# Fizički dizajn baze podataka

**Bošnjak, Josip**

**Undergraduate thesis / Završni rad**

**2015**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Pula / Sveučilište Jurja Dobrile u Puli**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:137:960942

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-04-24**

*Repository / Repozitorij:*

Digital Repository Juraj Dobrila University of Pula

Sveučilište Jurja Dobrile u Puli
Fakultet ekonomije i turizma
«Dr. Mijo Mirković»

**Josip Bošnjak**

# Fizički dizajn baze podataka

Završni rad

Pula, 2015.

Sveučilište Jurja Dobrile u Puli
Fakultet ekonomije i turizma
«Dr. Mijo Mirković»

**Josip Bošnjak**

# Fizički dizajn baze podataka

Završni rad

**JMBAG: 0303038620, redoviti student**

**Studijski smjer: Informatika**

**Predmet:Baza Podataka I**

**Mentor:Prof.dr.sc.Vanja Bevanda**

Pula, rujan 2015.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani _____, kandidat za prvostupnika _____ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student:

U Puli, **. **. 2015. _____

# Sadržaj


Uvod ..... 5
1.Baza podataka ..... 6
1.1.Sustav za upravljanje bazama podataka ..... 7
1.2.Jezici u sustavima za upravljanje bazama podataka ..... 8
1.3.Organizacija podataka i arhitektura baze podataka ..... 9
1.4.Ciljevi korištenja baza podataka ..... 10
2. Fizička građa i SQL standard ..... 11
2.1.SQL standard ..... 11
2.2.Fizička građa baze podataka ..... 12
2.3.Organizacija datoteke ..... 14
2.4.Organizacija indeksa ..... 19
3.Stvaranje početne verzije fizičke sheme ..... 21
3.1. Kreiranje indeksa ..... 23
3.2. Stvaranje i punjenje baze podataka ..... 25
3.3.Postavljanje upita ..... 26
3.4. Jednostavni upiti ..... 26
3.5.Složeniji upiti ..... 28
3.6. Grupirajući upiti ..... 28
4. Primjer fizičkog dizajna s alatom ORACLE SQL DEVELOPER ..... 30
Zaključak ..... 37
6.Popis tablica ..... 38
7.Popis slika ..... 39
Sažetak ..... 40
Summary ..... 41
8.Literatura ..... 42

# Uvod

Baza podataka predstavlja određeni objedninjeni skup podataka. Podaci se mogu objediniti i na papirnatom i digitalnom obliku. Danas se najviše koristi digitalno objedinjavanje podataka preko programa koji omogućuje upravljanje bazama podataka a naziva se sustav za upravljanje bazama podataka. Prvo je potrebno definirati podatak, znanje i informaciju. „Podatak je skup znakova zapisanih na određenom mediju, npr. papiru, filmu, magnetskom mediju. Možemo reći da je podatak neobrađena činjenica, niz znakova, slika, koncepata i instrukcija. Informacija je činjenica s određenim značenjem. Ona donosi novost, o nečemu obaviještava, otklanja neizvjesnost i služi kao podloga za odlučivanje. Znanje je uređeni skup informacija o nekom području. Svako područje organizira svoje znanje utvrđivanjem prikladnih koncepata, njihovih međusobnih odnosa i ograničenja.“ (Privrat, 2011). U ovom radu biti će opisan sustav za upravljanje bazama podataka, te kako će fizički dizajn baze podataka izgledati. U prvom poglavlju biti će prikazane definicije baze podataka i sustava za upravljanje bazama podataka. Potrebno je objasniti i organizaciju podataka te ciljeve korištenja baza podataka. U drugom poglavlju, biti će objašnjen sql standard, te će biti prikazana fizička organizacija datoteka i indeksa te građa baze podataka. U trećem poglavlju potrebno je kreirati fizičku bazu podataka sa inicijalnim punjenjem, gdje će biti korišteni DDL i DML jezici. U četvrtom poglavlju, biti će prikazane tri vrste upita, od jednostavnih, složenih pa sve do grupirajućih upita. Upiti će poslužiti za provjeru funkcionalnosti fizičke sheme. U zadnja dva poglavlja, biti će prikazana dva primjera fizičkog dizajna, jedan vezan za bazu podataka fakulteta, koji se nalazi u literaturi, i drugi primjer, baza podataka knjižnice, kojeg nema u korištenim literaturama.

# 1.Baza podataka

„Baza podataka je skup međusobno povezanih podataka,pohranjenih zajedno bez štetne ili nepotrebne zalihosti, a glavni im je cilj da ih koriste različite aplikacije. Podaci su pohranjeni u obliku naovisnom od programa koji ih koriste. Unos,izmjena i dohvat podataka obavlja se kroz zajedničko i kontrolirano sučelje.“ (Baranović & Zakošek, 2007). „ Glavni cilj uvođenja baze podataka je ubrzanje računalnih aplikacija,smanjenje troškova njihova održavanja i opskrbljivanje krajnjih korisnika podacima potrebnima da bi svoj posao obavili što je moguće efikasnije.“ (Šehanović , et al., 2002.). Koncepcija baze podataka ima svoju misaonu i fizičku logiku. Uporaba baze podataka podrazumijeva postojanje triju razina korisnika,i na svakoj od njih su razvijeni odgovarajući jezici i upute za obradu: 1.Krajnji korisnik,2.Kreator baze podataka i 3.Administrator baze podataka. „ Krajnji korisnik je analitičar koji se služi jezicima za pretraživanje. Kreator baze podataka je fizički oblikuje na način da izrađuje aplikacije za njezinu uporabu. On mora detaljno poznavati sustav upravljanja bazom podataka i radi na unaprijeđivanju njezinog funkcioniranja. Administrator baze podataka usko surađuje s krajnjim korisnikom i rabi posebne instrukcije i jezike za pretraživanje i rukovanje bazom podataka. On sudjeluje u kreiranju baze podataka,uspostavljanje njezine strukture i u svim promjenama koje se s tim u vezi javljaju. Sustav upravljanja podacima je shematski prikaz upravljačkog sustava baze podataka. Taj softverski sustav omogućava da baza podataka funkcionira u operativnom smislu,da se mijenja,usavršava i kontrolira. Danas na tržištu postoje brojna softverska rješenja za takve upravljačke sustave. Osnovna ideja tehnologije baza podataka je u tome da pojedina aplikacija ne stvara svoje vlastite datoteke na disku. Umjesto toga,sve aplikacije rade zajedničku i objedinjenu kolekciju podataka.“ (Šehanović , et al., 2002.). Aplikacija ne pristupa izravno podacima na disku. Ona barata sa podacima na posredan način, služeći se uslugama specijaliziranog softvera koji je zadužen da se brine za zajedničku kolekciju. Spomenuta zajednička kolekcija podataka je baš ovaj pojam,baza podataka. Specijalizirani softver koji posreduje između aplikacija i podataka naziva se sustav za upravljanje bazama podataka. Baza podataka je tehnologija koja je nastala s namjerom da ukloni slabosti tradicionalne automatske obrade podataka iz 60-ih i 70-ih godina 20.stoljeća.Ta tehnologija osigurala je veću produktivnost, kvalitetu i pouzdanost u razvoju aplikacija koje se temelje na pohranjivanju i pretraživanju podataka u računalu.

### 1.1.Sustav za upravljanje bazama podataka

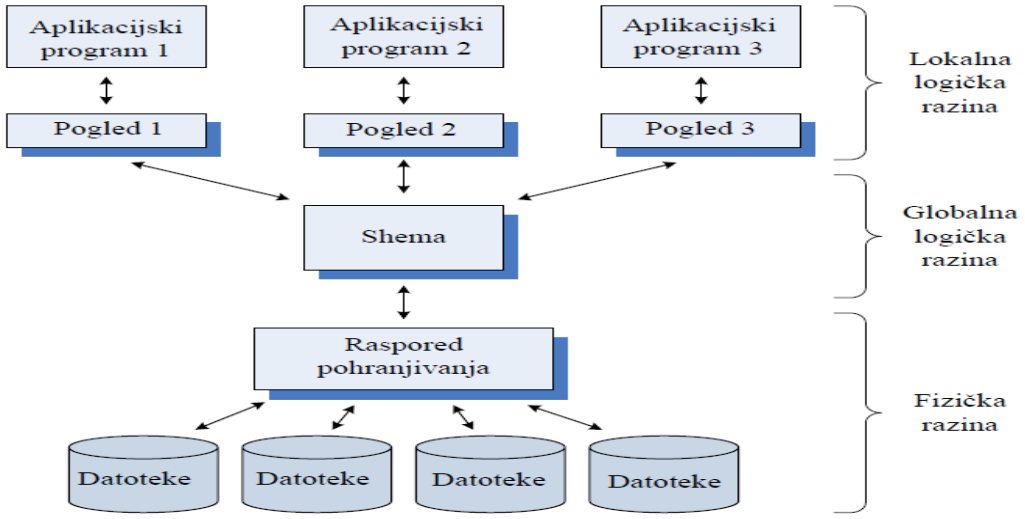
„Programski sustav koji omogućava upravljanje bazom podataka je sustav za upravljanje bazama podataka,SUBP (DBMS-Database Managment System). Korisnik ili korisnički program postavlja zahtjev za obavljanjem neke operacije s podacima, a SUBP ga analizira,provjerava,optimizira,transformira u niz operacija koje je potrebno obaviti na fizičkoj razini,obavlja operacije i vraća rezultat. Najvažnije zadaće SUBP-a je zaštititi bazu podataka od neovlaštenog korištenja (dana security), spriječiti narušavanja pravila integriteta (data integrity),osigurati obnovu podataka u slučaju uništenja (recovery), spriječiti štetne interferencije korisnika u višekorisničkim sustavima (concurrency),omogućiti korištenje rječnika podataka (podaci o podacima) te optimizirati sve funkcije i obavljati ih efikasno koliko je to moguće.“ (Manger, 2011.). Sustavi za upravljanje bazama podataka obično se kategoriziraju prema modelu podataka koji podržavaju:odnosni, orjentirani prema objektu,mrežni…Veliki dio SUBP je ipak neovisan o modelu podataka, te je zaokupljen upravljanjem faktorima poput performansi, podudarnosti, integriteta i obnove nakon hardverskih propusta. Slično kao i operacijski sustav,DBMS spada u temeljni softver kojeg većina korisnika i organizacija ne razvija samostalno, već ga kupuju zajedno sa računalom. Danas postoji nekoliko važnih i široko zastupljenih DBMS-a : DB2,ORACLE, MS SQL SERVER, MYSQL. Svi ovi proizvodi uključuju u sebi i dodatne alate za razvoj aplikacija,administriranje baze i slično. „ DB2 je proizvod IBM-a, namjenjen prvenstveno velikim mainframe računalima. ORACLE je proizvod istoimene tvrtke koje pokriva gotovo sve računalne platforme,npr. UNIX,LINUX, MS WINDOWS. MS SQL SERVER je Microsofov proizvod, namjenjen poslužiteljskim računalima s operacijskim sustavima MS WINDOWS. MySQL je besplatni proizvod tvrtke MySQL AB, popularan na raznim platformama, prvenstveno kao podrška web aplikacijama.“ (Manger, 2011.). SUBP je zadužen za upravljanje svim podacima i njihovom obradom te se sastoji od logičke razine modela koji služi za to  da SUBP-u daje elegantnu kontrolu nad podacima,a sastoji se od stranice ili bloka, ekstenta, segmenata te tabličnog prostora,koncepcijske razine koje se sastoji od entiteta i veza između njih, te fizičke razine koja definira način pohrane podataka na fizičkim diskovima.

## 1.2.Jezici u sustavima za upravljanje bazama podataka

„ Dijelovi sustava za upravljanje bazama podataka su jezik za definiciju podataka i jezik za upravljanje podacima a postoji i jezik za postavljanje upita. Jezik za opis podataka ili DDL služi projektantu baze ili administratoru u svrhu zapisivanja sheme ili pogleda. Tim jezikom definiramo podatke i veze među podacima,i to na logičkoj razini. Naredbe DDL obično podsjećaju na naredbe za definiranje složenih tipova podataka u jezicima poput COBOL-a ili C-a. Jezik za manipuliranje podacima ili DML služi programeru za uspostavljanje veze između aplikacijskog programa i baze. Naredbe DML omogućuju manevre po bazi,te jednostavne operacije kao što su upis,promjena,brisanje ili čitanje zapisa. U nekim je softverskim paketima DML zapravo biblioteka potprograma. U drugim paketima se radi o posebnom jeziku. Programer tada piše program u kojem su izmješane naredbe dvaju jezika,pa takav program treba prevoditi s dva prevoditelja (DML-precompiler, obični compiler).Jezik za postavljanje upita ili QL služi korisniku za pretraživanje baze. To je jezik koji podsjeća na govorni jezik. Naredbe su neproceduralne, tj. samo prikazuju rezultat,ali ne i postupak za dobivanje rezultata. Ovakva podjela je prilično zastarjela. Kod relacijskih baza,postoji tendencija da se sva tri jezika objedine u jedan jezik. Primjer takvog integriranog jezika je SQL. SQL služi za definiranje podataka,manipuliranje i pretraživanje. Integrirani jezik se može rabiti interaktivno ili se on može pojavljivati uklopljen u aplikacijske programe. Svi DBMS-i koji su danas u uporabi,koriste isključivo SQL za sve tri svrhe. Ovi jezici nisu programski jezici. Oni su nam nužni da bismo stvorili bazu i povezali se s njom,no nisu nam dovoljni za razvoj aplikacija koje će nešto raditi s podacima iz baze. Tradicionalni način razvoja aplikacija koje rade s bazom je uporaba klasičnih programskih jezika s ugnježđenim DML-naredbama. U današnje vrijeme,aplikacije se najčešće razvijaju u standardnim objektno orjentiranim programskim jezicima kao što su Java,C++,C#.Za interakcije s bazom rabe se unaprijed pripremljene klase objekata. Ovakva tehnika je dovoljno produktivna zbog uporabe gotovih klasa,a razvijeni program se lako dotjeruje,uklapa u veće sustave ili prenosi s jedne baze na drugu.“ (Manger, 2011.).

