvan kao string, koji dozvoljava fiksan decimalni zarez

Slika 3. Numerički tipovi podataka

Tipovi podataka za datum i vreme imaju sopstveni skup vrednosti i "nula" vrednost koja zamenjuje unešene vrednosti koje nisu validne. Na Slici 4. prikazan je format podataka za datum i vreme:

TIP PODATKA	OPIS
DATE	Datum u formatu YYYY-MM-DD.
DATETIME	Datum i vreme u formatu YYYY-MM-DD HH:MM:SS.
TIMESTAMP	Datum i vreme u formatu YYYYMMDDHHMMSS
TIME	Vreme u formatu HH:MM:SS.

Slika 4. Format podataka za datum i vreme

Binarni i tekstualni tipovi podataka namenjeni su za skladištenje binarnih i tekstualnih vrednosti to su:

- CHAR i VARCHAR - za skladištenje kraćih nizova karaktera,
- BINARY i VARBINARY – za skladištenje manjih količina binarnih podataka,
- BLOB i TEXT - za skladištenje binarnih nizova različite veličine,
- ENUM i SET - za skladištenje tekstualnih vrednosti koje su na listi vrednosti koja se kreira pri definisanju polja (kod ENUM je lista dozvoljenih elemenata veća u odnosu na tip SET).

DDL (Data Definition Language) - Naredbe za definisanje podataka

Ove naredbe omogućuju definisanje resursa relacione baze podataka: strukturu BP, tabele, atribute, tipove podataka, ograničenja, pomoćne indekse za direktan pristup itd. SUBP-a treba da omogući da izvršimo sledeće operacije:

1. kreiranje baze podataka,
2. kreiranje tabele baze podataka,
3. kreiranje indeksa nad kombinacijom kolona tabele,
4. kreiranje virtuelne tabele - "pogleda",
5. izmena definicije tabele,
6. izmena pogleda u bazi podataka,
7. promena imena tabeli u bazi podataka,
8. brisanje tabele iz baze podataka,
9. uklanjanje indeksa iz tabele,
10. uklanjanje baze podataka,

Osnovne naredbe za definisanje podataka su:

- **CRATE DATABASE** - kreiranje nove prazne baze sa svim potrebnim objektima
- **CREATE TABLE**– kreiranje fizičke tabele baze podataka, sa akcentom na tipove podataka, i kreiranje primarnih i sekundarnih ključeva (PRIMARY KEY, FOREIGN KEY)

- **CREATE VIEW** – kreiranje virtuelne imenovane tabele, “pogled”,
- **CREATE INDEX** – kreiranje indeksa nad jednom ili više kolona tabele ili pogleda,
- **ALTER TABLE** – izmena definicije tabele, izmena, dodavanje ili uklanjanje kolone (atributa) u tabeli baze podataka,
- **DROP DATABASE** - uklanjenje kompletne baze podataka koja je prethodno kreirana (ovom naredbom uklanjamo čitavu bazu podataka i sve njene objekte: tabele, atribute, tipove podataka, ograničenja, indekse itd.),
- **DROP TABLE** – uklanjanje tabele iz baze podataka,
- **DROP VIEW** – uklanjanje pogleda iz baze podataka,
- **RENAME TABLE** - promena imena neke od tabela koja je prethodno kreirana u bazi podataka.
- **DELETE FROM TABLE** - brisanje svih podataka iz tabele ali se prazna tabela i dalje čuva u bazi podataka tako da se kasnije ponovo može koristiti

DML (Data Manipulation Language) – Naredbe za upravljanje podacima

Osnovni razlog korišćenja **S**istema za **U**pravljanje **B**azama **P**odataka (SUBP) je jednostavan i lak rad sa podacima koji se nalaze u bazi podataka. Pod upravljanjem podacima podrazumeva se:

1. Unos (dodavanje) podataka
2. Pregled (korišćenje) podataka
3. Izmena podataka
4. Uklanjanje podataka

Naredbe za rukovanje podacima (Data Manipulation Statements) omogućavaju ažuriranje podataka u širem smislu (izmenu, dodavanje i brisanje) i izveštavanje (pribavljanje postojećih i izračunavanje novih informacija) iz baze podataka:

- **SELECT** – pristup podacima i pregled sadržaja baze podataka,
- **INSERT** – unošenje podataka, dodavanje redova u tabelu,
- **DELETE** – brisanje podataka, izbacivanje redova (zapisa) iz tabele,
- **UPDATE** – ažuriranje, izmena vrednosti podataka u koloni, uklanjanje vrednosti parametara zapisa

DCL (Data Control Language) – Naredbe za kontrolu pristupa podacima

Kontrola pristupa podacima obično podrazumeva mogućnost kreiranja više korisničkih profila (naloga) sa određenim pristupnim podacima (korisničko ime i lozinka) i pridruženim privilegijama. MySql SUBP poseduje interni sistem korisničkih naloga (sa parametrima: korisničko ime, lozinka, mrežna lokacija klijenta) sa privilegijama na nivou jedne ili više baza, tabela i/ili kolona. Podaci vezani za korisničke naloge i privilegije su skladišteni u sistemskoj bazi "mysql". Osnovne naredbe za rad sa korisničkim nalozima su:

