BAZE PODATAKA

O koncepciji koja omogućava funkcioniranje informacijskog društva

Čovjeku je urođena težnja da podatke koje prima iz svoje okoline na neki način uredi, organizira ih bilo u svom umu, bilo na neki drugi, "izvanjski" način, jer ih tako lakše pamti, lakše se njima služi i njima manipulira. Zbog toga je zapravo, možda i većina podataka s kojima se susrećemo u svakodnevnom životu već uređena više ili manje prikladno. Recimo, pojmovi u enciklopediji poredani su po abecednom redu. U reklamnim katalozima trgovačkih lanaca, proizvodi koji se nude grupirani su prema nekim zajedničkim karakteristikama. U knjizi izdanih računa nekog poduzeća, ti su računi poredani po datumu, odnosno vremenu njihovog izdavanja. Niz stavki na računu koji dobijemo na blagajni trgovine poredan je slučajno, ali su podaci svake stavke uređeni - najprije je primjerice napisan naziv robe, zatim njezina količina, pa cijena, pa iznos.

Pojam baze podataka vezan je uz problem organizacije veće količine podataka kojima se služimo u svom životu i radu. Premda se ti podaci mogu čuvati na bilo kojem mediju, recimo na papiru, pa bi se, u najopćenitijem smislu, svaki skup podataka organiziranih na određeni način mogao smatrati njihovom "bazom", ovaj pojam se u svom preciznijem značenju odnosi na podatke koji se skupljaju, čuvaju i obrađuju na nekom od medija elektroničkog računala. Ovakva je definicija baze podataka razumljiva, pogotovo danas kad se zapravo svaka veća količina podataka obično i čuva u elektronskom obliku, odnosno na magnetskom mediju kojemu elektroničko računalo može "pristupiti", ali i s obzirom na činjenicu da je podatke reprezentirane na neki drugi način prilično teško "organizirati", a da je vrijeme pristupa tim podacima gotovo neusporedivo prema vremenu pristupa podacima putem elektroničkog računala. Na poslijetku, možemo konstatirati da je uistinu tek suvremena informatička tehnologija omogućila čovječanstvu da raspolaže vrlo velikim - u teoriji gotovo neizmjernim - količinama podataka i njima se služi na vlastitu korist.

Bazu podataka precizno možemo definirati kao skup podataka koji se nalaze na nekom od medija elektroničkog računala (obično je to hard disk), a koji su organizirani tako da im se može pristupati na što je moguće brži i jednostavniji način. Da bi to bilo moguće, potrebno je imati i računalni softver pomoću kojega se taj skup podataka "uređuje" i "prezentira" - dakle pomoću kojega se novi podaci unose, postojeći ažuriraju ili brišu, unešeni se podaci organiziraju, a oni se potrebni pronalaze i stavljaju na uvid korisnicima baze. Ovakav se softver naziva sustavom upravljanja bazom podataka (Database Management System, DBMS). Napomenimo da se često u običnom govoru i među informatičkim stručnjacima pojmovi sustava upravljanja bazom podataka i baze podataka identificiraju, premda oni, kako vidimo nisu identični.

Temeljnu razliku među različitim sustavima za upravljanje bazom podataka predstavlja "model podataka" na kojemu se ovi sustavi zasnivaju. Pod pojmom modela podataka podrazumijeva se način na koji su podaci pohranjeni na medij elektroničkog računala strukturirani, a on uključuje:

-definiciju strukture podataka
-definiciju pravila integriteta podataka
-definiciju pravila manipulacije s podacima (jezika za manipulaciju podacima)

Razvoj baza podataka u smislu u kojemu je ovdje riječ, započeo je tokom šezdesetih godina prošlog stoljeća, kad je tehnologija omogućila izravan pristup pohranjenim podacima (ranije su se podaci čuvali na bušenim karticama ili magnetskim vrpcama, pa im se pristupalo i obrađivalo ih se sekvencijalno). U to vrijeme nastala su dva modela baza podataka - hijerarhijski i mrežni. Kasnije, početkom sedamdesetih, razvijen je relacijski model koji je danas dominantan. Premda su prva dva modela danas već postali anakronizam, reći ćemo ovdje nešto i o njima.

