UNIVERZITET U NIŠU
ELEKTRONSKI
FAKULTET

SEMINARSKI RAD

Sistemi za upravljanje bazama podataka

Tema:

Fizičko projektvanje baze i optimizacija podataka

Profesor:

Doc. dr Aleksandar Stanimirović

Student:

Tijana Spasić 1064

Niš, Mart 2020

Projektovanje baze podataka

Projektovanje baze podataka je proces organizovanja podataka prema modelu baze podataka. Modeliranje podataka je proces kreiranja modela podataka za podatke koji se smeštaju u bazu podataka i prvi je korak u procesu dizajniranja same baze. Projektant određuje koji se podaci moraju čuvati i kako se elementi podataka međusobno povezuju. Primarni ciljevi upotrebe modela podataka su:

- Osigurava da su svi podaci koji su zahtevani od baze podataka tačno predstavljeni
- Propuštanje podataka će dovesti do stvaranja neispravnih izveštaja i generisaće pogrešne rezultate
- Struktura modela podataka pomaže u definisanju relacijskih tabela, primarnih i stranih ključeva i smeštenih procedura
- Omogućava jasnu sliku o osnovnim podacima i programeri ih mogu koristiti za kreiranje fizičke baze podataka
- Korisno za identifikovanje nedostajućih i suvišnih podataka
- Iako je početno stvaranje modela podataka dugotrajno, čini nadogradnju i održavanje IT infrastrukture jeftinijom i bržom

Projektovanje podataka može biti konceptualno, logičko i fizičko.

Konceptualno projektovanje

Konceptualni model podataka identifikuje odnose na najvišem nivou između različitih entiteta. Karakteristike konceptualnog modela su:

- Uključuje važne entitete i relacije među njima
- Atribut nije naveden
- Primarni ključ nije naveden

Osim informacije o entitetima i relacijama između njih kroz konceptualni model se ne prikazuje nijedna druga informacija.

Logičko projektovanje

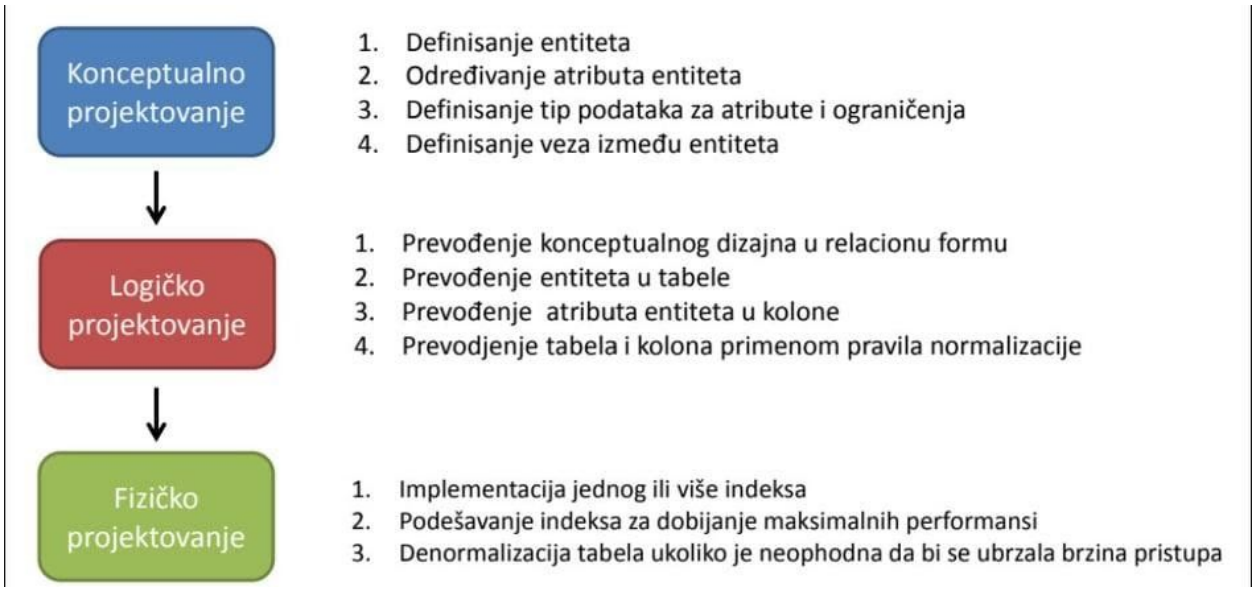
Logički model podataka opisuje podatke što je moguće detaljnije, bez obzira na to kako će se fizički implementirati u bazi podataka. Karakteristike logičkog modela su:

- uključuje sve entitete i relacije među njima
- Svi atributi svakog entiteta su tačno specificirani.
- Primenjuje se primarni ključ za svaki entitet
- Strani ključevi su takođe specificirani
- Na ovom nivou se odvija normalizacija

Fizičko projektovanje

Fizički model podataka pokazuje kako će se model graditi u bazi podataka. Fizički model baze podataka prikazuje sve strukture tabele, uključujući ime kolone, tip podataka kolone, ograničenja kolone, primarni i strani ključ i odnose između tabela.

Na slici 1.1. prikazane su faze projektovanja baze podataka i koraci u svakoj od njih.



Slika 1.1. Faze projektovanja baze podataka

Fizičko projektovanje baze podataka

Fizičko projektovanje baze podataka optimizuje performanse uz obezbedjivanje integriteta podataka, a izbegavajući nepotrebne viškove podataka. Zadatak fizičkog projektovanja je posao koji se zaista nikada ne završava. Potrebno je kontinuirano praćenje performansi i integriteta podataka kako vreme prolazi. Mnogi faktori zahtevaju periodična poboljšanja fizičkog dizajna.

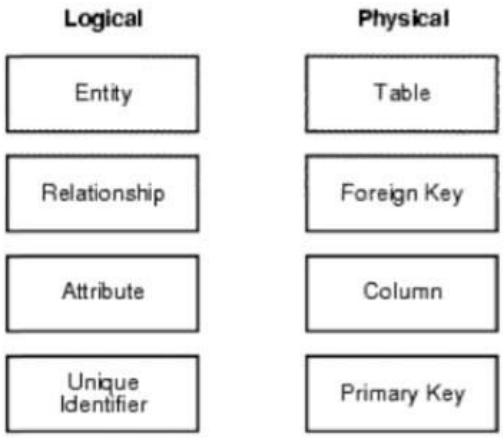
Fizičko projektovanje baze podataka je proces pretvaranja modela podataka u fizičku strukturu podataka određenog sistema za upravljanje bazama podataka (Database Management System – DBMS). Neke od karakteristika fizičkog modela podataka su:

- Fizički model podataka opisuje potrebu za podacima za jedan projekat ili aplikaciju, iako je možda integrisan sa drugim fizičkim modelima podataka zasnovanih na obimu projekta
- Model podataka sadrži odnose izmedju tabela koje se bave kardinalnošću i poništavanjem relacija
- Razvijen za specijalnu verziju DBMS-a, lokacije ili tehnologije koja će se koristiti u projektu
- Kolone treba da imaju tačne tipove podataka, dodeljene dužine i dodeljene početne vrednosti
- Definisani su i primarni i strani ključevi, indeksi, profili pristupa i autorizacije, itd.

Tabele i kolone se prave na osnovu informacija dobijenih tokom logičkog modelovanja. Primarni, jedinstveni i strani ključevi su definisani kako bi se obezbedila ograničenja. Podaci se mogu sažeti, a korisnicima se pruža alternativna perspektiva nakon što se kreiraju tabele. Fizičko modeliranje baze podataka zavisi od softvera koji se već koristi, znači da je softverski specifičan.

Budući da je fizičko projektovanje povezano sa načinom na koji se podaci fizički smeštaju, mora se razmotriti i nekoliko osnovnih koncepata o fizičkom skladištenju. Jedan od ciljeva su optimalne performanse i upotreba skladišnog prostora. Fizički dizajn uključuje strukture podataka i organizaciju datoteka, uzimajući u obzir da će softver baze podataka komunicirati sa

operativnim sistemom računara. Tipične zabrinutosti mogu biti oko dodeljivanja prostora za podatke i indekse, kompresija podataka, enkripcija itd. Na slici 2.1. grafički su prikazane razlike izmedju logičkog i fizičkog dizajna baze podataka