### 1.3.Organizacija podataka i arhitektura baze podataka

„Sustav za upravljanje bazama podataka omogućuje razdvajanje fizičke i logičke organizacije podataka. Logička organizacija podataka predstavlja organizaciju sa stajališta korisnika baze podataka ili programera, i ona je koncentrirana na vrste podataka i njihove međusobne logičke veze. Fizička organizacija predstavlja organizaciju fizičke pohrane podataka unutar računala. Oblik i organizacija pohranjenih podataka su često potpuno različiti od njihovog logičkog oblika i organizacije. U okviru toga, zadaća ovog sustava je omogućavanje korisniku i programeru manipuliranje podacima uz poznavanje samo logičkog opisa baze podataka, a ne nužno i poznavanja načina fizičke pohrane podataka.“ (Baranović & Zakošek, 2007). „ Arhitektura baze podataka se sastoji od tri sloja: fizičke, globalne logičke razine i lokalne logičke razine. Fizička razina se odnosi na fizički prikaz i raspored podataka na jedinicama vanjske memorije. Sama fizička razina se može podjeliti na više razina, od sasvim konkretnih staza i cilindara na disku, do pojmova datoteke i zapisa. Globalna logička razina odnosi se na logičku strukturu cijele baze. Opis globalne jezične definicije naziva se shema. Shema je tekst ili dijagram koji definira logičku strukturu baze i u skladu je sa zadanim modelom. Lokalna logička razina odnosi se na logičku predožbu o dijelu baze kojega rabi pojedina aplikacija. To je aspekt sa strane korisnika ili programera. Opis jedne lokalne jezične definicije naziva se pogled. To je tekst ili dijagram kojim se imenuju i definiraju svi lokalni tipovi podataka i veze među tim tipovima,u skladu sa modelom.




**Slika 1.Arhitektura baze podataka, izvor: Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

Fizička neovisnost podataka se postiže time što se pravi razlika između fizičke i globalne logičke razine, dok se logička neovisnost postiže razlikovanjem lokalne logičke razine i globalne logičke razine. Na taj način, troslojna arhitektura omogućuje ispunjavanje dvaju najvažnijih ciljeva koje se nastoje postići uporabom baze podataka.“ (Manger, 2011.).

### 1.4.Ciljevi korištenja baza podataka

Viša razina rada sa podacima očituje se u tome što tehnologija baza podataka nastoji isppuniti određene ciljeve. Prvi cilj je osiguravanje fizičke neovisnosti podataka. Razdvaja se logička definicija baze od njezine stvarne fizičke građe. Ako se fizička građa promijeni, to neće zahtjevati promjene u postojećim aplikacijama. Drugi cilj je osiguravanje logičke nezavisnosti podataka. Razdvaja se globalna logička definicija cijele baze podataka od lokalne logičke definicije za jednu aplikaciju. Ako se globalna logička definicija promijeni, to neće zahtjevati promjene u aplikaciji. Treći cilj korištenja baza podataka je osiguravanje fleskibilnosti podataka. U starijim mrežnim i hijerarhijskim bazama,načini pristupanja podacima bili su unaprijed definirani. Korisnik je mogao pretraživati podatke jedino onim redosljedom koji je bio predviđen u vrijeme oblikovanja i implementiranja baze. Danas se podrazumijeva da korisnik može slobodno prebirati po podacima, te uspostavljati veze među podacima. Ovom zahtjevu jedino zadovoljavaju relacijske baze podataka. Četvrti cilj je osiguranje istovremenog pristupa do podataka. Baza mora omogućiti da veći broj korisnika istovremeno rabe iste podatke. Pritom ti korisnici ne smiju ometati jednog drugoga, te svaki od njih treba imati dojam da sam radi s bazom. Peti cilj je očuvanje integriteta. Nastoji se automatski sačuvati točnost i konzistencija podataka, i to u situaciji kad postoje greške u aplikacijama, te konfliktne istovremene aktivnosti korisnika. Šesti cilj je osiguravanje mogućnosti oporavka nakon kvara. Mora postojati pouzdana zaštita baze u slučaju kvara hardvera ili grešaka u radu sistemskog softvera. Ostali ciljevi su zaštita od neovlaštene uporabe ograničavanjem prava korisnika, zadovoljavajuća brzina pristupa, te mogućnost podešavanja i kontrole. O ovom zadnjem cilju se brine osoba zadužena za praćenje performansi, mjenjanja parametara u fizičkoj građi, rutinsko pohranjivanje rezervnih kopija podataka te reguliranje ovlasti korisnika. Ta osoba se naziva administratorom baze podataka.

# 2. Fizička građa i SQL standard

„Glavni cilj fizičkog oblikovanja baze podataka je stvoriti fizičku shemu baze,tj. opis njezine fizičke građe. Fizička shema je u stvari tekst sastavljen od naredbi u SQL-u ili nekom drugom jeziku kojeg razumije sustav za upravljanje bazama podataka. Izvođenjem tih naredbi sustav za upravljanje bazama podataka automatski stvara fizičku bazu.“ (Manger, 2011.). Već smo definirali DDL i DML, pa je potrebno definirati i SQL standard te prikazati primjer uporabe DDL-a i DML-a.

## 2.1.SQL standard

„SQL je oznaka za Structured Query Languange. Razvijen je u 70-im godinama od strane IBM-a, u okviru relacijskog sustava za upravljanje bazama podataka System R. SQL je nakon toga postao standardnim jezikom za relacijske baze podataka. SQL je upitni jezik temeljen na relacijskoj algebri i predikatnom računu. Važna značajka jezika je neproceduralnost. Ugrađeni optimalizator upita pronalazi najefikasniji način obavljanja upita. SQL je koristi kao programski jezik i interaktivni upitni jezik. Kao programski jezik može se ugrađivati u jezike treće i četvrte generacije. SQL objedinjuje funkcije jezika za definiciju podataka (DDL) i jezika za upravljanje podacima (DML).Zadaća SQL-a je omogućiti definiciju podataka,upravljanje podacima i provođenje kontrole nad podacima u relacijskoj bazi podataka.“ (Baranović & Zakošek, 2007). Primjer SQL naredbe za primjenu DDL-a za bazu podataka automobili: CREATE TABLE automobili ( id INTEGER NOT NULL, proizvodac varchar (50), model varchar (50)).Primjer SQL naredbe iz DML dijela jezika: SELECT * FROM `automobili` WHERE `proizvodac`= 'Mitsubishi'.Iz relacije proizvodac dohvaća sve n-torke kojima je vrijednost atributa proizvodac jednak riječi Mitsubishi. Ovaj upit je izvukao sve modele automobila čiji je proizvođač Mitsubishi.

| ID | proizvodac | model |
|---|---|---|
| 3 | Mitsubishi | Lancer evolution VIII |
| 11 | Mitsubishi | Lancer evo IX |
| 12 | Mitsubishi | Lancer evo x final edition |

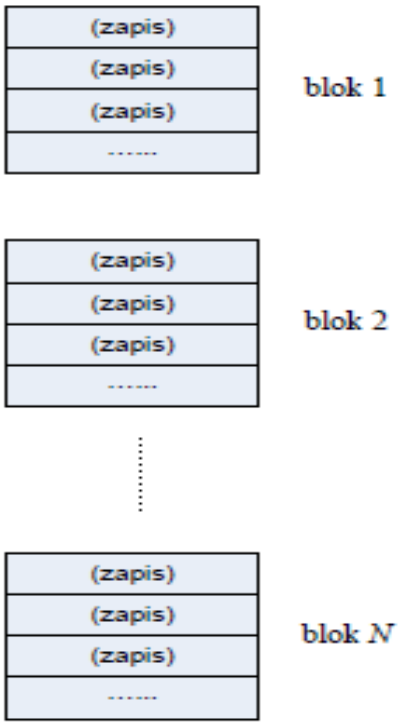
**Tablica 1.Prikaz izvršenog upita DML-om, izvor: izradio autor**

„SQL je standard prema ANSI – American National Standards Institute,ISO ( Organization for International Standardization),UNIX (X/Open),IBM Standard i FIPS (Federal Information Processing Standard).Prva verzija SQL standarda je ANSI X3.135-1986, poznata pod nazivom SQL-86.Standard obuhvaća osnovne operacije s relacijama,definiciju pogleda i nekih pravila integriteta i ugrađivanje SQL naredbi u jezike treće generacije kao što su Fortran, Cobol i C. Taj standard je proširen 1989.godine pod nazivom ANSI X3.135.1989.Sljedeća verzija standarda objavljena je 1992.godine, pod nazivom SQL-92,ali i poznatim pod nazivom SQL-2.Standard je donio velik broj elemenata jezika kao što je pravilo integriteta definirana na nivou relacije ili n-torke, DATETIME tip podataka,upravljanje transakcijama,privremene relacije… Najnoviji standard,SQL-99 ili SQL-3 sadržava nove mogućnosti poput ugradnje aktivnih pravila integriteta i okidača,rekurzivnih operacija,ugradnje novih,korisnički definiranih tipova podataka…U komercijalno raspoloživim sustavima za upravljanje bazama podataka SQL-2 standard je najšire prihvaćen. Ipak,zbog njegove složenosti, do sada još niti jedan sustav ne podržava taj standard u potpunosti. SQL-92 standard definira tri kategorije: Entry SQL, Intermediate SQL i Full SQL. Entry SQL kategorija je slična SQL-89 standardu. Intermediate SQL pokriva mogućnosti standarda koje su najznačajnije u primjeni u danas raspoloživim komercijalnim produktima. Proizvođači komercijalnih sustava ugrađuju i svoje,nestandardne DDL i DML naredbe. Ti su nestandardni dijelovi problematični jer programski kod postaje neprenosiv između različitih SQL sustava,a također se bitno otežava usaglašavanje oko budućih standarda.“ (Baranović & Zakošek, 2007).

## 2.2.Fizička građa baze podataka

Fizička baza podataka gradi se od datoteka i indeksa pohranjenih na disku. Elementi fizičke građe su: blokovi, zapisi i pokazivači. To je način na koji lakše možemo razumjeti kako sustav za upravljanje bazama podataka početnu shemu pretvara u fizičku bazu,sastavljenu od SQL naredbi CREATE TABLE i CREATE INDEKS. „ Baza podataka fizički se pohranjuje u vanjskoj memoriji računala,najčešće na magnetskom disku. Operacijski sustav računala djeli vanjsku memoriju u jednako velike blokove (sektore).Veličina bloka je konstanta operacijskog sustava i ona može iznositi 512 bajta ili 4096 bajta. Svaki blok jednoznačno je zadan svojom adresom. Osnovna operacija s vanjskom memorijom je prijenos bloka sa zadanom adresom iz vanjske memorije u glavnu, ili obratno. Dio glavne memorije koji

sudjeluje u prijenosu zove se buffer. Blok je najmanja količina podataka koja se može prenjeti. Vrijeme potrebno za prijenos bloka koje je mjereno u milisekundama, neusporedivo je veće od vremena potrebnog za bilo koju radnju u glavnoj memoriji mjereno u mili ili nanosekundama. Zato je brzina nekog algoritma za rad sa vanjskom memorijom određena brojem blokova koje algoritam mora prenijeti, a vrijeme potrebno za postupke u glavnoj memoriji je zanemarivo. Osnovna struktura koja se pojavljuje u fizičkoj građi baze podataka naziva se datoteka. Datoteka je konačni niz zapisa istog tipa pohranjeni u vanjskoj memoriji. Tip zapisa zadaje se kao uređena n-torka osnovnih podataka ili komponenti, gdje je svaki osnovni podatak opisan svojim imenom i tipom. Sam zapis sastoji se od konkretnih vrijednosti osnovnih podataka. Smatramo da su zapisi fiksne duljine, jedan zapis ima točno jednu vrijednost svakog od osnovnih podataka i ta vrijednost je prikazana fiksiranim brojem bajtova. Tipične operacije koje se obavljaju nad datotekom su ubacivanje novog zapisa, promjena postojećeg zapisa, izbacivanje zapisa ili pronalaženje zapisa gdje zadani podaci imaju zadane vrijednosti. Jedna datoteka obično služi za fizičko prikazivanje jedne relacije iz relacijske baze. Da bismo prikazali neku fizičku relaciju, svaku njezinu n-torku pretvaramo u zapis, te zapise poredamo u nekom redosljedu, pa ih pohranimo na disk. Na taj način dobivamo niz zapisa pohranjenih na disk, kojoj je naziv datoteka. Prilikom pretvorbe neke relacije u datoteku moramo uvesti brojne fizičke detalje koji nisu postojali u relacijskom modelu, primjerice točnu duljinu pojedinog retka u bajtovima, redosljed podataka unutar zapisa, redosljed zapisa itd. Slično kao i kod relacija, i za datoteke se može uvesti pojam ključa. Kandidat za ključ je osnovni podatak, ili kombinacija osnovnih podataka, čija vrijednost jednoznačno određuje zapis unutar datoteke. Ukoliko ima više kandidata za ključ, tada odabiremo jednog od njih da bude primarni ključ.Za razliku od relacije,datoteka ne mora imati ključ,jer mogu postojati duplicirani zapisi. Ako je datoteka nastala kao fizički prikaz relacije, tada ona ima primarni ključ i on se poklapa s onim kojim smo odabrali za relaciju. Zapise koje čine datoteku se pohranjuju u vanjskoj memoriji.“ (Manger, 2011.).Vanjska memorija se sastoji od blokova, pa se zapisi moraju rasporediti po blokovima. Zapis je obično manji od bloka, pa se više zapisa sprema u jedan blok. Pritom uzimamo da je u jednom bloku smješten cijeli broj zapisa, što znači da niti jedan zapis ne prelazi granicu između dva bloka, te da dio bloka možda ostaje neiskorišten. Ovakav način pohranjivanja omogućuje da jednoznačno odredimo položaj zapisa na disku. Adresa zapisa gradi se kao uređeni par adrese bloka i pomaka u bajtovima unutar bloka.