Podaci u bazi koja za osnovu ima hijerarhijski model strukturirani su u obliku uređenog niza "stabala" (tree). Stablo se sastoji od niza povezanih slogova (record), a svaki slog od niza polja (field), s tim da svakom slogu više hijerarhije u stablu odgovara veći broj slogova niže hijerarhije (odnos parent-child) - od sloga najviše hijerarhije (root) do slogova najniže (leaf). Slogovi se povezuju pomoću pokazivača (pointer)[1]. Primjer hijerarhijski uređene baze podataka je recimo baza podataka o izdanim računima za određenog kupca.

1 Pojam uobičajen u informatičkoj terminologiji kojim se označava varijabla koja sadrži memorijsku adresu - drugim riječima, koja pokazuje na lokaciju u memoriji gdje je smješten neki podatak.

Svakom kupcu odgovara niz izdanih računa, a svakom računu niz stavki, pa tako slog s podacima o kupcu (root) povezujemo s nizom slogova koji se odnose na izdane račune, a svaki slog izdanog računa povezujemo s nizom slogova stavki računa, što čini jedno stablo. Dakako to stablo nije i jedino, ako kupaca ima više.
Mrežni model predstavlja proširenje hijerarhijskog modela pri čemu je dopušteno da jedan child slog ima više parent slogova, pa tako veze slogova čine "mrežu". Primjer ovakve baze predstavlja recimo baza podataka o proizvodima (vrstama proizvoda) i njihovim kupcima - pri čemu isti kupac može kupiti više vrsta proizvoda u određenim količinama, ali jednako tako, istu vrstu proizvoda može kupiti više kupaca. Sada se veze kupaca i proizvoda ostvaruju putem "posrednih" zapisa koji sadrže recimo podatak o količini kupljenih proizvoda iste vrste, a povezani su s jedne strane sa odgovarajućim slogovima kupaca, a s druge s odgovarajućim slogovima vrste proizvoda.
Začetnikom relacijskog modela podataka smatra se britanski matematičar Edgar F. Codd koji je 1970. godine, u vrijeme dok je radio u jednom IBM-ovom istraživačkom laboratoriju u Kaliforniji, napisao članak pod naslovom "A Relational Model of Data for Large Shared Data Banks". U tome članku Codd je izložio strukturu relacijskog modela i njegove osnovne principe. Poticaj za ovaj rad predstavljala je relativna složenost hijerarhijskog, odnosno mrežnog modela i njihova ovisnost o problemu koji se rješava, tj. konkretnim podacima i načinu na koji su međusobno povezani. Nasuprot tome, relacijski je model općenit i zaokružen, s obzirom da ima dobro definiranu teorijsku osnovu (radi se o matematičkoj teoriji relacijske algebre), a k tomu, kako je već u spomenutom radu Codd uočio, "omogućava razvoj univerzalnog jezika za rad s podacima". Osnovni element relacijskog modela je relacija ili tablica koja se sastoji od stupaca (atributa) i redaka (slogova ili zapisa) za koje vrijede pravila da su vrijednosti unutar istog stupca (vrijednosti istog atributa) istog tipa, a da su podaci unutar sloga nezavisni. Definicija relacijskog modela uključuje cijeli niz pojmova, pravila (12 Coddovih pravila za reacijski sustav) i svojstava (strukturna svojstva, svojstva integriteta i manipulativna svojstva). Ukratko, ovdje možemo spomenuti šest vrsta tablica (relacija) - osnovne (realne) tablice, pogledi (view), trenutni pregledi (snapshot), rezultati pretraživanja (query result), međurezultati pretraživanja (intermediate result) i privremene tablice, zatim zahtjeve integriteta i referencijalnog integriteta podataka na osnovi