Slika 2.1. Grafički prikaz logičkog i fizičkog dizajna baze podataka

Strukture fizičkog dizajna

Prevodjenje šeme u konkretnu bazu podataka zahteva kreiranje sledećih stvari:

Prostori za tabele

Osnovni skladišni prostor se obično naziva prostor za tabele (engl. table space). Jedan prostor za tabele može da sadrži više tabela a u nekim sistemima jedna tabela može da bude i u više prostora za tabele. Na nivou prostora za tabele definišu se veličina fizičke stranice, način i uslovi baferisanja stranica, fizički uređaji (diskovi, particije, direktorijumi, fajlovi) koji čine taj prostor za tabele.

Particionisane tabele

Sadržaj tabele može da se podeli u više fizičkih celina (particija) na osnovu vrednosti ključa. Na primer, svi podaci o studijama bi mogli da se particionišu po godini upisa studenta čime se omogućava lakše arhiviranje starih podataka, različito upravljanje baferisanjem novih i starih podataka i slično.

Indeksi

Indeksi su pomoćne strukture podataka koje omogućavaju brže pristupanje podacima, odnosno brže pretraživanje po unapred izabranom ključu. Svaka tabela može da ima više indeksa sa različitim ključevima. Definicija indeksa obuhvata sledeće:

- kolone koje čine uslov uređivanja, odnosno ključ pristupanja
- odgovor na pitanje da li je indeks jedinstven ili nije
- odgovor na pitanje da li je indeks uređujući ili nije
- vrstu indeksa

Katanac

Uobičajan način implementiranja izolovanosti i transakcija pomoću mehanizma katanca. Postavljanje katanca na objekat je postupak koji obezbeđuje transakciji pristup objektu, i kojim transakcija istovremeno sprečava druge transakcije da pristupe tom objektu. Svaka transakcija na kraju svog izvršavanja otključava sve objekte koje je sama zaključala. Svaki katanac ima objekat koji se zaključava, trajanje i vrstu katanca. Objekat koji se zaključava može biti vrednost, red tabele, stranica tabele, cela tabela, prostor za tabele ili indeks.

Eskalacija katanca

Veličina katanca može da bude različita. Tako katanac može da zaključava jedan red ili više redova ili jednu stranicu ili više stranica. Veliki broj katanaca može značajno da uspori rad. Zato je broj katanaca ograničen. Kada se prevaziđe dopušten broj katanaca dolazi do eskalacije katanaca. Tada se više malih katanaca zamenjuje jednim većim. Na primer, više katanaca na redovima se zamenjuje katancem na stranici ili se više katanaca na stranicama zamenjuje katancem na tabeli.

Bafer za stranice

Bafer za stranice je memorijski prostor predvidjen za čuvanje kopije dela stranica jednog prostora za tabele, radi omogućavanja bržeg pristupa podacima. Što je bafer za stranice veći, to je broj pristupa disku manji. Dobro konfigurisanje prostora za tabele i bafera za stranice može da bude od presudnog uticaja na performanse.

Prevodjenje iz logičkog modela podataka u fizičku bazu podataka

Fizičko projektovanje baze podataka prevodi logički model podataka u skup SQL izraza koji definišu bazu podataka. Za sisteme relacionih baza podataka, relativno je lako prevesti logički model podataka u fizičku bazu podataka. Na kraju je kreirana SQL baza podataka u koju se mogu smeštati podaci.

Postoje 4 pravila prevodjenja iz logičkog modela podataka u fizičku bazu podataka:

1. Entiteti postaju tabele u fizičkoj bazi podataka.
2. Atributi postaju kolone u fizičkoj bazi podataka. Treba se izabrati odgovarajući tip podataka za svaku od kolona.
3. Jedinstveni identifikatori postaju kolone koje ne mogu da imaju NULL vrednosti. Ove kolone se nazivaju i primarni ključevi fizičke baze podataka. Moguće je kreirati jedinstveni indeks nad identifikatorima kako bi se iskoristila jedinstvenost.
4. Veze su modelirane kao strani ključevi.