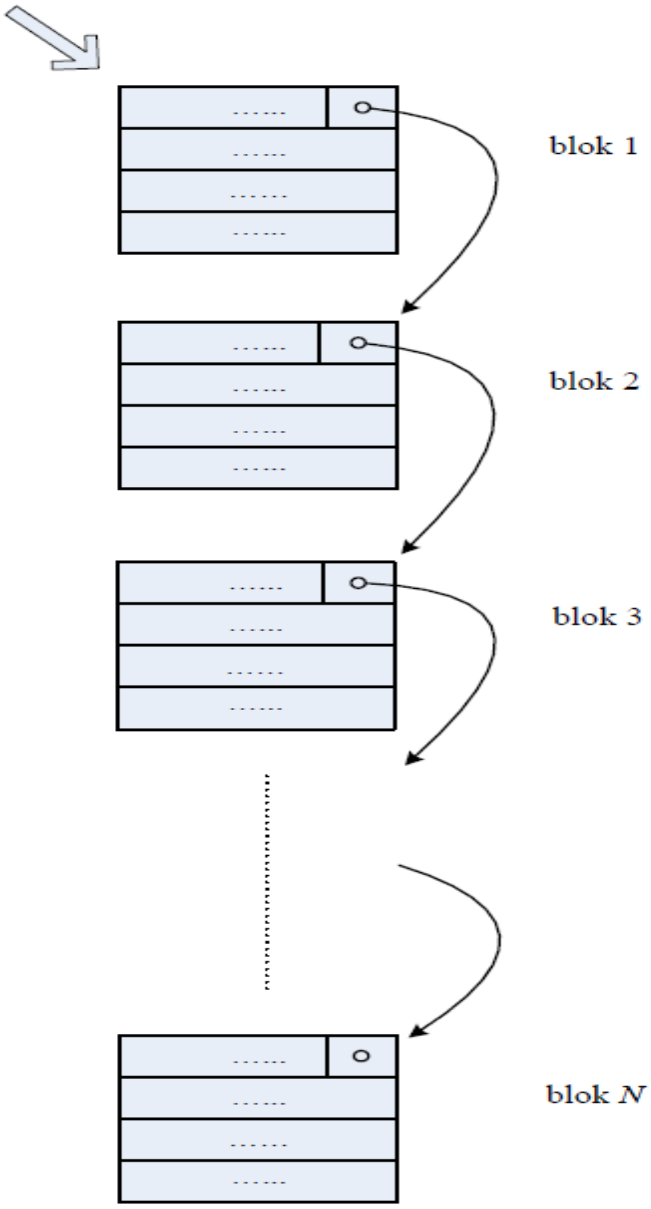

**Slika 2.Prikaz datoteke sastavljene od blokova u kojem su zapisi,izvor : Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

Datoteka se obično sastoji od velikog broja zapisa, pa cijela datoteka obično zauzima više blokova, kao što je prikazano na prethodnoj slici. Položaj i redosljed blokova koji čine istu datoteku određen je posebnim pravilima koja čine takozvanu organizaciju datoteke. Ti blokovi se ne moraju nalaziti na uzastopnim adresama na disku. Još jedan važan element fizičke građe je pokazivač. Riječ je o podatku unutar zapisa ili bloka jedne datoteke koji pokazuje na neki drugi zapis ili blok u istoj ili drugoj datoteci. Pokazivač se realizira tako da njegova vrijednost doslovno bude adresa zapisa ili bloka kojeg treba pokazati. To je fizički pokazivač. Mogući su i logički pokazivači koji pokazuju na implicitan način, npr. navođenje vrijednosti primarnog ključa zapisa kojega treba prikazati. Pokazivači se rabe u raznim organizacijama datoteke. Oni omogućuju uspostavljanje veza između zapisa ili blokova povezivanjem dijelova datoteke u cjelinu, te pristup iz jednog dijela cjeline u drugi.

### 2.3.Organizacija datoteke

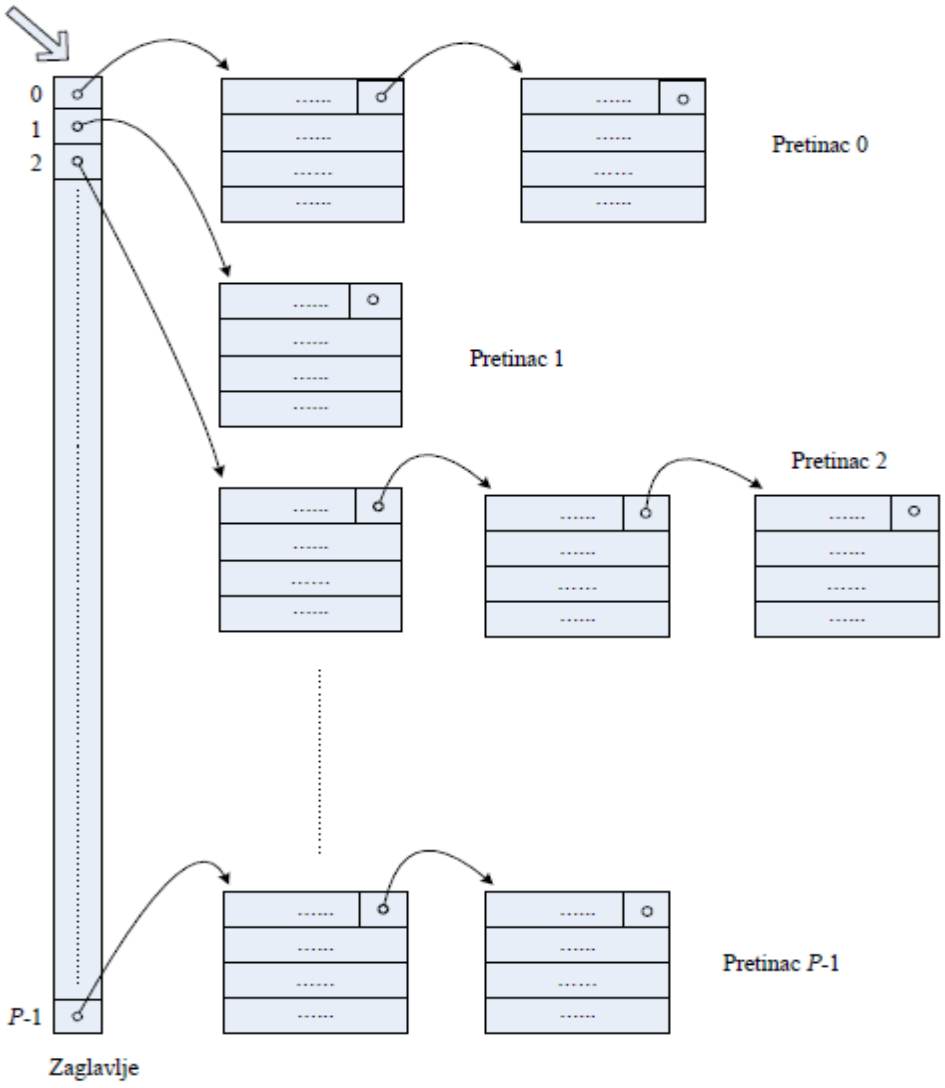
Relacija se iz relacijske baze fizički prikazuje kao datoteka u vanjskoj memoriji računala. Zapisi tih datoteka su raspoređeni u više blokova. Način međusobno povezivanja blokova iz iste datoteke određen je posebnim pravilima koje čine organizaciju datoteke. Postoji nekoliko

važnijih organizacija datoteke: jednostavna datoteka, hash datoteka, indeks-sekvencijalna datoteka,invertirana datoteka, hash datoteka sa podjeljenom hash funkcijom. Svaka od njih ima svoje prednosti i nedostatke u pogledu efikasnog obavljanja osnovnih operacija nad datotekom. Objasniti ćemo jednostavnu datoteku, hash datoteku te indeks sekvencijalnu datoteku. U jednostavnoj datoteci, zapisi su poredani u onoliko blokova koliko je potrebno. Ti blokovi su međusobno povezani u vezanu listu. Svaki blok sadrži fizički pokazivač na idući blok. Kao adresu cijele datoteke pamtimo adresu prvog bloka.




**Slika 2.1. Prikaz jednostavne datoteke, izvor: Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

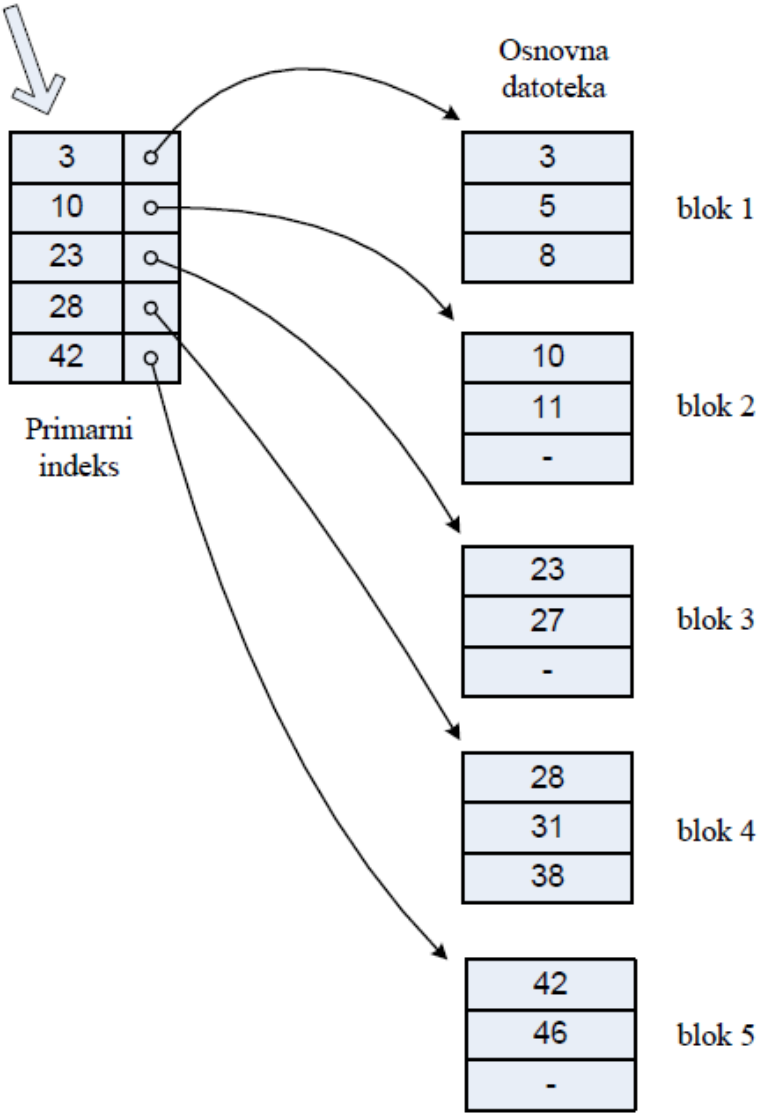
„Prednost jednostavne organizacije je da se kod nje lagano ubacuju, izbacuju i mjenjaju zapisi. Nedostatak ove jednostavne organizacije datoteke je da bilo kakvo traženje, npr. traženje zapisa sa zadanom vrijednošću primarnog ključa, zahtjeva sekvencijalno čitanje blok po blok. Znači jedno će traženje u prosjeku zahtjevati čitanje pola datoteke, pa vrijeme traženja linearno raste s veličinom datoteke.“ (Manger, 2011.). U hash datoteci zapisi su raspoređeni u P cjelina, takozvanih pretinaca označenih rednim brojevima 0,1,2,…P-1.Svaki pretinac građen je kao vezana lista blokova. Zadana je takozvana hash funkcija h()-ona daje redni broj h(k) pretinca u kojeg treba spremiti zapis s vrijednošću ključa k. Ista funkcija kasnije omogućuje i brzo pronalaženje zapisa sa zadanom vrijednošću ključa.