kojih se definiraju primarni (primary) i strani (foreign) ključevi, te pravla manipulacije podacima. Ova se manipulacija vrši posredstvom računalnog jezika na koji je, kao što smo već rekli, ukazao sam Codd, a koji je bio razvijen sredinom sedamdesetih godina prošlog stoljeća, najprije pod nazivima SQUARE i SEQUEL, da bi iz njih nešto kasnije nastao SQL (Structured Query Language), što je naziv koji poznajemo i danas. SQL je početkom devedesetih godina standardiziran, no različiti sustavi upravljanja bazom podataka obično imaju neke specifičnosti u svom "ugrađenom" SQL-u.
Na osnovi Coddovih ideja, IBM je u prvoj polovici sedamdesetih godina prošlog stoljeća započeo s razvojem relacijskog sustava upravljanja bazom podataka (RDBMS) nazvanim System R. Istovremeno su Eugene Wong i Michael Stonebraker sa sveučilišta Berkeley krenuli u razvoj vlastitog relacijskog sustava pod nazivom INGRES. Ova dva sustava postala su operativna do sredine sedamdesetih. Dalji razvoj prvoga doveo je do IBM-ovog relacijskog sustava poznatog danas kao DB2, dok su neki ljudi prvotno uključeni u razvoj INGRES-a započeli s vlastitim projektima, pa su tako nastali relacijski sustavi Sybase i Informix. Grupa kalifornijskih programera predvođena Larryjem Ellisonom započela je opet 1977. vlastiti projekt koji su nazvali Oracle. Pošto je IBM tj. DB2 u prvo vrijeme bio orijentiran prema velikim (mainframe) računalima, tržište manjih sustava (miniračunala, mikroračunala), koje se u to doba počelo nezaustavljivo širiti, pružilo je priliku spomenutoj konkurenciji, tako da su Oracle, Informix i Sybase gotovo preko noći postali velike, brzo rastuće korporacije. Oracle je danas po ukupnom prihodu, ali i profitu treća najveća svjetska informatička kompanija, odmah iza Microsofta i IBM-a, a i inače spada među vodeće "igrače" na svjetskoj korporativnoj sceni (ukupni godišnji prihod preko dvadesettri milijarde dolara, neto prihod oko pet i pol). Njezin osnivač i najveći dioničar, koji je ujedno i njezin CEO (Chief Executive Officer, glavni izvršni (generalni) direktor) Larry Ellison sa svojih preko dvadeset milijardi dolara "ušteđevine" četvrti je najbogatiji čovjek na svijetu. Prvi je, pogađate, osnivač i najveći dioničar Microsofta Bill Gates. Ovdje je zgodno primijetiti da i Billy na bazama dobro zarađuje. Naime Sybase je krajem devedesetih zapao u teškoće, pa je njegov RDBMS kupio Microsoft i iskoristio ga kao osnovu za razvoj svoga vlastitog relacijskog sustava pod nazivom SQL Server. Danas ovaj Microsoftov sustav predstavlja možda i jedinu pravu prijetnju Oracleovoj dominaciji, ako se imaju u vidu samo komercijalni RDBMS-ovi. Informixa je snašla slična sudbina kao i Sybasea, pa se 2001. "utopio" u IBM-u. Stonebraker je sredinom osamdesetih pokrenuo razvoj novog relacijskog sustava zvanog Postgres. Danas je Postgres odnosno PostgreSQL najbolji RDBMS koji spada u domenu slobodnog softvera, a po svojim mogućnostima ne zaostaje za onim komercijalnima (detalji o ovom RDBMS-u mogu se naći na web

stranicama PostgreSQL projekta **www.postgresql.org**). Drugi slobodni RDBMS koji zavrjeđuje pozornost naziva se MySQL.[2] U vezi ova dva posljednja relacijska sustava primijetit ćemo samo da se, dakako, ni Stonebraker, niti vodeći razvijatelji MySQL-a, Finac Michael Widenius i Šveđanan David Axmark, baš nisu previše obogatili.

