Fizično načrtovanje podatkovnih baz

Iztok Savnik, FAMNIT

Pregled

- Po načrtovanju z ER in logičnem načrtovanju dobimo konceptualni načrt podatkovne baze.
- Naslednji koraki.
  - Izbira indeksov
  - Odločitve glede povezanosti indeksov.
  - Optimizacija konceptualnih shem.
  - Optimizacija vprašanj.
- Podatki za fizično načrtovanje
  - Delovna obremenitev.
    - Najpomembnejša vprašanja in pogostost izvajanja.
    - Najpomembnejši popravki in pogostost izvajanja.
    - Željene performanse za vprašanja in popravke nad podatkovno bazo.

Fizično načrtovanje

- Implementacija logičnega načrta
- Pohitritev vprašanj
- Spremembe fizičnega načrta za boljšo izvedbo

Fizično načrtovanje

- Implementacija logičnega načrta
  - Katere relacije naj imajo indekse? Kateri atributi naj bodo v iskalnem ključu? Naj kreiramo več indeksov nad relacijo?
  - Kakšen indeks izbrati? Povezan? Razpršilni/drevesni?
- Pohitritev vprašanj
  - Kako optimizirati pomembna vprašanja. Napake v poizvedbah. Restrukturiranje poizvedb.
- Spremembe fizičnega načrta za boljšo izvedbo
  - Potrebne spremembe konceptualne sheme? Alternativne normalizirane sheme? Obstaja več možnih dekompozicij.
  - Denormalizacija? Nižja normalna oblika primernejša?
  - Horizontalna dekompozicija, replikacija, pogledi ...

Podatki za fizično načrtovanje

- *“Delovna obremenitev”* vsebuje vse poizvedbe, ki jih dana aplikacija generira.
- Splošno:
  - Študij tipov UPDATE/INSERT/DELETE/UPDATE in atributov, ki sodelujejo.
- Za vsako selekcijo v *delovni obremenitvi:*
  - Katere relacije dosegajo?
  - Katere atribute izbere?
  - Kateri atributi sodelujejo pri pogojih za izbiranje in stike? Kako selektivni so pogoji?
- Za vsak update v delovni obremenitvi:
  - Kateri atributi sodelujejo v pogoju izbire ali stika? Kako selektivni so pogoji?

Implementacija logičnega načrta

- Izbor indeksov
  - Lastnosti indeksov
  - Uporaba indeksov za poizvedbe
  - Vrste indeksov
- Fizično shranjevanje podatkov
  - Tablespace
  - Prazen prostor
  - Uporaba diskov
- Uporaba spomina

Izbor indeksov

- Tip indeksa:
  - Razpršilni indeksi
  - Drevesni indeksi
- Lastnosti indeksa:
  - Primarni/sekundarni indeks
  - Povezan/nepovezan indeks
- Indeks z več atributi v iskalnem ključu:
  - Veliko število pogojev v iskalnih zapisih.
  - Vrstni red atributov pri indeksu z več atributi je pomemben.
- Evaluacija poizvedbe samo z indeksi:
  - Povezanost indeksa s tabelo ni pomembna pri poizvedbah samo z indeksi.

Selekcija indeksov za stike

- Izbor glede na pogoj stika:
  - Razpršilni indeks je dober za uporabo vgnezdene zanke z indeksom (enakost).
  - Drevesni indeks je dober za uporabo vgnezdene zanke z indeksom (področje).
    - Enakost z velikim številom ujemanj
  - Povezanost indeksa:
    - Povezan v primeru, da pogoj ne vsebuje ključa notranje relacije.
    - Notranje n-terice so potrebne višje v drevesu poizvedbe.
  - *Povezano* B+ drevo nad pogoji stika je primerno za stik z zlivanjem.

Primer 1

```
SELECT  Z.zime, O.sef
FROM  Zaposleni Z, Oddelki O
WHERE  O.oime='Igrace' AND Z.oid=O.oid
```

- Razpršilni indeks na *O.oime* podpira selekcijo "*Igrace*".
  - Zunanja relacija je *Oddelki*.
  - Če imamo ta indeks ni potreben indeks na *O.oid*.
- Razpršilni indeks na *Z.oid* omogoči impl. stika.
  - Indeks poskrbi za ujemanje notranje relacije (*Zaposleni*) za vsako zunanjo n-terico (*Oddelek*).
- WHERE vsebuje: `` ... AND Z.star=25'' ?
  - N-terice *Zaposleni* lahko poiščemo z indeksom na *Z.star*, potem naredimo stik *z Oddelki* C, ki zadoščajo selekciji z *oime*. Primerljivo s strategijo z uporabo indeksa na Z.oid.
  - Če je torej indeks na Z.star že kreiran imamo manj motivacije za kreiranje indeksa na *Z.oid*.