**Slika2.2.Hash datoteka, izvor: Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

Fizički pokazivači na početke pretinca čine zaglavlje koje se smješta u prvi blok ili prvih nekoliko blokova datoteke. Adresu zaglavlja pamtimo kao adresu cijele datoteke. Zaglavlje je obično dovoljno malo, pa se za vrijeme rada sa datotekom može držati u glavnoj memoriji. Skup mogućih vrijednosti ključa obično je znatno veći od broja pretinaca. Zato je važno da h() uniformno (jednoliko) distribuira vrijednosti ključa na pretince. Tada se neće dogoditi da se pretinci neravnomjerno pune, pa će sve vezane liste biti podjednako kratke. Dobra hash funkcija zasniva se na tome da se vrijednost ključa k shvati kao cijeli broj, te da h(k) bude ostatak kod djeljenja k s brojem pretinaca P. Nedostatak hash datoteke je da ona ne može sačuvati sortirani redosljed po ključu za zapise. Hash funkcija ima tendenciju razbacivanja podataka na kvazi slučajan način. Također,ona nije pogodna ako želimo pronaći zapise gdje je vrijednost ključa u nekom intervalu. Hash datoteka je suprotna od jednostavne organizacije. Prednost hash datoteke je u tome što ona osigurava gotovo izravan pristup na osnovu ključa. Indeks sekvencijalna datoteka zasniva se na tome da uz osnovnu datoteku s podacima rabi i takozvani indeks. Indeks je pomoćna datoteka koja olakšava traženje zapisa u osnovnoj datoteci. Indeks koji omogućuje traženje po primarnom ključu naziva se primarni indeks. Zapisi u primarnom indeksu su parovi oblika (k,p) gdje je k vrijednost ključa, a p je fizički pokazivač na zapis u osnovnoj datoteci koji sadrži tu vrijednost ključa. Zbog svojstava primarnog ključa, za zadanu vrijednost k u indeksu može postojati najviše jedan par (k,p). Indeks koji omogućuje traženje po podatku koji nije ključ naziva se sekundarni indeks. Zapisi u sekundarnom indeksu su parovi oblika ( v , p) gdje je v vrijednost podataka, a p je fizički ili logički pokazivač na jedan od zapisa u osnovnoj datoteci koji sadrži tu vrijednost podatka. Taj odabrani podatak nema svojstvo ključa, pa u indeksu može postojati više parova s istim v, dakle mogu postojati parovi (v.p1), (v,p2), (v,p3)… Indeks sekvencijalna datoteka je najpopularnija organizacija zasnovana na indeksu. Riječ je o osnovnoj datoteci koja je organizirana jednostavno i kojoj je zbog dužeg traženja po primarnom ključu dodan indeks. Građa je prikazana slikom 2.3, a vrijednost ključa predstavljaju brojevi. Ako je osnovna datoteka sortirana uzlazno, tada primarni indeks mora biti razrjeđen, to znači da on ne mora sadržavati pokazivače na sve zapise u osnovnoj datoteci, već je dovoljno da sadrži adrese blokova i najmanju vrijednost ključa za svaki blok. Kao adresu cijele datoteke pamtimo adresu indeksa. Glavna prednost ove organizacije da ona omogućuje prilično brz pristup na osnovu ključa. Daljnja prednost je da ona omogućuje čitanje svih zapisa osnovne datoteke sortiranih po ključu. Moguće je lako pronaći zapise gdje je vrijednost ključa u nekom intervalu. Indeks sekvencijalna datoteka predstavlja dobar spoj hash i jednostavne datoteke, i

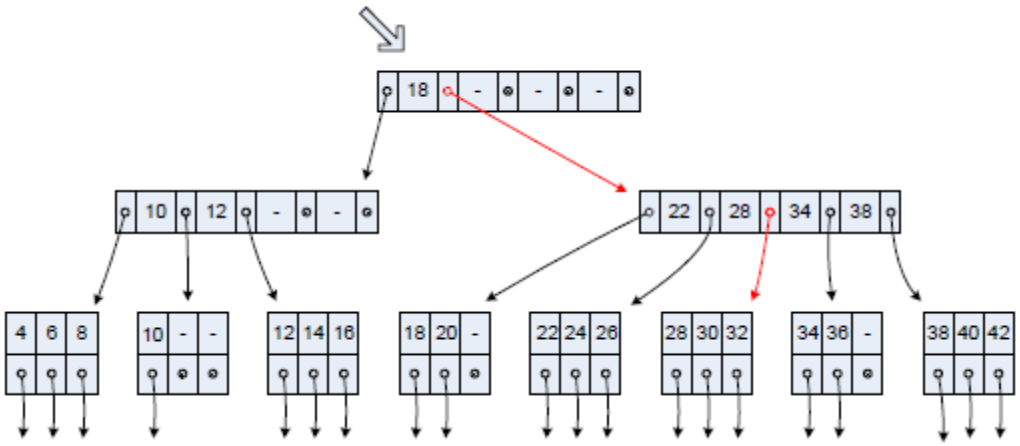
to je razlog zbog čega je toliko popularna. Nedostatak ove organizacije je da se kod nje znatno kompliciraju operacije ubacivanja, mijenjanja i brisanja. Svaka promjena u osnovnoj datoteci zahtjeva da se provede i odgovarajuća promjena u indeksu. Indeks je sam po sebi složena struktura koja troši dodatni prostor na disku.




**Slika 2.3 Prikaz indeks sekvencijalne datoteke, izvor: Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

## 2.4.Organizacija indeksa

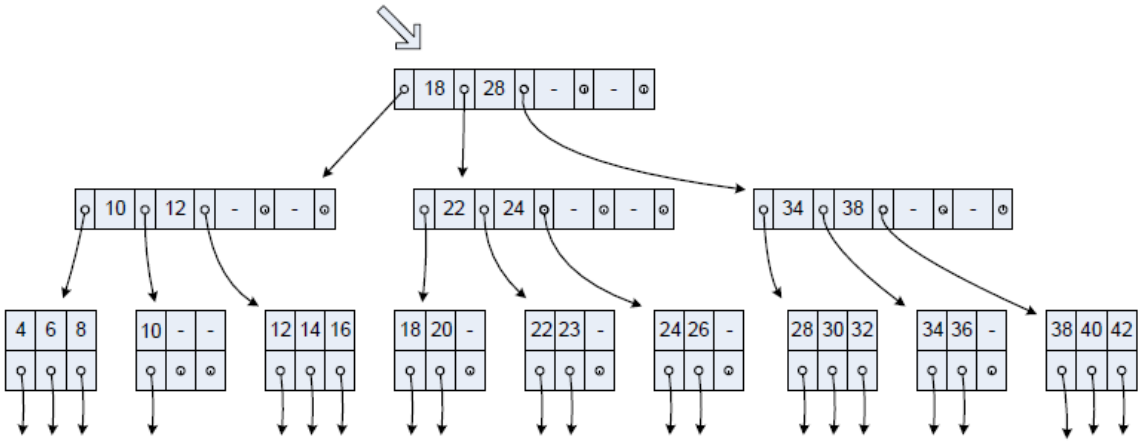
Indeks je sam za sebe također jedna datoteka, pa mora također biti organiziran na odgovarajući način. Uobičajeni način prikazivanja fizičkog indeksa je u obliku b stabla. Riječ je o hijerarhijskoj strukturi podataka koja omogućuje da indeks zaista efikasno obavlja osvoje zadaće, a to su brzo pronalaženje zadane vrijednosti podataka te čuvanja sortiranog redosljeda svih vrijednosti. B stablo reda m je m-narno stablo sa određenim svojstvima. Korijen je ili list ili ima bar dvoje djece. Svaki čvor, izuzev korijena i listova, ima između m/2 i m djece. Svi putovi od korijena do lista imaju istu duljinu. Drugo i treće svojstvo osiguravaju razgranatost stabla u širinu i da su sve grane podjednako visoke. Prikaz gustog primarnog indeksa možemo prikazati pomoću ovog stabla. Prikaz ostalih indeksa je vrlo sličan. Gusti primarni indeks prikazuje se kao B stablo sagrađeno od blokova radne memorije tako da jedan čvor bude jedan blok. Veza između roditelja i djeteta realizira se tako da u blok roditelju piše fizički pokazivač na blok dijete.




**Slika 2.4.Prikaz gustog primarnog indeksa kao B stabla reda 5, izvor: Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

Na slici vidimo gusti primarni indeks neke datoteke prikazan kao B-stablo reda 5. „ Indeks u obliku B-stabla se mmože shvatiti kao hijerarhija jednostavnijih indeksa. Unutrašnji čvor ima sadržaj oblika po,k1,p1,k2,p2,…,kr,pr gdje je pi pokazivač na i-to dijete dotičnog čvora 0<=i<=r, k je vrijednost ključa 1<=i<=r. Vrijednosti ključa unutar čvora su sortirane, dakle k1<=k2<=…kr. Sve vrijednosti ključa u podstablu koje pokazuje p0 su manje od k1. Za 1<=i<r, sve vrijednosti ključa u podstablu kojeg pokazuje pi su u poluotvorenom intervalu

[ki,ki+1). Sve vrijednosti ključa u pod-stablu kojeg pokazuje pr su veće ili jednake kr. List sadrži parove oblika (k,p) gdje je k vrijednost ključa, a p je fizički pokazivač na pripadni zapis u osnovnoj datoteci. Parovi unutar lista su uzlazno sortirani po k. List ne mora biti sasvim popunjen. Jednom zapisu osnovne datoteke odgovara točno jedan par (k,p) u listovima B-stabla. U indeksu koji je prikazan kao B-stablo moguće je vrlo brzo za zadanu vrijednost ključa k pronaći pokazivač p na odgovarajući zapis u datoteci. U tu svrhu slijedimo put od korijena do lista koji bi morao sadržavati par (k,p). To se radi tako da redom čitamo unutrašnje čvorove te usporedimo k sa k1,k2…kr. Ako je k1<=k<ki+1,dalje čitamo čvor kojeg pokazuje pi. Ako je k<k1, dalje čitamo čvor s adresom p0. Ako je k>=kr, koristimo se adresom pr. Kad nas taj postupak konačno dovede u list, tražimo u njemu par sa zadanim k. Efikasnost prikaza indeksa pomoću B-stabla počiva na činjenici da u realnim situacijama B-stablo nikad nema preveliku visinu, to jest sastoji se od maksimalno 3-4 razine. Zbog odnosa veličine bloka, duljine ključa i duljine adrese, red B-stabla m može biti poprilično velik, pa B-stablo postaje  široko i nisko. Mali broj razina znači mali broj čitanja blokova sa diska pri traženju. Za razliku od traženja koje se odvija brzo i efikasno, ubacivanje podataka u B-stablo je kompliciran operacija koja često zahtjeva da se neki od čvorova rascijepi na dva, te da se nakon toga izvrši promjena i u nadređenom čvoru. Lančana reakcija promjena može doći sve do korijena, koji se također može rascijepiti čime se visina stabla povećava za 1. Izbacivanje podataka iz B-stabla odvija se analogno kao ubacivanje, samo u obrnutnom smjeru. Prilikom izbacivanja može doći do sažimanja čvorova, te do smanjenja visine stabla. „ (Manger, 2011.).




**Slika 2.5 Ubacivanje vrijednosti 23 u B-stablo s prethodne slike, izvor : Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

# 3.Stvaranje početne verzije fizičke sheme

Fizička shema baze je tekst sastavljen od naredbi u SQL-u. Izvođenjem tih naredbi sustav za upravljanje bazama podataka stvara fizičku građu baze, pa se fizička shema može shvatiti kao opis fizičke građe. Taj opis je samo implicitan budući da se iz njega ne može doslovno pročitati kako će datoteke biti organizirane. Projektant ima poprilično ograničen utjecaj, a većinu detalja automatski određuje sustav za upravljanje bazama podataka, pomoću ugrađenih pravila. Najvažnija SQL naredba koja se pojavljuje u fizičkoj shemi baze je naredba CREATE TABLE. „ Njome se definira jedna relacija iz baze, dakle ime relacije, te imena i tipovi atributa. Također je moguće zadati koji atribut ili kombinacija atributa čine primarni ključ te relacije, te smije li neki atribut imati neupisane vrijednosti ili ne smije. Početnu verziju fizičke sheme dobivamo tako da svaku relaciju iz razvijene relacijske sheme opišemo jednom naredbom CREATE TABLE.“ (Manger, 2011.). Tipove atributa odredimo najbolje što možemo u skladu s pripadnim rječnikom podataka. Kod određivanja tipova obično moramo napraviti neke kompromise budući da je popis tipova koje podržava sustav za upravljanje bazom podataka ograničen. Slika 3 koja se nalazi ispod, prikazuje početnu fizičku shemu za bazu podataka o fakultetu, a dobivena je na temelju relacijske sheme. Rabi se Mysql inačica SQL-a. Pojavljuje se pet naredbi CREATE TABLE od kojih svaka odgovara jednoj relaciji. Definirani su primarni ključevi. „Izvođenjem naredbi sa slike 3, MYSQL će automatski stvoriti fizičku bazu gdje će svaka od pet relacija biti prikazana kao jedna datoteka. Rabiti će se jedna vrsta indeks- sekvencijalne organizacije koja se naziva MylSAM. Zapisi datoteke biti će raspoređeni u blokove, te će u datoteku automatski biti ugrađen primarni indeks koji osigurava svojstvo primarnog ključa i ubrzava pretraživanje po ključu. Naredba CREATE TABLE mogle su se napisati i bez navođenja primarnog ključa. U tom slučaju, MySQL bi se za prikaz relacije mogao koristiti i jednostavnom datotekom. Ne bi bilo ugrađenog indeksa, ne bi se garantirala jedinstvenost vrijednosti ključa, a traženje po ključu bi zahtjevalo sekvencijalno čitanje cijele datoteke. Neki sustavi za upravljanje bazama podataka poput Oracla, om ogućuju i prikaz relacije u obliku hash tablice. Tada u odgovarajućoj naredbi CREATE TABLE moramo navesti posebnu opciju. Odabirom hash tablice, dodatno se ubrzava traženje po ključu, a usporavaju se sve operacije koje ovise o sortiranom redosljedu po ključu. Jedinstvenost vrijednosti ključa u hash tablici DBMS može garantirati provjerom sadržaja pretinca prilikom upisa podataka.“ (Manger, 2011.).

```
CREATE TABLE STUDENT
  (JMBAG NUMERIC(10) UNSIGNED NOT NULL,
   PREZIME CHAR(20),
   IME CHAR(20),
   GODINA_STUDIJA ENUM('1','2','3','4','5'),
   PRIMARY KEY(JMBAG));

CREATE TABLE PREDMET
  (SIFRA_PREDMETA NUMERIC(5) UNSIGNED NOT NULL,
   NASLOV CHAR(80),
   IME_ZAVODA CHAR(40),
   OIB_NASTAVNIKA NUMERIC(11) UNSIGNED,
   SEMESTAR ENUM('Z','L'),
   ECTS_BODOVI NUMERIC(2) UNSIGNED,
   PRIMARY KEY(SIFRA_PREDMETA));

CREATE TABLE NASTAVNIK
  (OIB NUMERIC(11) UNSIGNED NOT NULL,
   PREZIME CHAR(20),
   IME CHAR(20),
   IME_ZAVODA CHAR(40),
   BROJ_SOBE NUMERIC(3) UNSIGNED,
   PLACA NUMERIC(5) UNSIGNED,
   PRIMARY KEY(OIB));

CREATE TABLE ZAVOD
  (IME_ZAVODA CHAR(40) NOT NULL,
   OIB_PROCELNIKA NUMERIC(11) UNSIGNED,
   OPIS_DJELATNOSTI CHAR(160),
   PRIMARY KEY(IME_ZAVODA));

CREATE TABLE UPISAO
  (JMBAG NUMERIC(10) UNSIGNED NOT NULL,
   SIFRA_PREDMETA NUMERIC(5) UNSIGNED NOT NULL,
   DATUM_UPISA DATE,
   OCJENA ENUM('2','3','4','5'),
   PRIMARY KEY(JMBAG,SIFRA_PREDMETA));
```