Nakon ovih historijsko-financijskih razmatranja, vratimo se onim tehničkim. Naime, netko se može zapitati - Kako ta teorija, odnosno praksa RDBMS-ova funkcionira kad podataka u bazi ima jako puno? Zamislimo recimo bazu podataka u koju se upisuju podaci o pozivima putem mobilne mreže nekog mobilnog operatera. Takve su baze vrlo realne, očito je da ne postoji drugi način da operater vodi evidenciju o pozivima, kako bi svojim korisnicima mogao ispostaviti račun za svoje usluge.

Međutim, s interneta saznajemo da naši vodeći mobilni operateri imaju oko dva milijuna korisnika. Dakle ako svaki od njih za mjesec dana učini i svega 10 poziva (a to je zasigurno neki minimum) u bazu se tokom mjesec dana slije oko dvadeset milijuna zapisa o pozivima. Stane li tolika količina podataka uopće u bazu i može li se s tolikom količinom podataka izaći na kraj?

Da bismo odgovorili na ovo pitanje, razmotrimo kao prvo način na koji su strukturirani podaci u ovoj našoj bazi. Jednostavan relacijski model podataka u konkretnom slučaju sastojao bi se od dvije relacije (tablice) - prve koja sadrži podatke o pretplatnicima i druge s podacima o pozivima. Atributi (polja) u prvoj tablici bili bi redni broj (šifra) pretplatnika, njegovo ime, adresa i eventulano neki drugi osobni podaci. Atributi (polja) u drugoj bi bili redni broj (šifra) poziva, vremena uspostave i prekida poziva, šifra pretplatnika kojemu odgovara dotični poziv, te još barem pozvani telefonski broj. Ove bi dvije tablice, dakle, bile povezana preko tzv. ključeva, koje u ovom konkretnom slučaju predstavljaju polja šifre pretplatnika u jednoj i drugoj tablici. Svakoj šifri pretplatnika u tablici poziva (tzv. strani ključ) mora odgovarati šifra pretplatnika u tablici pretplatnika (tzv. primarni ključ), što se osigurava zahtjevom za referencijalnim integritetom podataka tj. baze - ovdje konkretno radi se o zahtjevu da je nemoguće unijeti poziv sa šifrom pretplatnika koja ne odgovara niti jednom od onih koje postoje u tablici pretplatnika (o ovome zahtjevu brigu vodi sam RDBMS). Svakom pretplatniku, dakako, može odgovarati više poziva u tablici pozivi, ali svakom pozivu odgovara samo jedan pretplatnik u tablici pretplatnici (one-to many relationship).

Sada možemo razmotriti problem prostora koji će ovi podaci zauzeti. Svi se relevantni podaci o pozivu zasigurno mogu strpati u kojih dvadesetak bajtova. Relevantni podaci o pretplatniku zauzet će znatno više, možda i preko stotinu bajtova, no njih u bazi ima puno manje (deset puta manje) pa će njihov udio u bazi biti znatno manji. Na osnovi gore navedenih iznosa procjenjujemo de će ukupna mjesečna količina podataka o pozivima u našoj bazi iznositi nekoliko stotinu megabajta (do jednog gigabajta), što za današnje diskovne kapacitete, koji se već mjere u terabajtima, uopće nije neka značajna cifra. Napomenimo ovdje da se veće i značajnije baze podataka obično drže na posebnim uređajima za pohranu podataka koji sadrže veći broj diskova, na koje se podaci spremaju redundantno - ako jedan disk otkaže, podaci se ne gube jer su spremljeni i na nekom drugom. Inače, u sam je sustav za upravljanje bazom podataka redovno ugrađena mogućnost arhiviranja podataka (backup), kao i njihove rekonstrukcije (recovery) u slučaju havarije (hardverske ili softverske).