Takodje, razmaci nisu dozvojeni u imenima entiteta u fizičkoj šemi jer se ta imena moraju prevoditi u SQL upite za kreiranje tabela. Nazivi tabela treba da budu u skladu sa SQL pravilima imenovanja. Atributi primarnog ključa mogu biti bilo koji tip podataka koji se može indeksirati. Primarni ključevi su u najvećem broju slučajeva tipa *int*, jer je mnogo brže pretraživati numerička polja u bazama podataka. Medjutim mogu biti i drugog tipa npr *char*, funkcionisanje je isto. Suština je da ovaj izbor treba da bude vođen kriterijumima za izbor identifikatora. Veze se modeliraju dodavanjem stranog ključa u jednu od tabela uključenih u odnos. Strani ključ je jedinstveni identifikator ili primarni ključ tabele sa druge strane veze. Najčešći odnos kod veza je 1 prema N. Taj odnos se preslikava postavljanjem primarnog ključa na strani “1“ u tabelu na strani “N“. Veze 1 prema 1 treba preslikati tako što se odabere jedna tabela, i dodeliti joj kolonu stranog ključa koja odgovara koloni primarnog ključa u drugoj tabeli.

Ukoliko uzmemo primer baze podataka za CD nakon primene prva tri pravila navedena iznad dobićemo fizičku bazu podataka prikazanu u tabeli 1:

Table	Column	Data Type	Notes
CD	CDId CDTitle	INT TEXT(50)	Primary Key
Artist	ArtistId ArtistName	INT TEXT(50)	Primary Key
Song	SongId SongName	INT TEXT(50)	Primary Key
RecordLabel	RecordLabelId RecordLabelName	INT TEXT(50)	Primary Key

Tabela 1. Definicija fizičke tabele za CD bazu podataka

U ovoj tabeli primećujemo da nijedan entitet u nazivu ne sadrži razmak, kao i to da su svi primarni ključevi tipa *int*.

Strani ključevi

Sa ovom tabelom imamo početnu tačku za kreiranje fizičke šeme. Ostalo je da prevedemo veze u fizički model podataka. Veze modelujemo dodavanjem stranog ključa u jednu od tabela koje pripadaju toj vezi. Strani ključ je jedinstveni identifikator, ili primarni ključ, tabele koja se nalazi sa druge strane veze. U ovom primeru nemamo nijednu 1 na 1 vezu. Kako bismo napravili veze u našoj tabeli potrebno je da se uradi sledeće:

- Staviti recordLabelId kolonu u tabelu CD
- Staviti CDId kolonu u SONG tabelu
- Staviti ArtistID kolonu u SONG tabelu

Tabela 2 prikazuje tabelu nakon primenjenih stavki

Table	Column	Data Type	Notes
CD	CDId	INT	Primary Key
	CDTitle	TEXT(50)	Foreign Key
	RecordLabelId	INT	
Artist	ArtistId	INT	Primary Key
	ArtistName	TEXT(50)	
Song	SongId	INT	Primary Key
	SongName	TEXT(50)	Foreign Key
	CDId	INT	Foreign Key
	ArtistID	INT	
RecordLabel	RecordLabelId	INT	Primary Key
	RecordLabelName	TEXT(50)	

Tabela 2. Fizički model podataka za CD bazu podataka

Sada imamo kompletnu fizičku šemu baze podataka. Ostaje još jedan zadatak, da prevedemo šemu u SQL. Za svaku tabelu u našoj šemi piše se jedna CREATE TABLE naredba. Tipično bi trebalo kreirati jedinstvene indekse na primarnim ključevima da bi se primenila jedinstvenost. Modeli podataka bi trebalo da budu nezavisni u odnosu na bazu podataka.

```
CREATE TABLE CD (
cd_id INT NOT NULL PRIMARY KEY,
cd_title VARCHAR(50)
record_label INT FOREIGN KEY REFERENCES Record_label(record_label_id));

 CREATE TABLE Artist (
artist_id INT NOT NULL PRIMARY KEY,
artist_name VARCHAR(50));
```

```
CREATE TABLE Song (
song_id INT NOT NULL PRIMARY KEY,
song_name VARCHAR(50),
song_length TIME,
cd_id INT FOREIGN KEY REFERENCES CD(cd_id),
artist_id INT FOREIGN KEY REFERENCES Artist(artist_id));

CREATE TABLE Record_label (
record_label_id INT NOT NULL PRIMARY KEY,
record_label_name VARCHAR(50));
```