- **CREATE USER -** kreiranje novih korisničkih naloga u SUBP. Pravo na korišćenje ove naredbe ima administrator Sistema, kao i korisnici koji imaju globalno pravo na ovu privilegiju i korisnici koji imaju INSERT privilegiju za sistemsku **mysql** bazu podataka.
- **RENAME USER -** izmena postojećih korisničkih naloga u SUBP. Pravo na korišćenje ove naredbe ima administrator sistema kao i korisnici koji imaju globalno pravo na CREATE USER privilegiju i korisnici koji imaju UPDATE privilegiju za sistemsku **mysql** bazu podataka.
- **DROP USER -** brisanje postojećih korisničkih naloga u SUBP. Pravo na korišćenje ove naredbe ima administrator Sistema, kao i korisnici koji imaju globalno pravo na CREATE USER privilegiju i korisnici koji imaju DELETE privilegiju za sistemsku **mysql** bazu podataka.
- **SET PASSWORD -** promena pristupne lozinke postojećih korisničkih naloga u SUBP. Ovom naredbom je moguće promeniti lozinku za sopstveni nalog, kao i lozinku za ostale korisničke naloge, ukoliko aktivni korisnički nalog ima UPDATE privilegiju za sistemsku **mysql** bazu podataka.

Osnovne naredbe za rad sa privilegijama su:

- **GRANT -** dodela prava korišćenja tabele drugim korisnicima od strane vlasnika tabele,

- **REVOKE -** oduzimanje prava korišćenja tabele drugim korisnicima,
- **FLUSH PRIVILEGES -** učitavanje parametara vezanih za privilegije.

Naredba GRANT omogućava administratorima SUBP-a da određenom koričkom nalogu dodele određene privliegije nad određenim objektima baze podataka. Ukoliko se privilegije dodeljuju nalogu koji ne postoji u SUBP, on se automatski kreira, tako da je ovom naredbom moguće i kreirati nove korisničke naloge. Za korišćenje ove naredbe neophodno je biti administrator Sistema, ili imati **GRANT OPTION** privilegiju kojom je moguće dodeliti privilegije iz skupa privilegija koje su dodeljene aktivnom korisničkom nalogu.
Naredba REVOKE omogućava administratorima SUBP-a da određenom koričkom nalogu oduzmu određene privliegije nad određenim objektima baze podataka. Za korišćenje ove naredbe neophodno je biti administrator Sistema, ili imati GRANT OPTION privilegiju kojom je moguće oduzeti privilegije iz skupa privilegija koje su dodeljene aktivnom korisničkom nalogu.
Naredba FLUSH služi za ponovo učitavanje parametara sistema koji se nalaze keširani u memoriji. Za izvršavanje ove naredbe potrebno je imati RELOAD privilegiju.

SQL Upiti

SQL naredba upita **SELECT -** Najznačajnija i najčešće korišćena SQL naredba za manipulaciju podacima. Kod svakog upita zadajemo (u principu): koje podatke tražimo kao rezultat, iz kojih tabela to tražimo, koji uslov treba da zadovolje podaci da bi bili uključeni u rezultat, u kom redosledu želimo prikaz podataka.

Prost upit nad jednom tabelom

Podrazumeva se naredba upita SELECT, nad jednom tabelom. Kao rezultat daje niz redova (ili jedan ili nijedan) koji zadovoljavaju eventualno zadati uslov (**WHERE**). **SELECT lista** – podrazumeva se specifikacija podataka u rezultatu upita. **Specifikacija** – zadata jednim ili sa više izraza odvojenih zarezima (R-lista). Rezultat upita ne mora biti relacija (unikatnost).

Sintaksa za SELECT (prost upit nad jednom tabelom):

```
SELECT * | {[ALL | DISTINCT] R-Lista}
  FROM ImeTabele
  [WHERE R-Predikat]
  [ORDER BY  ImeKolone [DESC]
          {, ImeKolone [DESC]} …];
```

- ***** - specijalni slučaj R-liste, kada u rezultat želimo da uključimo sve kolone tabele,
- **ALL** – iz rezultata ne uklanja istovetne redove,
- **DISTINCT** – iz rezultata uklanja istovetne redove,
- **R-Lista** – zadaje se kao jedan ili više R-Izraza, pored naziva kolone javljaju se i konstante,
- Operacije poređenja kod klauzule **WHERE** mogu biti u formi relacionog izraza:

Izraz1 {<,<=, =, <>, >=, >} Izraz2

Takođe, odredba **WHERE** (gde) omogućuje složene logičke izraze koji se prave pomoću binarnih logičkih operacija:

- izdvajanje (selekciju) redova koji zadovoljavaju neki uslov,
- izdvajanje redova koji zadovoljavaju više uslova (**AND**),
- izdvajanje redova koji zadovoljavaju bar jedan od uslova (**OR**),
- izdvajanje redova koji zadovoljavaju složene uslove (**AND** i **OR**),
- izdvajanje redova čija je vrednost unutar nekih granica (**BETWEEN**),
- izdvajanje redova čija vrednost pripada nekoj listi vrednosti (**IN**),
- izdvajanje redova koji ne zadovoljavaju neke uslove (**NOT**, **IS NOT**),
- izdvajanje redova ako neka vrednost postoji (**EXISTS**) itd.