**Slika 3. Početna verzija fizičke sheme za bazu podataka o fakultetu, izvor : Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

### 3.1. Kreiranje indeksa

„ Nad jednom relacijom može biti izgrađeno više indeksa, od kojih svaki može sadržavati jedan ili više atributa. Ako je nad relacijom R kreiran indeks za atribut A, može se nesekvencijalno pristupiti do n-torki uz korištenje atributa A kao ključa za dohvat. Ako je nad relacijom R kreiran kompozitni indeks za atribute A,B,C do n-torke se može pristupiti korištenjem bilo koje od navedenih kombinacija atributa kao ključa ključa za dohvat- A, AB, ABC. Prilikom postavljanja upita u jeziku SQL, sustav sam određuje koji će se od indeksa iskoristiti za pristup.“ (Baranović & Zakošek, 2007) Ukoliko odgovarajućeg indeksa nema, pristup n-torkama može biti samo sekvencijalan. Osim radi poboljšanja performansi sustava, indeksi se kreiraju i radi osiguranja jedinstvenosti vrijednosti atributa u relaciji. „Ukoliko je indeks kreiran kao indeks sa jedinstvenim vrijednostima, sustav ne dopušta da se u relaciji pojave dvije n-torke koje bi imale jednake vrijednosti atributa nad kojim je izgrađen takav indeks. Ovakva karakteristika indeksa koristi se za osiguravanje jedinstvenosti ključa u relaciji. Indeks može biti izgrađen tako da su vrijednosti u indeksnim blokovima poredane od manjih prema većim, a isto vrijedi i za obrnuto. Sustav za upravljanje bazom podataka indeks može pretraživati od naprijed ili straga, tako da se indeks može koristiti za sortiranje zapisa u smjeru u kojem su poredane vrijednosti u indeksnim blokovima, i obrnuto. Poredak vrijednosti u indeksnim blokovima indeksa sastavljenih od samo jednog atributa nije bitan. Ako je indeks sastavljen od više atributa, i prema tim atributima se obavlja sortiranje, o poretku vrijednosti u indeksu ovisi mogućnost korištenja tog indeksa prilikom sortiranja. Ukoliko je indeks kreiran za atribute x DESC, y DESC, tada se taj indeks koristi za sortiranje u smjeru x DESC, y DESC, te za sortiranje u smjeru x ASC, y ASC, ali ne i za sortiranje x ASC, y DESC ili obrnuto.“ (Baranović & Zakošek, 2007). Indeks koji omogućava sortiranje za zadnja dva oblika je oblik x DESC, y ASC. „Primjer kreiranja indeksa : CREATE DISTINCT INDEX uniqueZupanija ON Zupanija (nazZupanija). Indeks se može uništiti naredbom DROP INDEX indeks Name. Indeks bi trebalo primjenjivati u slučajevima kada postoje atributi za koje se obavlja spajanje relacija, za atribute koje se često koriste za postavljanje uvjeta selekcije te za atribute za koje se često obavlja grupiranje ili sortiranje.

Prilikom kreiranja indeksa treba voditi računa o nekim njihovim negativnim aspektima, te ih treba koristiti samo tamo gdje je njihova uporaba opravdana. Indeksi zauzimaju značajan prostor. Ažuriranje vrijednosti atributa nad kojima je izgrađen indeks traje znatno dulje nego ažuriranje vrijednosti nad kojima nema indeksa. Indekse ne bi trebalo primjenjivati u sljedećim slučajevima: ukoliko vrijednosti atributa za kojeg se gradi indeks imaju relativno mali broj različitih vrijednosti. ako u relaciji postoji veliki broj upisa ili brisanja n-torki, preporučljivo je da u takvim slučajevima postojeće indekse izbrišemo, te ih ponovno izgradimo nakon obavljenih promjena nad podacima,te ukoliko relacija sadrži vrlo mali broj n-torki (do stotinu).“ (Baranović & Zakošek, 2007). U takvim slučajevima sustav lakše pristupa sekvencijalnoj pretrazi, nego prolaskom kroz strukturu B-stabla, objašnjenoj u potpoglavlju 3.4. „ Zbog navođenja primarnih ključeva i uporabe primarnih indeksa, fizička shema sa slike 4 osigurava da će se u svim datotekama traženje po primarnom ključu odvijati brzo. Traženje po drugim podacima biti će sporo jer će zahtjevati sekvencijalno čitanje odgovarajućih datoteka. Pretraživanje po odabranim podacima koji nisu ključevi možemo ipak ubrzati ako sustavu za upravljanje bazom podataka naredimo da izgradi sekundarne indekse. U tu svrhu se rabe CREATE INDEX. „ (Manger, 2011.).

```
CREATE INDEX SP_IND ON STUDENT (PREZIME);
CREATE INDEX UJ_IND ON UPISAO (JMBAG);
CREATE INDEX US_IND ON UPISAO (SIFRA_PREDMETA);
```

**Slika 3. 1. Sekundarni indeksi za bazu fakulteta, izvor: Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

Slika 3.1. prikazuje dodatak shemi sa slike 3 kojim se uvodi indeks za atribut PREZIME u relaciji STUDENT a indeksi za atribute JMBG odnosno SIFRA_PREDMETA u relaciji UPISAO. Stvaranjem tih sekundarnih indeksa, indeks-sekvencijalna datoteka za prikaz relacije STUDENT odnosno UPISAO nadograđuje se do invertirane organizacije. Time postaje moguće brzo pronalaženje studenta po prezimenu ili brzi ispis svih studenata sortirano po prezimenu. Mogu se efikasno pronalaziti svi predmeti koje je upisao zadani student, te svi studenti koji su upisali zadani predmet. Uvođenjem sekundarnih indeksa poboljšavaju se performanse baze prilikom pretraživanja. No, svaki sekundarni indeks predstavlja dodatni

teret za sustav za upravljanje bazom podataka budući da on zauzima prostor na disku i mora se ažurirati. Projektant baze ne smije pretjerati sa indeksima, već treba procjeniti koje su stvarne potrebe aplikacije. Indeks je potreban samo za one podatke po kojima se vrlo često pretražuje, ili se zahtjeva velika brzina odziva.

### 3.2. Stvaranje i punjenje baze podataka

„ Spomenuto je da fizička baza podataka nastaje izvođenjem naredbi iz fizičke sheme. Izvođenje svih naredbi sa slike 3 i slike 3.1. stvaraju se sve datoteke i indeksi koji čine fizičku bazu podataka o fakultetu. Detalji postupka se razlikuju ovisno o sustavu za upravljanje bazama podataka,a u slučaju MYSQL-a oni izgledaju ovako: CREATE DATABASE fakultet; USE FAKULTET; SOURCE CreateTables.txt;. To su naredbe koji se izvode na interaktivan način unutar komadne ljuske mysql. Pretpostavljeno je da je cjelokupna shema slike 3 i 3.1. pohranjena kao jedna datoteka s imenom CreateTables.txt. Prva naredba stvara praznu bazu fakultet, bazu u kojoj nema niti jedne relacije. Druga naredba otvara bazu a treća naredba pokreće CreateTables.txt kao komandnu datoteku čime če se izvesti sve sql naredbe sa slike 4 i slike 4.1. Nakon ovog postupka, fakultetska baza ima sve potrebne relacije, no te relacije će biti prazne, u njima neće biti n-torki. Da bismo mogli nešto raditi sa bazom, bazu moramo napuniti podacima. Inicijalno punjenje u principu je moguće obaviti standardnim SQL naredbama za ažuriranje. U sql-u postoje naredbe nalik na SELECT koje služe za ubacivanje n-torke u relaciju, odnosno promjenu ili brisanje jedne ili više n-torki. Naredbe su u ovakvom obliku: INSERT INTO STUDENT VALUES (0036398757, 'Markovic','Marko','1'); INSERT INTO NASTAVNIK VALUES (13257600947,'Cantor','Georg',' Zavod za matematiku',102,12000); UPDATE PREDMET SET OIB_NASTAVNIKA = 67741205512 WHERE NASLOV = 'Baze podataka'; DELETE FROM UPISAO WHERE JMBAG = 1191203289. Prve dvije naredbe ubacuju studenta odnosno novog nastavnika, treća mijenja nastavnika za predmet Baze podataka, a četvrta briše sve upise predmeta za studenta sa zapisanim JMBAG-om. Osim standardnih SQL naredbi, u većini sustava za upravljanje bazom podataka stoje nam na raspolaganju i dodatni mehanizmi kojima se inicijalno punjenje baze može obaviti na efikasniji način. MySql nam omogućuje da inicijalni sadržaj jedne relacije pripremimo u obliku tekstualne datoteke, te da jednom onda naredbom taj sadržaj unesemo u bazu.

Pretpostavimo da tekstualna datoteka StudentData.txt sadrži sljedeće podatke za 7 studenata. Svaki redak datoteke odgovara jednoj n-torki iz relacije STUDENT. Podaci su ovakvi: 0036398757 Markovic Marko 1, 1191203289 Petrovic Petar 2, 1192130031 Horvat Dragica 2, 1191205897 Jankovic Marija 1, 0165043021 Kolar Ivan 3, 0036448430 Grubisic Katica 3, 0246022858 Vukovic Janko 1. Zatim koristimo sljedeće naredbe: USE fakultet; LOAD DATA INFILE "StudentData.txt" INTO TABLE STUDENT;. Ove naredbe izvode iz komandne ljuske mysql, tj. njima se sadržaj iz tekstualne datoteke pretvara u n-torke. Nakon inicijalnog punjenja, baza je spremna za rad. U slučaju naše fakultetske baze podataka bilo bi moguće izvesti upite. Također pomoću raznih alata i programskih jezika mogli bismo dalje razvijati aplikacije koje bi služile za dalje ažuriranje podataka, izvještavanje, računjanje statistika i slično.“ (Manger, 2011.).

### 3.3.Postavljanje upita

Postavljanje upita može se interpretirati kao primjena matematičkih operacija kojima se iz postojećih relacija dinamički stvaraju nove virtualne relacije. Jezici za postavljanje upita u relacijskim bazama podataka smatraju se dijelom relacijskog modela.

### 3.4. Jednostavni upiti

„Svi upiti u SQL-u pa tako i oni najjednostavniji, postavljaju se naredbom SELECT. Ta naredba nije isto kao što je i operacija selekcije iz relacijske algebre, ali može obavljati njezinu zadaću. Rezultat izvođenja naredbe SELECT shvaća se kao nova, bezimena i privremena relacija koja ispisuje na zaslonu ili prosljeđuje u aplikacijski program.“ (Manger, 2011.). Prvi primjer upita za našu bazu fakulteta je ovaj : SELECT JMBAG, PREZIME, IME FROM STUDENT WHERE GODINA_STUDIJA = 2;. Odgovor se nalazi na slici.

| JMBAG | PREZIME | IME |
|---|---|---|
| 1191203289 | Petrović | Petar |
| 1192130031 | Horvat | Dragica |

**Slika 3.4.1. Odgovor na prvi primjer upita, izvor : Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

Upit je zahtjevao da se pronađu JMBAGOVE, prezimena i imena svih studenata na drugoj godini studija.

Drugi upit: SELECT DISTINCT OIB_NASTAVNIKA FROM PREDMET;. Odgovor na ovakav upit je na slici.

| OIB_NASTAVNIKA |
|---|
| 33571209458 |
| 44102179316 |
| 25810043761 |
| 67741205512 |

**Slika 3.4.2.Odgovor na drugi upit, izvor : Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

Ovaj upit ispisuje OIB-e nastavnika koji predaju bar jedan predmet. Primjer trećeg upita ispisuje sve podatke o nastavnicima u obliku rang liste s obzirom na njihove plaće. To je ovaj upit: SELECT * FROM NASTAVNIK ORDER BY PLACA DESC;. Odgovor:

| OIB | PREZIME | IME | IME_ZAVODA | BROJ_SOBE | PLACA |
|---|---|---|---|---|---|
| 33571209458 | Codd | Edgar | Zavod za računarstvo | 127 | 16000 |
| 25810043761 | Goedel | Kurt | Zavod za matematiku | 305 | 14000 |
| 67741205512 | Turing | Alan | Zavod za računarstvo | 315 | 13000 |
| 13257600947 | Cantor | Georg | Zavod za matematiku | 102 | 12000 |
| 44102179316 | Klein | Felix | Zavod za matematiku | 252 | 12000 |
| 50076128203 | Pascal | Blaise | Zavod za matematiku | 101 | 11000 |

**Slika 3.4.3. Odgovor na treći upit, izvor: Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

Četvrti upit će ispisivati JMBAG-ove onih studenata koji su iz predmeta sa šifrom 56001 dobili ocjenu veću od dva. Postoji dva načina kako taj upit možemo postaviti. Prvi način: SELECT JMBAG FROM UPISAO WHERE SIFRA_PREDMETA = 56001 AND OCJENA >2;. Drugi način: SELECT JMBAG FROM UPISAO WHERE SIFRA_PREDMETA = 56001 AND OCJENA IN (3,4,5);.

| JMBAG |
|---|
| 0165043021 |
| 0036448430 |

**Slika 3.4.4.Odgovor na četvrti upit , izvor: Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

### 3.5.Složeniji upiti

„Upiti u SQL-u mogu poprimiti mnogo složenije oblike. Dozvoljeno je odjednom raditi više relacija, te kombinirati podatke iz njih. Jedna SELECT nareba može se ugnjezditi unutar druge, tako da rezultat prve naredbe služi kao dio uvjeta drugoj narebi.“ (Manger, 2011.).Navesti ćemo jedan primjer složenog upita. Upit ide ovako : SELECT STUDENT, JMBAG, PREZIME FROM STUDENT, UPISAO WHERE STUDENT .JMBAG = UPISAO.JMBAG AND SIFRA_PREDMETA = 72009;. Možemo i na drugi način: SELECT JMBAG, PREZIME FROM STUDENT WHERE JMBAG IN ( SELECT JMBAG FROM UPISAO WHERE SIFRA_PREDMETA = 72009);. Odgovor upita izgleda ovako :

| JMBAG | PREZIME |
|---|---|
| 1192130031 | Horvat |
| 1191205897 | Janković |
| 0246022858 | Vuković |

#### 3.5.1.Odgovor na složeniji upit , izvor: Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.

„Prvi način je zasnovan na jednostrukoj SELECT naredbi koja simultano rabi dvije relacije. U tom slučaju, stvara se Kartezijev produkt, tj. naredba proizvodi sve kombinacije n-torke iz prve relacije s n-torkama iz druge relacije. Uvijet iza WHERE izbaciti će one kombinacije n-torki koje nas ne zanimaju. Samo ime prvog atributa iza SELECT-a smo morali proširiti sa imenom relacije kao prefiksom- da bi se izbjegla dvoznačnost. Drugo rješenje za upit se rabi ugnježđena SELECT naredba koja pronalazi popis JMBAG-ova studenata koji su upisali traženi predmet. Operator IN provjerava nalazi li se zadana vrijednost podataka u zadanom skupu vrijednosti.“ (Manger, 2011.).