Optimizacija podataka

Postoji nekoliko opcija u postizanju ciljanih performansi i sve one se mogu podeliti u tri kategorije:

1. Optimizacija na nivou interne organizacije podataka
    - Ostvaruje se kroz upravljanje internom organizacijom podataka, pomoćnim komponentama i resursima
    - Ne menja se logički model
2. Optimizacija na nivou upita
    - Vrši se pisanje upita na način koji omogućava njihovo efikasnije izvršavanje
    - Ne menja se logički model
3. Optimizacija na nivou strukture podataka
    - Fizička struktura podataka se menja u odnosu na logički model

Optimizacija na nivou interne organizacije podataka

Optimizacija na nivou interne organizacije podataka podrazumeva upravljanje fizičkom organizacijom podataka (prostori za tabele, stranice, baferi stranica), upravljanje pomoćnim komponentama (indeksima) i upravljanje memorijom.

Fizička organizacija podataka

Logička organizacija kao osnovno mesto čuvanja podataka vidi relaciju, odnosno tabelu. Fizička organizacija ide i dalje od toga. Ozbiljni sistemi omogućavaju veoma precizno upravljanje elementima fizičke organizacije podataka. Fizička organizacija podataka počiva na mnogo važnih koncepata kao što: prostori za tabele, stranice, baferi za stranice, particionisane tabele, kompresija podataka, katanci i različiti drugi koncepti često specifični za određene implementacije.

Indeksi

Indeksi su pomoćne strukture podataka koje omogućavaju brže pristupanje podacima, odnosno brže pretraživanje po unapred izabranom ključu. Svaka tabela može da ima više indeksa sa različitim ključevima.

Jedinstveni indeksi su indeksi koji ne dozvoljavaju ponavljanje istog ključa. Koriste se i kao sredstvo za implementiranje integriteta ključa.

Grupišući indeksi podrazumevaju da su redovi u tabeli poređani u odgovarajućem poretku. Dozvoljeno je imati najviše jedan takav indeks u tabeli. Značajno ubrzavaju izdvajanje podsekvence redova u datom poretku. Ovakvi indeksi se nazivaju i uređujući indeksi. Kandidati za grupišuće indekse su kolone koje se često traže u opsezima, po kojima se često uređuje rezultat, koje pripadaju stranom ključu po kome se najčešće vrši spajanje i kolone primarnog ključa. Prednosti koje donose ovi indeksi su višestruko ubrzano čitanje nizova redova po uslovu, a slabosti su dodatno usporeno održavanje i to što ne ubrzavaju pristupanje pojedinačnim redovima.

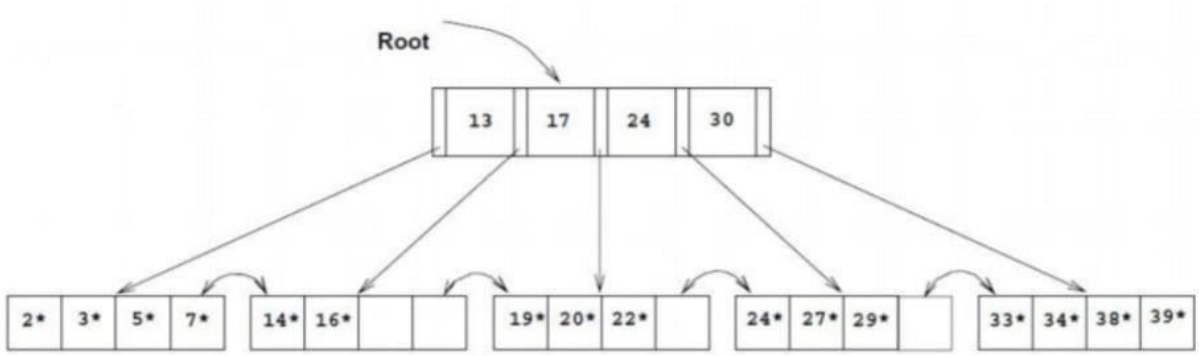
Bit-mapirani indeks je indeks koji se sastoji od niza vrednosti ključa. Iza svake vrednosti ključa sledi niz bitova, za svaki red tabele po jedan. Pri tome bit je jedan ako odgovarajući red ima baš tu vrednost ključa. Prednost ovih indeksa je u slučaju postojanja relativno malo različitih vrednosti ključa, jer tada ovakva struktura postaje efikasnija od B-stabla, a slabost je u slučajevima sa mnogo različitih vrednosti ključeva, kada indeks postaje veoma veliki, a time i slabo efikasan.