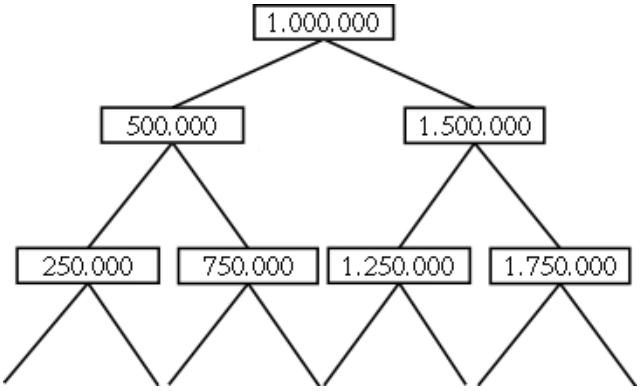
Što se tiče vremena potrebnog za "obradu" podataka iz našeg primjera, priča je nešto kompliciranija. Razmotrimo recimo problem kreiranja mjesečnog računa za određenog korisnika mobilne mreže. Računalo, odnosno RDBMS treba "pregledati" bazu, u njoj pronaći podatke o pozivima koje je taj korisnik ostvario, na osnovi vremena trajanja poziva, a prema unaprijed određenoj tarifi izračunati cijenu poziva i onda sve te iznose pozbrajati. Pretpostavimo da se podaci o pozivima nalaze na hard disku računala. Vrijeme pristupa podacima (blokovima) na disku određeno je brzinom vrtnje diska[3] i pomicanja glava za čitanja podataka u poprečnom smjeru - nešto se vremena potroši i na usporedbe (odgovara li redni broj zadanog korisnika rednom broju korisnika u pročitanom zapisu), nešto i za računanja (iznosa od interesa), no ova su dva posljednja vremena znatno manja od prvoga. Uobičajena vremena pristupa "blokovima" odnosno "trackovima"[4] na hard disku mjere se u milisekundama. Pošto blokovi, odnosno trackovi na disku mogu

2 Pojam slobodnog softvera precizno je definiran u dokumentu pod nazivom GNU General Public License (GNU GPL) Zaklade za slobodni softver. Više o toj temi vidi u članku koji slijedi.

3 Uobičajena brzina vrtnje diska, odnosno njegovih "ploča", iznosi 7200 okretaja u minuti, pa je za jedan okret potrebno 8,33 milisekundi.

4 Radi se o "fizičkim" strukturama na hard disku računala, skupinama "magnetskih regija" u koje se upisuju pojedini bitovi - elementarne jedinice informacije.

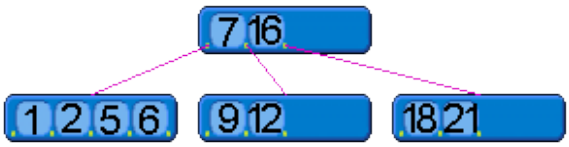
sadržavati desetine, odnosno stotine zapisa iz naše tablice, možemo procijeniti da je za jedan korak u njezinom pretraživanju (uređaj pristupa zapisu, provjeri odgovara li on pozivu zadanog korisnika i u slučaju pozitivnog odgovora izračuna cijenu poziva, te je doda do tada izračunatom iznosu) potrebne, u prosjeku, desetinke ili stotinke milisekunde (ms). Ako uzmemo da je ovo vrijeme recimo pet stotinki milisekunde, ispalo bi da je za računanje iznosa koji je određeni korisnik dužan platiti mobilnom operateru potrebno
0.05 ms * 20.000.000 pretraga = 1000 s = 16,67 min
što je dakako neprihvatljivo dugo vrijeme.
No ovdje ipak postoji čarobni štapić kojim se ovo vrijeme može smanjiti za desetke ili stotine tisuća puta. Njega predstavljaju indeksi, odnosno pojam indeksiranja podataka. U gornjoj priči pretpostavili smo da su zapisi u tablici poziva pohranjeni jedan za drugim, po nekom slučajnom rasporedu (kojim se u bazu i unose). To uistinu i jest tako, no ove podatke možemo na jedan posredan način urediti, tako da za tablicu s podacima o pozivima korisnika mobilne mreže kreiramo indeks na polju (atributu) koje odgovara rednom broju odnosno šifri korisnika.
Da bismo na što jednostavniji način razjasnili o čemu se ovdje radi, razmotrit ćemo strukturu (podataka) koja se naziva binarno stablo. Zamislimo da zapisi u tom stablu sadrže podatak o šifri korisnika i pokazivač na zapis koji odgovara mobilnom pozivu toga korisnika u (osnovnoj) tablici s podacima o mobilnim pozivima (ako tih podataka ima više, onda se može raditi o nizu pokazivača). Ovi su podaci složeni tako da se zapis na "vrhu stabla" (root) odnosi na korisnika sa "središnjim" rednim brojem - u ovom slučaju to je broj 1.000.000 (milijun, jer korisnika kako smo rekli ima dva milijuna). Taj je zapis pokazivačima povezan sa dva child zapisa - onim koji odgovara korisniku s rednim brojem na sredini između 1 i 1.000.000, a to je broj 500.000, te onim koji odgovara korisniku s rednim brojem na sredini između 1.000.000 i 2.000.000, a to je broj 1.500.000. Svaki od ovih zapisa opet ima dva child zapisa koji opet odgovaraju rednim brojevima korisnika na sredini područja kojega omeđuju brojevi koji se pojavljuju na višim razinama hijerarhije, te početni i konačni redni broj korisnika (1 i 2.000.000) - ovdje su konkretno ti brojevi 250.000 i 750.000, te 1.250.000 i 1.750.000. Po analogiji formiraju se sve niže hijerarhije, dok se ne iscrpe svi redni brojevi korisnika (vidi sliku 1). Koliko ovo stablo ima "razina", ako je broj korisnika 2.000.000?