### 3.6. Grupirajući upiti

Odgovori na naše upite su se sastojali od vrijednosti koje se doslovno pojavljuju u pojedinim n-torkama pojedinih relacija baze. Često su nam potrebne i vrijednosti koji odgovaraju cijelim grupama n-torki. Takve grupne vrijednosti nastaju primjenom neke od grupnih funkcija. Postoji funkcija koja daje broj n-torki u grupi, ili zbroj vrijednosti nekog izraza za sve n-torke u grupi, ili minimum vrijednosti izraza,maksimum, prosjek i slično.

U grupirajućim upitima, n-torke se iz baze razvrstavaju u grupe, pa se na njih primjenjuju grupne funkcije. Prvi primjer grupirajućeg upita izgleda ovako: SELECT MIN(PLACA), MAX (PLACA), SUM (PLACA) FROM NASTAVNIK;. Odgovor na upit izgleda ovako:

| MIN(PLACA) | MAX(PLACA) | SUM(PLACA) |
|---|---|---|
| 11000 | 16000 | 78000 |

**Slika 3.6.1.. Odgovor na prvi primjer grupirajućeg upita, izvor: Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

Ovo je najjednostavnija vrsta grupirajućeg upita. Grupa je samo jedna i sastoji se od svih n-torki relacije NASTAVNIK. To znači da će upit proizvesti samo jedan redak ispisa. Primjenom grupnih funkcija pronalazimo minimum, maksimum odnosno zbroj vrijednosti atributa PLACA na skupu svih nastavnika. Drugi primjer grupirajućeg upita biti će složeni upit a on izgleda ovako: SELECT PREZIME, IME, SUM (ECTS_BODOVI) ZBROJ_ECTS FROM UPISAO, STUDENT, PREDMET WHERE UPISAO.JMBAG= STUDENT.JMBAG AND UPISAO .SIFRA_PREDMETA = PREDMET.SIFRA_PREDMETA AND GODINA_STUDIJA >=2 GROUP BY UPISAO.JMBAG HAVING SUM ( ECTS_BODOVI)>= 10 ORDER BY 3 DESC;. Ovaj upit ispisuje imena i prezimena studenata s druge ili treće godine koji su skupili barem 10 ECTS bodova, zajedno s brojem skupljenih ECTS bodova. Ispis će biti sortiran silazno prema broju ECTS bodova. Odgovor izgleda ovako :

| PREZIME | IME | ZBROJ_ECTS |
|---|---|---|
| Horvat | Dragica | 16 |
| Kolar | Ivan | 16 |
| Petrović | Petar | 10 |

**Slika 3.6.2.Odgovor na složeni grupirajući upit, izvor: Manger, R., 2011. Baze podataka. 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.**

Ovo je složeni primjer u kojem se pojavljuje više relacija, izdvajanje n-torki pomoću WHERE, grupiranje, izdvajanje grupa pomoću HAVING, te sortiranje ispisa. WHERE djeluje prije grupiranja, a HAVING i ORDER BY nakon grupiranja.

# 4. Primjer fizičkog dizajna s alatom ORACLE SQL DEVELOPER

Dakle, prvi primjer fizičkog dizajna smo imali u obliku baze podataka za fakultet. Sljedeći primjer će biti baza podataka knjižnjice. Potrebno je kreirati bazu koja će sadržavati naziv knjižnica, popis djelatnika koje rade u knjižnjici uključujući podatke o njihovim plaćama, njezinim članovima te datumima kada su se upisali i ispisali iz knjižnice. Osim knjiga, u bazi podataka će biti prikazana i jedna dodatna tablica za filmove i glazbu, jer osim književne literature, knjižnica nudi i posudbu multimedijske građe. Prvo je potrebno kreirati tablice. Definicija prve tablice izgleda ovako:

CREATE TABLE KNJIŽNICE (OIB_KNJIŽNICE NUMBER(11) NOT NULL , NAZIV_KNJIŽNICE VARCHAR2(50) NOT NULL ,ADRESA_KNJIŽNICE VARCHAR2(120) NOT NULL , CONSTRAINT KNJIŽNICE_PK PRIMARY KEY (OIB_KNJIŽNICE ) ENABLE );.

Koristimo ORACLE SQL DEVELOPER verziju 4.1. Kreirana je tablica knjižnice u kojoj je primarni ključ,tj. primarni indeks oib knjižnice, jer ne može postojati dva oiba, oib je jedinstveni identifikacijski broj koji može predstavljati ustanove, te fizičke i pravne osobe. Potrebno je dodati i indeks na NAZIV_KNJIŽNICE pod nazivom KNJIZNICE_UK1. To je sekundarni indeks koji će označavati jedinstvenost naziva, jer ne može postojati više knjižnica sa istim nazivom. Naš DDL izgleda ovako:

CREATE TABLE KNJIŽNICE (OIB_KNJIŽNICE NUMBER(11) NOT NULL , NAZIV_KNJIŽNICE VARCHAR2(50) NOT NULL , ADRESA_KNJIŽNICE VARCHAR2(120) NOT NULL , CONSTRAINT KNJIŽNICE_PK PRIMARY KEY (OIB_KNJIŽNICE )ENABLE );ALTER TABLE KNJIŽNICE ADD CONSTRAINT KNJIŽNICE_UK1 UNIQUE (NAZIV_KNJIŽNICE )ENABLE);.

Kreirana je prva tablica, sljedeći korak je ispunjavanje tablice podacima. Potrebno je unjeti malo više podataka kako bi pokazali funkcionalnost indeksa u bazi. Mogli smo staviti i indeks na adresu, međutim, zbog jedinstvenog naziva imena knjižnice, indeks na adresi nije potreban. Sljedeći podaci koje je potrebno ubaciti u tablicu knjižnica neće biti realni, nego izmišljeni jer je ovo samo prikaz fizičke baze podataka, na nerealnim podacima. Sada je potrebno ubaciti podatke:

INSERT INTO "DIP"."KNJIŽNICE" ("OIB_KNJIŽNICE", "NAZIV_KNJIŽNICE", "ADRESA_KNJIŽNICE") VALUES ('86425566612', 'Gradska Knjižnica Požega', 'Stjepana Radića 16 Požega'),INSERT INTO "DIP"."KNJIŽNICE" ("OIB_KNJIŽNICE", "NAZIV_KNJIŽNICE", "ADRESA_KNJIŽNICE") VALUES ('78945633333', 'Gradska Knjižnica Pula', 'Zagrebačka 1285 Pula'), INSERT INTO "DIP"."KNJIŽNICE" ("OIB_KNJIŽNICE", "NAZIV_KNJIŽNICE", "ADRESA_KNJIŽNICE") VALUES ('78944444444', 'Čitaonica Dragutin Lerman', 'Osiječka 53 Osijek').

Postupak je potrebno ponoviti i za sve ostale tablice, a nakon završetka svih ostalih tablica, potrebno je testirati funkcionalnost postavljanjem upita i prikazivanjem odgovora na te upite, kao što smo to prikazali u prvom primjeru. Sada će biti potrebno napraviti DDL za sve ostale tablice, a izgleda ovako:

DJELATNICI: CREATE TABLE DJELATNICI (OIB_DJELATNIKA NUMBER(11) NOT NULL , NAZIV_KNJIŽNICE VARCHAR2(50) NOT NULL , IME_DJELATNIKA VARCHAR2(50) NOT NULL , OPIS_POSLA VARCHAR2(50) NOT NULL , DATUM_POČETKA_RADA DATE , DATUM_ZAVRŠETKA_RADA DATE , PLAĆA NUMBER(8, 2) NOT NULL , CONSTRAINT DJELATNICI_PK PRIMARY KEY ( OIB_DJELATNIKA )ENABLE );,

ČLANOVI: CREATE TABLE ČLANOVI (OIB_ČLANA NUMBER(11) NOT NULL , NAZIV_KNJIŽNICE VARCHAR2(50) NOT NULL , IME_ČLANA VARCHAR2(50) NOT NULL , BROJ_POSUĐENIH_KNJIGA NUMBER(1) NOT NULL , POPIS_POSUĐENIH_KNJIGA VARCHAR2(255) NOT NULL , DATUM_POSUĐENE_KNJIGE DATE NULL , ZADANI_DATUM_VRAĆANJA DATE NULL , KONAČNI_DATUM_VRAČENE_KNJIGE DATE , ZAKASNINA NUMBER(10, 2) , "DATUM_UPISA" DATE NOT NULL ENABLE, "DATUM_ISPISA" DATE, CONSTRAINT ČLANOVI_PK PRIMARY KEY (OIB_ČLANA )ENABLE );,

KNJIGE: CREATE TABLE KNJIGE (SERIJSKI_BROJ NUMBER(10) NOT NULL , AUTOR VARCHAR2(100) NOT NULL , NAZIV_KNJIGE VARCHAR2(100) NOT NULL , GODINA_IZDANJA NUMBER(4) NOT NULL , STATUS_KNJIGE VARCHAR2(30) NOT NULL , NAZIV_KNJIŽNJICE VARCHAR2(50) NOT NULL , OIB_ČLANA NUMBER(11) , CONSTRAINT KNJIGE_PK PRIMARY KEY (SERIJSKI_BROJ

)ENABLE );ALTER TABLE KNJIGEADD CONSTRAINT KNJIGE_UK1 UNIQUE ( NAZIV_KNJIGE )ENABLE;.

Naredba alter table je naredba koja omogućava da promijenimo tablicu ako želimo naknadno dodati novi red ili indeks. Slijedi još jedna tablica pod nazivom multimedija koja će sadržavati multimedijsku građu poput filmova i glazbe. DDL:

CREATE TABLE MULTIMEDIJA (SERIJSKI_BROJ_SADRŽAJA NUMBER(10) NOT NULL , IME_SADRŽAJA VARCHAR2(50) NOT NULL , VRSTA_SADRŽAJA VARCHAR2(50) NOT NULL , STATUS_SADRŽAJA VARCHAR2(30) NULL , NAZIV_KNJIŽNICE VARCHAR2(50) NOT NULL , OIB_ČLANA NUMBER(11) NULL , DATUM_POSUDBE DATE NULL , ZADANI_ROK_VRAĆANJA DATE NULL , KONAČNI_ROK_VRAĆANJA DATE NULL , ZAKASNINA NUMBER(10, 2) , CONSTRAINT MULTIMEDIJA_PK PRIMARY KEY (SERIJSKI_BROJ_SADRŽAJA )ENABLE );.

Završena je faza stvaranja tablica, sada je potrebno napuniti ostale tablice podacima jer smo tablicu knjižnica već popunili. DML izgleda ovako:

INSERT INTO "DIP"."DJELATNICI" (OIB_DJELATNIKA, "NAZIV_KNJIŽNICE", IME_DJELATNIKA, OPIS_POSLA, "DATUM_POČETKA_RADA", "PLAĆA") VALUES ('86052648999', 'Gradska Knjižnica Požega', 'Marko Josipović', 'Knjižničar', TO_DATE('2010-03-01 08:15:56', 'YYYY-MM-DD HH24:MI:SS'), '3500,50'),

INSERT INTO "DIP"."DJELATNICI" (OIB_DJELATNIKA, "NAZIV_KNJIŽNICE", IME_DJELATNIKA, OPIS_POSLA, "DATUM_POČETKA_RADA", "PLAĆA") VALUES ('74899965215', 'Gradska Knjižnica Požega', 'Marko Petrović', 'Auditor književne građe', TO_DATE('2010-03-01 08:17:02', 'YYYY-MM-DD HH24:MI:SS'), '3300,00'),

INSERT INTO "DIP"."ČLANOVI "("OIB_ČLANA", "NAZIV_KNJIŽNICE", "IME_ČLANA","BROJ_POSUĐENIH_KNJIGA","POPIS_POSUĐENIH_KNJIGA",ZAKASNINA,DATUM_UPISA)VALUES('86256974555', 'Gradska Knjižnica Požega', 'Josip Bošnjak', '0', 'nema posuđenih', '0', TO_DATE('2012-08-01 08:21:42','YYYY-MM-DD HH24:MI:SS')),