Indeksi sa strukturom B stabla – ideja je da se podaci čuvaju u balansiranom stablu:

- Svaki podatak je sadržan u listu jednake dubine
- Čvorovi i listovi sadrže po više podataka
- Veličina čvorova odgovara stranici diska

Danas su uobičajni indeksi upravo oni sa strukturom B-stabla. Prednosti ovih indeksa su jednostavni i efikasni algoritmi za održavanje, a slabosti su što ne rade posebno dobro ako je

mnogo redova a malo različitih vrednosti ključa, odnosno ako ima mnogo ponavljanja ključa. Na slici 4.1. su prikazani indeksi sa strukturom B stabla



Slika 4.1. Prikaz indeksa sa strukturom B stabla

Heš tabele su alternativa klasičnim indeksima. Ovde se računaju heš vrednosti na osnovu ključnih atributa, a onda se pomoću dobijene vrednosti neposredno pristupa podacima. Alternativa ovom pristupu bi bilo da se vrši indeksiranje po heš vrednostima. Prednost heš tabela je efikasnije pristupanje pojedinačnim redovima sa tačno zadatim ključem, a slabost je ta što heš tabele nisu dobre za sekvencijalno pristupanje većem broju redova ili ako operator poređenja nije jednakost.

Idealan broj i vrsta indeksa zavise od vrste, namene i strukture tabele i baze podataka, kao i načina upotrebe.

Upravljanje memorijom

Važno je da se pri administriranju SUBP (Sisem za upravljanje bazom podataka) dobro upravlja memorijom. Ako se memorija ne koristi dovoljno, ili se koristi pogrešno, suviše će se pristupati disku. SUBP ne koristi virtualnu memoriju za pristupanje stranicama baze podataka, već samo bafere stranica. SUBP koristi memoriju na različite načine što uključuje: glavne i pomoćne bafere stranica, liste katanaca, keš kataloga baze podataka, keš paketa, interni hip baze podataka, hip pomoćnih alata, memoriju agenata..

Optimizacija upita

Da bi se neki upit izvršio nad bazom podataka, najpre se pravi plan izvršavanja upita. Ovaj plan najčešće obuhvata sledeće:

- Redosled koraka
- Operacije koje se izvršavaju u pojedinim koracima
- Strukture podataka koje se upotrebljavaju
- Način svakog pojedinačnog pristupanja podacima
- Procenjenu cenu svakog od koraka i celog posla

Većina savremenih SUBP omogućava korisniku da sagleda plan izvršavanja pre nego što se pokrene izračunavanje upita. Pri računanju cene pojedinačne operacije uzimaju se u obzir sledeće operacije:

- Cena uredjaja
- Broj redova u tabeli
- Popunjenost stranica
- Selektivnost upita
- Cena same operacije

Većina tih informacija je tačna samo ako su ažurni statistički podaci o sadržaju baze podataka.

Katalog kao sistemska baza podataka ima veliku važnost u procesu optimizacije upita. On sadrži informacije o raznim objektima u sistemu kao što su bazne tabele, pogledi, indeksi, baze podataka, planovi izvršavanja upita. Karakteristično za katalog je da obično uključuje i tabelu koja sadrži po jednu n-torku za svaku tabelu u sistemu, uključujući i sistemske tabele, zatim tabelu svih atributa svih tabela u sistemu, tabelu svih indeksa svih tabela u sistemu kao i tabele pogleda, baza podataka, uslova integriteta, stranih ključeva, autorizacije, optimizovanih upita, sinonima itd. Ove tabele sadrže informacije o imenu objekta, vlasniku, broju atributa, tabeli za koju je atribut ili indeks vezan, tip atributa itd. Obzirom da je sistemski katalog relacionog sistema relaciona baza podataka, ona se može pretraživati istim upitnim jezikom kao i korisničke

baze podataka. Međutim, tabele kataloga se ne mogu ažurirati iskazima ažuriranja upitnog jezika jer to može biti vrlo opasno po podatke u bazi i pristup podacima.