Sl.1 - Shema binarnog stabla iz opisanog primjera

Primijetimo da je ovo pitanje inverzno pitanju iz onog čuvenog problema sa brojem zrnaca pšenice na šahovskoj ploči, kod kojega je zadan broj šahovskih polja, tj. broj razina našeg binarnog stabla, a traži se broj zrnaca pšenice, tj. broj korisnika mobilne mreže. Već i elementarno poznavanje matematike omogućava nam da riješimo problem sa šahovskom pločom i zrncima pšenice
broj zrnaca pšenice = $2^{\text{broj šahovskih polja}} - 1$
Odavde opet lako nalazimo broj razina našeg binarnog stabla
broj razina binarnog stabla = $\log_2$ (broj korisnika mobilne mreže + 1)
Iz posljednjeg izraza lako je izračinati da je za pronalaženje podatka u našem binarnom stablu potrebno maksimalno 21 "korak" tj. pristup i uspoređivanje ($\log_2$ 2.000.000 = 20.93). Nakon pronalaženja traženog podatka (zapisa koji odgovara zadanom rednom broju korisnika), možemo neposredno pristupiti zapisima o pozivima zadanog korisnika u tablici mobilnih poziva (jer zapis u binarnom stablu sadrži pokazivače na te zapise, odnosno informacije o lokacijama tih zapisa u tablici mobilnih poziva). Dakle, za pristup ovim