Klauzula **FROM** ("odakle"), je obavezna klauzula kojom se specificira ImeTabele (to je ime osnovne tabele ili pogleda nad kojim se vrši upit.
R_Predikat je uslov prikazivanja rezultata, i predstavlja logički izraz izračunljiv nad svakim pojedinim redom tabele. Rezultat upita se dobija samo za one vrednosti R_Predikata koje daju istinitosnu vrednost. Najčešće je to relacioni izraz (**>,<,=,…**) sa kolonama, a sa desne strane može se javiti i konstanta.
ORDER BY – daje željeni redosled prikaza rezultata. Podrazumeva se rastući redosled (ASC). U suprotnom se navodi DESC uz odgovarajuću kolonu. Uvek je poslednja klauzula u SELECT bloku.

Kod upita se obično traži prikaz samo određenih kolona, ili prikaz svih kolona u redosledu koji je drugačije određen. Ovo odgovara operaciji projekcije, ali se ne elimišu višestruka ponavljanja istih vrednosti

Prost upit nad jednom tabelom sa svodnim rezultatom

Sintaksa za SELECT (prost upit nad jednom T sa izvedenim rezultatom):

```
SELECT G-Lista
                FROM ImeTabele
                [WHERE R-Predikat];
```

G-Izrazi - najčešće ih čine posebne SQL funkcije (svodne ili agregatne funkcije).

Svodne funkcije:

- **SUM** (ImeKolone) Nalazi sumu svih ne-NULL vrednosti zadate kolone
- **AVG** (ImeKolone) Nalazi prosečnu vrednost svih ne-NULL vrednosti zadate kolone
- **MIN** (ImeKolone) Nalazi minimalnu vrednost svih ne-NULL vrednosti zadate kolone
- **MAX** (ImeKolone) Nalazi maksimalnu vrednost svih ne-NULL vrednosti zadate kolone
- **COUNT(*)** Nalazi ukupan broj redova u tabeli
- **COUNT** - bez DISTINCT nalazi ukupan broj ne-NULL vrednosti zadate kombinacije kolona, dok sa DISTINCT nalazi ukupan broj različitih ne-NULL vrednosti zadate kombinacije kolona.

Svodni upit nad jednom tabelom

Sintaksa za SELECT za svodni upit nad jednom tabelom:

```
SELECT ListaKolona [ListaFunkcija]
        FROM ImeTabele
        [WHERE R-Predikat]
        GROUP BY ListaKolona
        HAVING G-Predikat
        [ORDER BY  Element [DESC]
                {, Element [DESC]} …];
```

Napomene:

- **ListaFunkcija** - čine je jedna ili više agregatnih funkcija
- **WHERE -** zadaje se uslov koji svaki red u tabeli ImeTabele mora da zadovolji
- **GROUP BY -** navodi se jedna ili više kolona po kojima se vrši grupisanje (kolone koje se navode ne moraju biti uz SELECT, kolone koje su uz SELECT moraju se naći uz GROUP BY
- **HAVING** - formira se uslov koji svaki red formiran svođenjem mora da zadovolji da bi bio uključen u rezultat (mogu da se jave kolone i funkcije koje nisu uz SELECT)

Upiti nad više tabela

Podrazumevaju spajanje tabela po nekom uslovu. Iza FROM klauzule SELECT naredbe navodi se više tabela odvojenih zarezima. Za kolone koje se nalaze u više tabela obavezno je navođenje:

- ✓ *ImeTabele.ImeKolone*
- ✓ *NadimakTabele.ImeKolone*

R-Predikat- navodi se uslov spajanja u formi uslova jednakosti vrednosti odgovarajućih kolona u tabelama.
Upit nad više tabela bez uslova spajanja daje kao rezultat Dekartov proizvod tih tabela.

Upiti sa podupitima

Podrazumeva SELECT naredbu koja se nalazi u sklopu WHERE i HAVING klauzula. Izvršavanje podupita prethodi vrednovanju predikata u datim klauzulama.
Klasifikacija podupita prema rezultatu:

- ✓ S-Upit: Skalarni upit (kao rezultat se dobija jedna vrednost)
- ✓ K-Upit – kolonski upit
- ✓ R-Upit – upit opšteg tipa