Optimizacija upita je manuelno ili automatsko odabiranje najpovoljnijeg plana izvršavanja upita radi postizanja boljih performansi. Većina savremenih SUBP raspolaže solidnim optimizatorima upita. Optimizacija upita je komponenta procesora upitnog jezika koja je neophodna, bar kod velikih sistema i za velike baze podataka, da bi sistem uopšte mogao da zadovolji traženu efikasnost. Posebnu pogodnost kod optimizacije upita pružaju relacioni sistemi, s obzirom na dovoljno apstraktni nivo relacionih izraza kojima se upiti zadaju.

Automatska optimizacija daje značajne prednosti u kvalitetu, jer čovek često nije u mogućnosti da postigne kvalitet optimizacije koji postiže sistem. Ova prednost sistema nad čovekom je posledica činjenice da sistem „ima uvid“ u vrednosti podataka (a ne samo u njihovu strukturu) koji čovek – korisnik nema, kao i, s obzirom na brzinu rada sistema, posledica mogućnosti proizvođenja većeg broja alternativnih načina izvršavanja upita i procene njihove efikasnosti. Optimizator koristi statističke informacije iz sistemskog kataloga, formule za procenu veličine međurezultata i cene operacija niskog nivoa.

Manuelna optimizacija upita je ranije bila mnogo potrebnija nego danas. Na različite načine se SUBP može sugerisati koji plan izvršavanja da primeni. Danas je uloga manuelne optimizacije upita važnija u fazi pravljenja fizičkog modela nego u eksploataciji. Ne optimizuje se upit, nego se analiziraju planovi izvršavanja radi sagledavanja i prevazilaženja eventualnih slabosti predloženog modela. Pri pronalaženju najboljeg plana izvršavanja uzimaju se u obzir raspoloživi podaci o bazi podataka iz kataloga:

- struktura tabela i ključeva
- struktura upita
- postojeći indeksi
- statistički podaci o sadržaju tabela i atributa
- podaci o brzini fizičkih uređaja
- veličina bafera stranica

Optimizacija na nivou strukture podataka

Ako imamo uzorak upita i njihove planove izvršavanja, onda možemo da sprovedemo optimizaciju strukture baze podataka u cilju povećavanja performansi uzorka upita. Ovde govorimo o optimizaciji fizičkog dizajna menjanjem fizičke strukture baze podataka.

Uzorak upita je skup tipičnih upita za koje se zna da će činiti najčešći oblik pristupanja bazi podataka. Dobar uzorak se sastoji od:

- skupa upita i promena baze podataka
- procenjene učestalosti izvršavanja i posebno vršno opterećenje za svaki upit i promenu
- formulisanih ciljnih performansi

Na osnovu uzorka

- Za upite:
    - Prepoznaje se koje se relacije koriste
    - Koji atributi se izdvajaju
    - Koji atributi učestvuju u uslovima spajanja i restrikcije
- Za promene baze, pred toga se prepoznaje i:
    - Vrsta promene
    - Atributi koji se menjaju

Cilj strukturne optimizacije je prepoznavanje indeksa koje moramo da napravimo kao i prepoznavanje potrebnih promena u fizičkoj šemi:

- Alternativna normalizacija
- Denormalizacija
- Uvođenje pogleda da bi se sakrile načinjene promene

Alternativna normalizacija

Pri normalizovanju ne postoji samo jedno moguće rešenje. Ukoliko jedno rešenje ne daje zadovoljavajuće rezultate, možda će drugo dati.