INSERT INTO "DIP"."ČLANOVI" ("OIB_ČLANA", "NAZIV_KNJIŽNICE", "IME_ČLANA","BROJ_POSUĐENIH_KNJIGA","POPIS_POSUĐENIH_KNJIGA","DAT UM_POSUĐENE_KNJIGE","ZADANI_DATUM_VRAĆANJA","KONAČNI_DATUM_V RAČENE_KNJIGE",ZAKASNINA,DATUM_UPISA) VALUES ('14555569332', 'Gradska Knjižnica Požega', 'Marko Antolović', '1', 'Marko Jurendić, Informatika', TO_DATE('2012-01-01 08:23:17', 'YYYY-MM-DD HH24:MI:SS'), TO_DATE('2012-01-31 08:23:33', 'YYYY-MM-DD HH24:MI:SS'), TO_DATE('2015-08-20 08:23:49', 'YYYY-MM-DD HH24:MI:SS'), '0', TO_DATE('2010-02-01 08:24:23', 'YYYY-MM-DD HH24:MI:SS')),

INSERT INTO "DIP"."KNJIGE" (SERIJSKI_BROJ, AUTOR, NAZIV_KNJIGE, GODINA_IZDANJA, STATUS_KNJIGE, "NAZIV_KNJIŽNJICE") VALUES ('1455552369', 'Marko Jurendić', 'Informatika', '2010', 'Dostupno', 'Gradska Knjižnica Požega'),

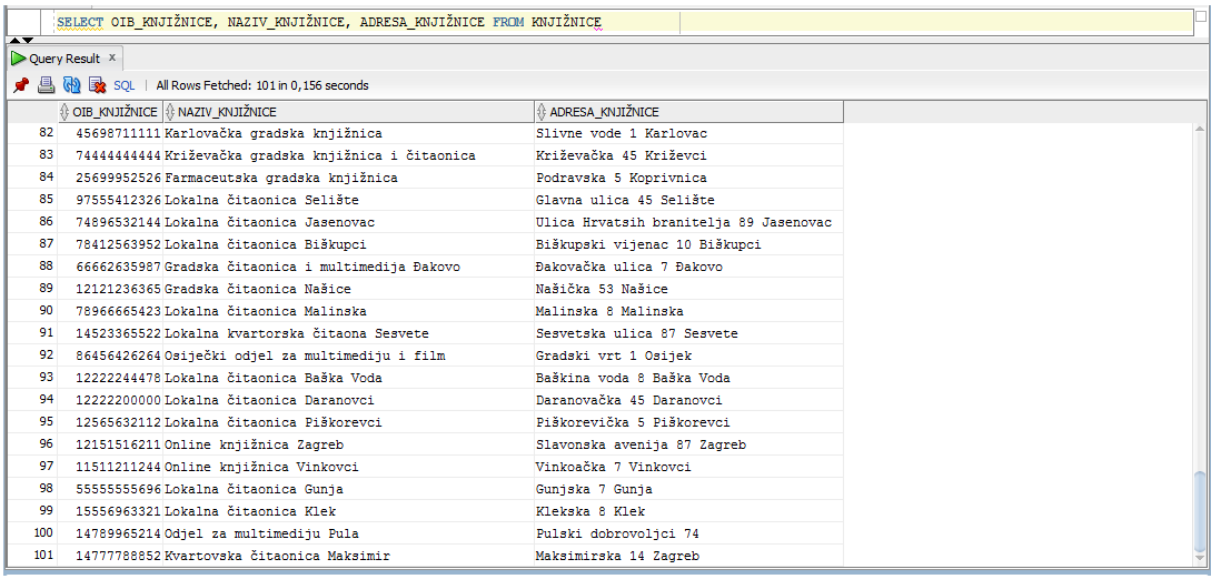
INSERT INTO "DIP"."KNJIGE" (SERIJSKI_BROJ, AUTOR, NAZIV_KNJIGE, GODINA_IZDANJA, STATUS_KNJIGE, "NAZIV_KNJIŽNJICE") VALUES ('1478885699', 'Josip Josipović', 'Glazbena umjetnost', '2005', 'Dostupno', 'Gradska Knjižnica Požega'),

INSERT INTO "DIP"."MULTIMEDIJA" ("SERIJSKI_BROJ_SADRŽAJA", "IME_SADRŽAJA", "VRSTA_SADRŽAJA", "STATUS_SADRŽAJA", "NAZIV_KJNIŽNICE") VALUES ('2436457984', 'Učilica za 8 razred', 'CD sa interaktivnim sadržajem ', 'Dostupno', 'Gradska Knjižnica Požega'),

INSERT INTO "DIP"."MULTIMEDIJA" ("SERIJSKI_BROJ_SADRŽAJA", "IME_SADRŽAJA", "VRSTA_SADRŽAJA", "STATUS_SADRŽAJA", "NAZIV_KJNIŽNICE") VALUES ('7888885641', 'Životinje u divljini', 'DVD sa dokumentarnim sadržajem', 'Dostupno', 'Gradska Knjižnica Požega').

Nakon faze ubacivanja podataka, slijedi provjera funkcionalnosti baze. To ćemo napraviti pomoću upita. Upit će izgledati otprilike ovako:

SELECT OIB_KNJIŽNICE, NAZIV_KNJIŽNICE, ADRESA_KNJIŽNICE FROM KNJIŽNICE.



```
SELECT OIB_KNJIŽNICE, NAZIV_KNJIŽNICE, ADRESA_KNJIŽNICE FROM KNJIŽNICE
```

Query Result | SQL | All Rows Fetched: 101 in 0,156 seconds

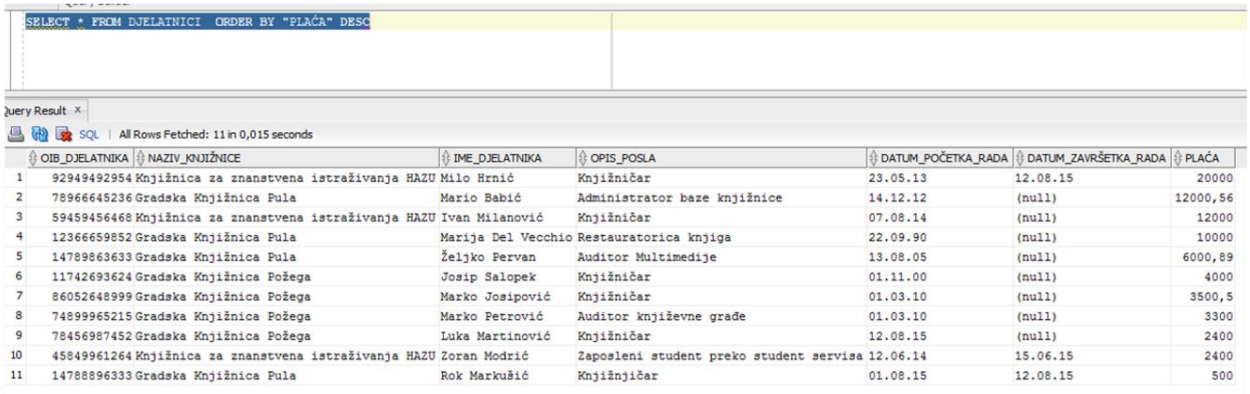
| | OIB_KNJIŽNICE | NAZIV_KNJIŽNICE | ADRESA_KNJIŽNICE |
|---|---|---|---|
| 82 | 45698711111 | Karlovačka gradska knjižnica | Slivne vode 1 Karlovac |
| 83 | 74444444444 | Križevačka gradska knjižnica i čitaonica | Križevačka 45 Križevci |
| 84 | 25699952526 | Farmaceutska gradska knjižnica | Podravska 5 Koprivnica |
| 85 | 97555412326 | Lokalna čitaonica Selište | Glavna ulica 45 Selište |
| 86 | 74896532144 | Lokalna čitaonica Jasenovac | Ulica Hrvatsih branitelja 89 Jasenovac |
| 87 | 78412563952 | Lokalna čitaonica Biškupci | Biškupski vijenac 10 Biškupci |
| 88 | 66662635987 | Gradska čitaonica i multimedija Đakovo | Đakovačka ulica 7 Đakovo |
| 89 | 12121236365 | Gradska čitaonica Našice | Našička 53 Našice |
| 90 | 78966665423 | Lokalna čitaonica Malinska | Malinska 8 Malinska |
| 91 | 14523365522 | Lokalna kvartorska čitaona Sesvete | Sesvetska ulica 87 Sesvete |
| 92 | 86456426264 | Osiječki odjel za multimediju i film | Gradski vrt 1 Osijek |
| 93 | 12222244478 | Lokalna čitaonica Baška Voda | Baškina voda 8 Baška Voda |
| 94 | 12222200000 | Lokalna čitaonica Daranovci | Daranovačka 45 Daranovci |
| 95 | 12565632112 | Lokalna čitaonica Piškorevci | Piškorevička 5 Piškorevci |
| 96 | 12151516211 | Online knjižnica Zagreb | Slavonska avenija 87 Zagreb |
| 97 | 11511211244 | Online knjižnica Vinkovci | Vinkoačka 7 Vinkovci |
| 98 | 55555555696 | Lokalna čitaonica Gunja | Gunjska 7 Gunja |
| 99 | 15556963321 | Lokalna čitaonica Klek | Klekska 8 Klek |
| 100 | 14789965214 | Odjel za multimediju Pula | Pulski dobrovoljci 74 |
| 101 | 14777788852 | Kvartovska čitaonica Maksimir | Maksimirska 14 Zagreb |

**Slika 5.1. Rezultati izvršenog upita na tablici knjižnice, izvor: izradio autor**

Drugi upit koji ćemo postaviti biti će vezan za djelatnike a izgledati će ovako:

SELECT * FROM DJELATNICI ORDER BY "PLAĆA" DESC.

Rezultat je prikazan na slici. Upit izlistava plaću djelatnika pojedinih knjižnica od najveće plaće prema najmanjoj.



```
SELECT * FROM DJELATNICI ORDER BY "PLAĆA" DESC
```

Query Result | SQL | All Rows Fetched: 11 in 0,015 seconds

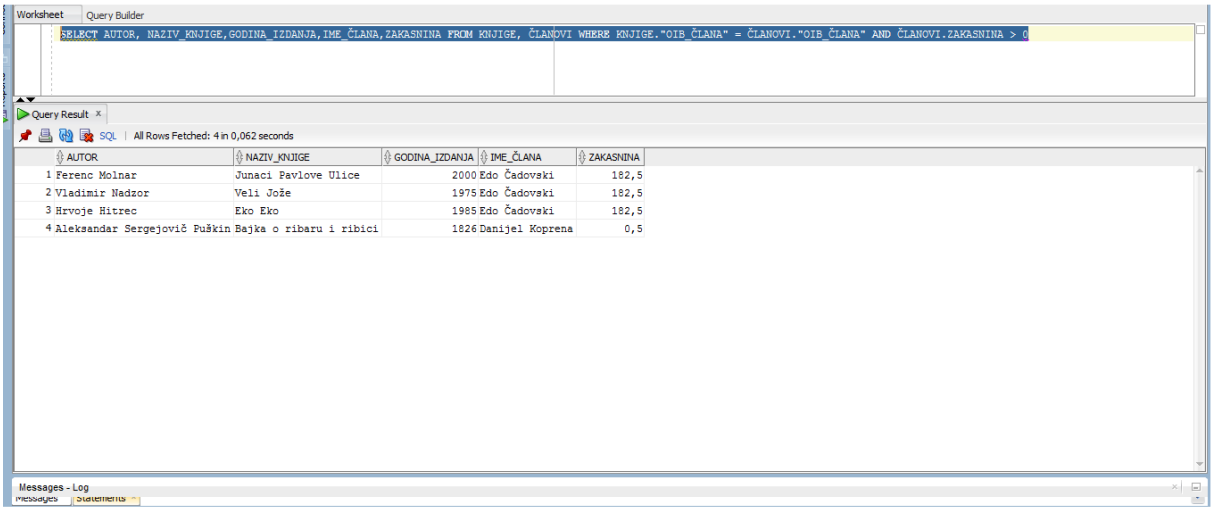
| | OIB_DJELATNIKA | NAZIV_KNJIŽNICE | IME_DJELATNIKA | OPIS_POSLA | DATUM_POČETKA_RADA | DATUM_ZAVRŠETKA_RADA | PLAĆA |
|---|---|---|---|---|---|---|---|
| 1 | 92949492954 | Knjižnica za znanstvena istraživanja HAZU | Milo Hrnić | Knjižničar | 23.05.13 | 12.08.15 | 20000 |
| 2 | 78966645236 | Gradska Knjižnica Pula | Mario Babić | Administrator baze knjižnice | 14.12.12 | (null) | 12000,56 |
| 3 | 59459456468 | Knjižnica za znanstvena istraživanja HAZU | Ivan Milanović | Knjižničar | 07.08.14 | (null) | 12000 |
| 4 | 12366659852 | Gradska Knjižnica Pula | Marija Del Vecchio | Restauratorica knjiga | 22.09.90 | (null) | 10000 |
| 5 | 14789863633 | Gradska Knjižnica Pula | Željko Pervan | Auditor Multimedije | 13.08.05 | (null) | 6000,89 |
| 6 | 11742693624 | Gradska Knjižnica Požega | Josip Salopek | Knjižničar | 01.11.00 | (null) | 4000 |
| 7 | 86052648999 | Gradska Knjižnica Požega | Marko Josipović | Knjižničar | 01.03.10 | (null) | 3500,5 |
| 8 | 74899965215 | Gradska Knjižnica Požega | Marko Petrović | Auditor književne građe | 01.03.10 | (null) | 3300 |
| 9 | 78456987452 | Gradska Knjižnica Požega | Luka Martinović | Knjižničar | 12.08.15 | (null) | 2400 |
| 10 | 45849961264 | Knjižnica za znanstvena istraživanja HAZU | Zoran Modrić | Zaposleni student preko student servisa | 12.06.14 | 15.06.15 | 2400 |
| 11 | 14788896333 | Gradska Knjižnica Pula | Rok Markušić | Knjižnjičar | 01.08.15 | 12.08.15 | 500 |

**Slika 5.2 Rezultati upita izlistavanja djelatnika sortiranog prema plaći,izvor: izradio autor**

Treći upit će predstavljati nešto složeniji upit u kojemu spajamo više tablica. Upit izgleda ovako:

SELECT AUTOR, NAZIV_KNJIGE,GODINA_IZDANJA,IME_ČLANA,ZAKASNINA FROM KNJIGE, ČLANOVI WHERE KNJIGE."OIB_ČLANA" = ČLANOVI."OIB_ČLANA" AND ČLANOVI.ZAKASNINA > 0.