podacima (pa i za izračun iznosa mjesečnog računa), trebat će nam približno 0.05 ms * 21 + vrijeme potrebno za pristup zapisima u osnovnoj tablici (onima koji se odnose na zadanog korisnika, a njih, kako smo rekli, ima desetak), što sve skupa iznosi nekoliko desetaka mikrosekundi (drugi pribrojnik je dakako znatno veći od prvoga), što je prihvatljivo.
Napomenimo da se indeksi redovno "postavljaju" na polja šifre tj. identifikatora zapisa u tablici, pogotovo ako se radi o zapisima na koje se (preko pokazivača) vežu zapisi iz nekih drugih tablica (takva polja, kao što smo već rekli, nazivamo primarnim ključevima), jer se tako u raznim manipulacijama s podacima u bazi ovi zapisi pronalaze dovoljno brzo, tj. u vremenskom intervalu koji korisnik ne može registrirati kao "zastoj u radu" - recimo u našem konkretnom slučaju, za pretraživanje tablice mobilnih poziva potrebno je samo 25 koraka ($\log_2 20.000.000 = 24.25$), što odgovara vremenu pretraživanja od približno 0.05 ms * 25 = 1.25 ms.
Ovdje treba primijetiti da se pri linearnom povećanju broja podataka u bazi, a to je uobičajeno za mnoge realne baze (poput ove iz našeg primjera), vrijeme potrebno za njihovo pretraživanje povećava logaritamski, što je zapravo vrlo sporo povećanje - ovdje se radi o analogonu odnosa između geometrijske i aritmetičke progresije.
Ovo razmatranje o indeksima, kao i brojke koje su gore navedene, treba smatrati okvirnim. U stvarnosti se zapravo umjesto indeksa u formi binarnih stabala (binary tree index) koriste tzv. B-tree indeksi (vidi sliku 2) kod kojih je broj mogućih childova za dani parent veći od 2 i ima neku fiksnu vrijednost, a u nekim posebnim slučajevima i za neke posebne "tipove" podataka koriste se i indeksi drugih vrsta (bitmap indeksi, hash indeksi, R-tree indeksi i dr.). B-tree indeksi omogućavaju lakše "balansiranje" svoje strukture - primijetimo problem dodavanja ili brisanja podataka (zapisa) unutar binarne strukture - a i brzina pretraživanja u odnosu na one binarne povećava se za faktor $\log_2(N) / \log_n(N)$, gdje je N ukupni broj zapisa u tablici, a n fiksni broj mogućih child zapisa u B-tree strukturi. Ove pogodnosti "plaćaju" se prostorom koji zauzimaju podaci za B-tree indeks, koji je znatno veći, no kao što smo pokazali na početku ovoga razmatranja, potreban prostor na mediju ne predstavlja neki problem. Napomenimo također da se kod modernih RDBMS-ova znatan dio "obrade" zapravo ne odvija na hard disku, već u privremenoj memoriji računala tj. RAM-u (kod Oracleovog RDBMS-a radi se o memorijskom "području" zvanom Database Buffer Cache, gdje se podaci s hard diska prebacuju prije obrade), a tamo sve ide za nekoliko redova veličine brže, što također znatno povećava brzinu rada sustava.



Sl.2 - Shema B-tree indeksa sa 5 mogućih child zapisa
(preuzeto iz Wikipedije)

Recimo na kraju da su baze podataka, odnosno sustavi njihovog upravljanja danas postale neizbježan čimbenik gotovo svih poslovnih aktivnosti, a sve veći značaj dobivaju i u našem privatnom životu. Posebni smjer razvoja ovih sustava predstavljaju "geografski informacijski sustavi" (GIS) koji služe za čuvanje i obradu "prostornih podataka" (podaci o zemljišnim parcelama, o vodovima raznih komunalnih i električnih instalacija, gradskim ulicama i uopće cestovnoj infrastrukturi, o pokrivenosti "signalom" mobilnih mreža) koji se koriste u radu državne uprave (katastar), komunalnih i elektrodistribucijskih poduzeća, mobilnih operatera, za potrebe sustava "cestovne navigacije" i sl. Neke posebne vrste baza podataka koriste se i u znanstvenim istraživanjima (primjerice u velikom projektu kartiranja ljudskog genoma korištena su neka Oracleova rješenja). I premda se čini da je u ovih gotovo pola stoljeća rada i razvoja na ovom području učinjeno sve što se trebalo i moglo učiniti, autor ovoga teksta je uvjeren da će nam dalji razvoj ove grane informatičke znanosti u budućnosti donijeti još mnogo značajnih dostignuća, pa i iznenađenja.

Literatura:
R. Vujnović - SQL i relacijski model podataka
PostgreSQL Documentation (ver. 7.x i 8.x)
Oracle Documentation (ver. 8.x i 9.x)
Wikipedija

Zagreb, studeni 2009.

Dorađeno i dopunjeno u studenom 2010.