Primeri različitih normalizacija:

Neka relacija P opisuje projekte: P(*BrProjekta, Grad, Naziv*)

Imamo sledeće funkcionalne zavisnosti:

• fd1: *BrProjekta → Grad*

• fd2: *BrProjekta → Naziv*

Ako u jednom gradu postoji najviše jedan projekat i svi projekti imaju različite nazive, onda možemo da smatramo da postoji i:

• *fd3: Grad → Naziv*

• *fd4: Naziv → Grad*

• *fd5: Naziv → BrProjekta*

Onda sledi da imamo alternativne normalizacije:

Normalizacija 1

P(BrProjekta, Grad)

PN(Grad, Naziv)

Normalizacija 2

P(BrProjekta, Naziv)

PG(Naziv, Grad)

Normalizacija 3

P(BrProjekta, Grad)

PN(BrProjekta, Naziv)

Denormalizacija

Neke tehnike denormalizacije su:

- Podela tabele (dekompozicija)
- Spajanje tabela (kompozicija)
- Dupliranje podataka

Dekompozicija

Ako imamo tabelu sa mnogo redova ili kolona, koja se često menja, onda to može da bude neefikasno jer indeksi postaju „duboki“ i često se čita sa diska a i upiti koji ne koriste indekse su tim pre neefikasni. Jedan način da se rad ubrza je da se tabela podeli na dve ili više. Postoje dve vrste podele: horizontalna i vertikalna.

Kod horizontalne podele tabele redovi tabele se podele u dve tabele sa istom strukturom. Pri tom obično jedna sadrži takozvane nove ili aktivne redove, a druga takozvane stare ili arhivske podatke. Dobra strana ovakve vrste podele je što ubrzavaju neke operacije (pre svega menjanje i dodavanje novih podataka), a loša strana je što se dodatno komplikuju neke operacije.

Kod vertikalne podele podataka, ako se većina kolona ne koristi u svim upitima, već samo relativno retko, onda se česte operacije mogu ubrzati vertikalnom podelom. Prave se dve ili više tabela od kojih svaka sadrži kolone primarnog ključa a sve ostale kolone se grupišu prema upotrebi. Dobra strana ove vrste podele je što se ubravaju neke česte operacije a loša strana što se takodje dodatno komplikuju neke operacije, pre svega analitički upiti nad svim podacima.

Kompozicija

Tabele koje se često spajaju u upitima, može da bude korisno spojiti u jednu tabelu. Pozitivna strana je to što je spajanje veoma skupa operacija pa ovakva promena može da ima značajne efekte. Loše strane su:

- Dobijena tabela obično narušava normalne forme i sadrži neke redundantnosti
- To dodatno otežava održavanje podataka i staranje o integritetu
- Može da se značajno poveća broj redova ili ukupno zauzeće prostora

Dupliranje podataka

Jedna od tehnika optimizacije može da bude da se neki podaci svesno ponove u više tabela. Dobra strana dupliranja podataka je to što se izbegavaju suvišna spajanja ili unije, a loše strane je redundantnost, odnosno otežano održavanje i povećano zauzeće prostora.

Uvođenje pogleda da bi se sakrile načinjene promene

Ako se promeni fizička struktura baze podataka, onda se na konceptualnom nivou uvode novi pogledi koji skrivaju načinjene izmene. Ovo nije deo fizičkog modela, ali je posledica promena u fizičkom modelu.

Literarura

[1] Dr Jelena Graovac, “Projekotvanje baza podataka”, Web-sajt
http://poincare.matf.bg.ac.rs/~jgraovac/courses/projbp/2016_2017/projbp_skripta.pdf?fbclid=IwAR3vpws_HnMlD0kPgMNV7y0KYaTJqFLTWuGDm4Nq4L5K-rgTwNrkQpgzAnw

(pristupljeno marta 2020)

[2] “What is data modelling?”, Web-sajt
https://www.guru99.com/data-modelling-conceptual-logical.html?fbclid=IwAR1OTDBCiIOH6Aa5DWxB18JKNA-RueigxgwKj_Evg0UH1_w3WgVMhHfbm68

(pristupljeno aprila 2020)

[3] O’Reilly, “Physical database design”, Web-sajt
https://docstore.mik.ua/orelly/weblinux2/mysql/ch07_04.htm?fbclid=IwAR2e62_fMESt_A5n3pa_vyDgxRWtdzUVNr8ZX-P1e-IRZzM0tEzMX7tIwQ8

(pristupljeno aprila 2020)

[4] eWebArchitexture, “Physical database design”, Web-sajt
http://ewebarchitecture.com/web-databases/physical-database-design?fbclid=IwAR1wFk6ARPmqblCKhwYmT7p23C57PFlZx59CZTgO7bJESeBOSBrCT2moHgM

(pristupljeno aprila 2020)