Upit izlistava naziv autora, naziv knijge, godinu izdanja knjige iz tablica knjiga i članova u kojem je oib člana u tablici knjige jednaka oibu člana u tablici članovi i u kojoj je zakasnina veća od nule. Izlistava  popis knjiga te uz popis, ime člana koji  ima zakasninu. U našem slučaju, jedna osoba se nalazi tri puta na popisu i ima dug prema knjižnici od 182 kune. Rezultat je na slici.

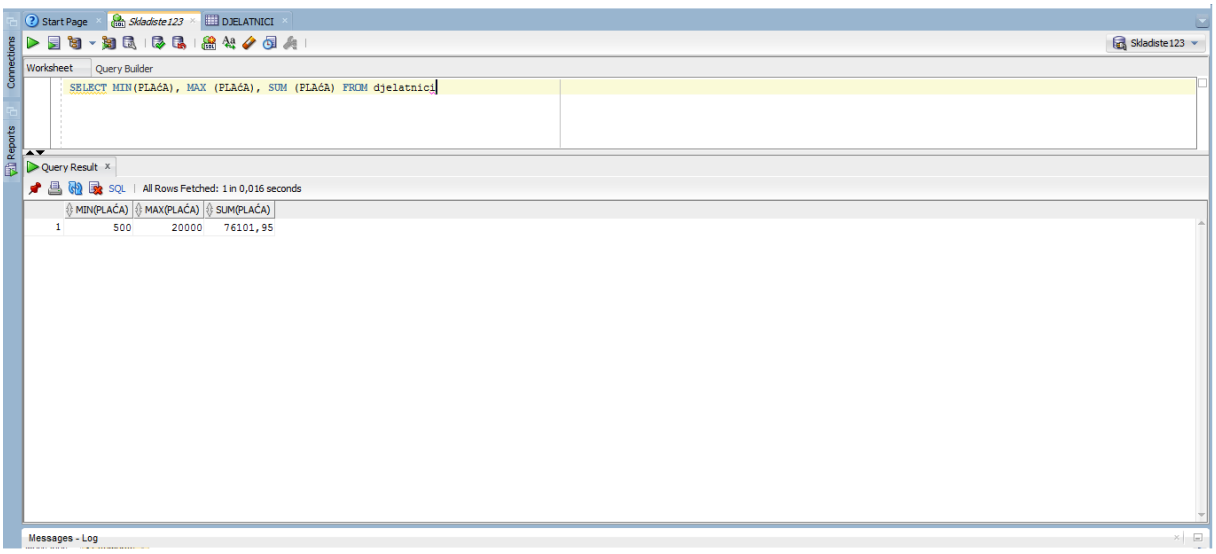



**Slika 5.3. Rezultat složenog upita, izvor: izradio autor**

Zadnji upit kojeg ćemo postaviti je primjer grupirajućeg upita. Upit izgleda ovako:

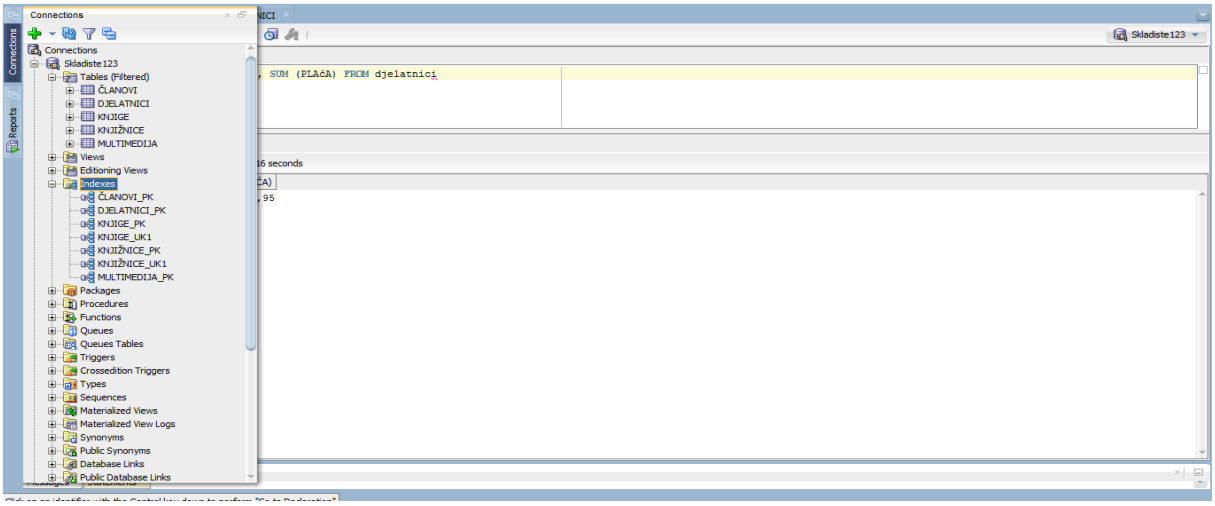
SELECT MIN(PLACA), MAX (PLACA), SUM (PLACA) FROM DJELATNICI.

Upit ispisuje minimalnu plaću koji ima neki djelatnik knjižnice, kao i maksimalnu plaću. A zatim zbraja sve plaće i prikazuje ukupni trošak za sve djelatnike u svim knjižnicama o kojima postoje podaci o djelatnicima i njihovim plaćama. Rezultat je prikazan na slici.




**Slika 5.4. Primjer grupirajućeg upita na djelatnicima knjižnica, izvor:izradio autor**

Još ćemo dodati i prikaz svih indeksa koje postoje na bazi, a nalaze se na slici 5.5.




**Slika 5.5. Prikaz sveukupnih indeksa na  primjeru baze Knjižnica, izvor: izradio autor**

# Zaključak

Baza podataka je skup podataka organiziran na zajednički način bez ponavljanja. Termin baza podataka može označavati i podatke koji su zapisani na papir. Preporučljivo je da se baza podataka stručno naziva sustavom za upravljanje bazama podataka. Glavni cilj uvođenja baze podataka je ubrzanje računalnih aplikacija, smanjenje troškova njihova održavanja, i opskrbljivanje krajnjih korisnika podacima potrebnima da bi se što efikasnije radilo svoj posao. Sustav za upravljanje bazom podataka je programski sustav koji će omogućiti upravljanje bazom podataka. Korisnik će postaviti zahtjev za obavljanje neke operacije, a sustav za upravljanje bazom podataka će ga analizirati, provjeravati, optimizirati, transformirati u niz operacija koje je potrebno obaviti na fizičkoj razini,te će obavljati te operacije i vraćati rezultat. U sustavima za upravljanje bazama podataka koriste se DML, QL i DDL jezik koji će omogućiti stvaranje fizičke sheme, postavljanje upita te unos podataka u sustav. Sustav za upravljanje podataka omogućava razdvajanje fizičke i logičke organizacije podataka. Fizičko oblikovanje predstavlja stvaranje fizičke sheme korištenjem DDL jezika. U to se ubrajaju naredbe CREATE TABLE, CREATE DISTINCT INDEX, a za prikaz tih podataka se koristi DML naredba SELECT. Fizička baza podataka se sastoji od datoteka i indeksa pohranjenih na disku. Postoji nekoliko organizacija datoteka : jednostavna datoteka, hash datoteka, indeks-sekvencijalna datoteka, invertirana datoteka, hash datoteka sa podjeljenom funkcijom. Najpopularnija je indeks sekvencijalna datoteka. Za brže pretraživanje je datoteci dodan indeks. Također, postoji i nekoliko organizacija indeksa. Uobičajeni prikaz indeksa je u obliku b-stabla. SUBP punimo preko naredbe INSERT INTO naziv VALUES. Nakon VALUES u zagradu ćemo postaviti vrijednosti u kojima moramo napisati i apostrofe, osim kod brojčanih vrijednosti kao što je recimo u našem slučaju JMBAG. Kako bismo mogli biti sigurni da su podaci ubačeni u bazu, potrebno je postaviti upite. Postoje jednostavni, složeni i grupirajući upiti. Upite je potrebno postaviti naredbom SELECT. Kod složenijih upita pojavljuju se i dodatne relacije, a preko njih je moguće kombinirati podatke iz tih relacija. Kod grupirajućih upita, potrebno je postaviti upit u kojima će biti moguće sortirati podatke u obliku rang liste, a postoje i funkcije koje će imati mogućnost dati broj n-torki u grupi ili zbrojiti vrijednosti nekog izraza, minimum, prosjek i mnoge druge funkcije. Baza podataka predstavlja osnovu za gotovo svaku aplikaciju.

# 6.Popis tablica

Tablica 1.Prikaz izvršenog upita DML-om……5

# 7.Popis slika

Slika 1.Arhitektura baze podataka. ..... 9
Tablica 1.Prikaz izvršenog upita DML-om ..... 11
Slika 2.Prikaz datoteke sastavljene od blokova u kojem su zapisi. ..... 14
Slika 2.1. Prikaz jednostavne datoteke. ..... 15
Slika2.2.Hash datoteka. ..... 16
Slika 2.3 Prikaz indeks sekvencijalne datoteke. ..... 18
Slika 2.4.Prikaz gustog primarnog indeksa kao B stabla reda 5. ..... 19
Slika 2.5 Ubacivanje vrijednosti 23 u B-stablo s prethodne slike. ..... 20
Slika 3. Početna verzija fizičke sheme za bazu podataka o fakultetu. ..... 22
Slika 3. 1. Sekundarni indeksi za bazu fakulteta. ..... 24
Slika 3.4.1. Odgovor na prvi primjer upita. ..... 26
Slika 3.4.2.Odgovor na drugi upit. ..... 27
Slika 3.4.3. Odgovor na treći upit. ..... 27
Slika 3.4.4.Odgovor na četvrti upit. ..... 27
3.5.1.Odgovor na složeniji upit ..... 28
Slika 3.6.1.. Odgovor na prvi primjer grupirajućeg upita. ..... 29
Slika 3.6.2.Odgovor na složeni grupirajući upit. ..... 29
Slika 5.1. Rezultati izvršenog upita na tablici knjižnice ..... 34
Slika 5.2 Rezultati upita izlistavanja djelatnika sortiranog prema plaći ..... 34
Slika 5.3. Rezultat složenog upita ..... 35
Slika 5.4. Primjer grupirajućeg upita na djelatnicima knjižnica ..... 36
Slika 5.5. Prikaz sveukupnih indeksa na primjeru baze Knjižnica ..... 36

## Sažetak

Cilj ovog rada je bio prikazati fizički dizajn baze podataka, tj. kako napraviti fizičku shemu baze sa njihovim atributima i tipovima podataka. Baza podataka predstavlja skup organiziranih podataka zapisani na računalu bez nepotrebne redundancije. Sustav za upravljanje bazom podataka je precizniji izraz, jer je riječ o softveru koji omogućava izvođenje naredbi koje je korisnik upisao. Glavni cilj sustava za upravljanje podacima je olakšati korisnicima rad i poboljšati efikasnost u izvršenju poslova. Glavne zadaće sustava za upravljanje bazama podataka je također zaštititi bazu podataka od neovlaštenog korištenja, spriječiti narušavanje referencijalnog integriteta baze, osigurati obnovu podataka u slučaju uništenja, omogućiti višekorisnički rad te optimizirati sve funkcije i obavljati ih efikasno. Kreiranje tablica u bazi podataka započinje naredbom CREATE TABLE. Ujedno predstavlja i jezik za definiranje podataka, DDL. Punjenje podataka u tablice započinje se naredbom INSERT INTO ime tablice ('id','ime') VALUES ('1','nekoime'). To je ujedno i naredba za manipuliranje podacima ili DML. Od DML-a, još postoje i UPDATE, DELETE, SELECT. Select naredba je upit koji služi za ispis podataka uz neka ograničenja ako postoje ili za ispis svih podataka. Dakle, u ovom radu, postupak kreiranja baze podataka opisan je na dva primjera,na primjeru knjižnice i baze podataka fakulteta.

## Summary

The aim of this work was to display the physical design of the database, IE. How to make a physical database schema with their attributes and data types. A database is a collection of organized data stored on your computer without unnecessary redundancy. Database management system is a more accurate term, because it is a software that allows you to perform commands that the user typed. The main objective of data management is to facilitate users to work and improve efficiency in the execution of jobs. The main tasks of database management systems is also to protect the database from unauthorized use, to prevent the violation of referential integrity base, ensuring the restoration of data in the event of destruction, enable multi-user operation and optimise all the functions and perform them efficiently. Create tables in a database begins with a CREATE TABLE statement. It also represents a language for defining data, DDL. Filling the data in the table begins with the command INSERT INTO the name of the ('id','name') VALUES ('1','somename'). It is also the command to manipulate data or DML. From DML-and, still exist and UPDATE, DELETE, SELECT. Select the command is a query that is used to print the data with some limitations if there are, or to print all the data. So, in this paper, the process of creating a database is described in the two examples, on the example of libraries and databases of the faculty.

# 8.Literatura

a)materijali za predavanje

1.Baranović, M. & Zakošek, S., 2007. *Baze podataka.* Zagreb: Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva .

b) skripta

2.Manger, R., 2011. *Baze podataka.* 2. ur. Zagreb: Sveučilište u Zagrebu PMF- matematički odsjek.

c) završni rad

3.Privrat, H., 2011. *Objektno orjentirane baze podataka.* Pula: Sveučilište Jurja Dobrile u Puli Odjel za ekonomiju i turizam dr. Mijo Mirković.

d) knjiga

4.Šehanović , J., Hutinski, Ž. & Žugaj, M., 2002.. *Informatika za ekonomiste.* Pula: Sveučilište u Rijeci .