Primer 2

```
SELECT  Z.zime, O.sef
FROM  Zaposleni Z, Oddelki O
WHERE  Z.placa BETWEEN 20000 AND 30000
  AND Z.hobi='Znamke' AND Z.oid=O.oid
```

- *Zaposleni* je lahko zunanja relacija.
  - Zakaj? Majhno število n-teric.
  - Razpršilni indeks na Z.oid?
  - Lahko uporabimo B+ drevo na Z.placa, ALI indeks na Z.hobi.
    - Potrebujemo samo enega od obeh (dostop do relacije z enim).
    - Izbor je odvisen od selektivnosti.
    - Po "pravilu" je običajno selekcija po enakosti boljša.
- Kot oba primera predlagata, je naša izbira odvisna od plana, ki ga pričakujemo od optimizatorja.
  - *Potrebno je razumevanje delovanja optimizatorja!*
  - Ad-hoc izbor: *atribut v poizvedbi* ➔ *indeks*, ni nujno da vedno deluje.

Povezani indeksi in stik

```
SELECT  Z.zime, O.sef
FROM  Zaposleni Z, Oddelki O
WHERE  O.oime='Igrace' AND Z.oid=O.oid
```

- Povezan indeks je posebno pomemben pri dostopu do notranjih n-teric.
  - *Oddelki* je zunanja relacija. Dobro je imeti indeks na *Z.oid* povezan.
- Recimo, da WHERE stavek vsebuje:

  WHERE Z.hobi='Znamke' AND Z.oid=O.oid
  - *Zaposleni* damo kot zunanjo relacijo. Možna uporaba stika z zlivanjem. Povezan indeks na *O.oid* lahko pomaga.
- *Povzetek*: Povezan indeks je pomemben vedno, ko se pri selekciji ujema veliko n-teric.

Shranjevanje podatkov

- Tablespace
  - Podatkovna področja, ki vsebujejo tabele
  - Grupiranje tabel v fizično enoto
- Prazen prostor
  - Vsak blok ima rezerviran prazen prostor za nove zapise
  - Velikost praznega prostora lahko definiramo
  - Začetno je branje počasnejše

Uporaba RAM

- Strežniki imajo velike količine RAM
  - Velikost bazena strani lahko definiramo
  - Manjše tabele lahko shranimo v RAM
  - Delo sistema se pohitri za več nivojev velikosti
- Podatkovne baze v RAM
  - > 100 GB RAM

Umerjanje poizvedb

- Restrukturiranje SQL poizvedb
- Zaklepanje

Umerjanje poizvedb

- Če poizvedbe tečejo počasi preveri uporabo indeksov ter starost statistik.
- Včasih SUPB ne izvaja plana, ki ga uporabnik pričakuje. Običajno v primerih:
  - Selekcije vsebujejo vrednosti null.
  - Selekcije vsebujejo operacije nad nizi ali števili.
  - Selekcije vsebujejo pogoje z OR.
  - SUPB ne vsebuje vseh operacij: strategije z indeksi, algoritme za stike, ali vsebuje slabo oceno velikosti.
- Preveri plan, ki ga sistem uporablja in potem prilagodi indekse, restrukturiraj vprašanje, itd.

Restrukturiranje SQL poizvedb

- Komplicirano, če sodelujejo:
  - NULL, duplikati, agregacijske funkcije, pod-vprašanja.
- Smernica: Uporabi en blok poizvedbe, če je mogoče.

```
SELECT DISTINCT *
  FROM Mornarji M
 WHERE M.mime IN
        (SELECT L.mime
           FROM MladiMornarji L)
```

=

```
SELECT DISTINCT M.*
  FROM Mornarji M,
       MladiMornarji L
 WHERE M.mime = L.mime
```

- Ni vedno mogoče ...

```
SELECT *
  FROM Mornarji M
 WHERE M.mime IN
        (SELECT DISTINCT L.mime
           FROM MladiMornarji L)
```

≠

```
SELECT M.*
  FROM Mornarji M,
       MladiMornarji L
 WHERE M.mime = L.mime
```

Kompleksna vprašanja

- Izogibamo se začasnim tabelam
- Iščemo povpr. plačo zaposlenih iz oddelkov, ki jih vodi 'Mara'.
- Bolje je ne kreirati začasno tabelo
- Prepustimo optimizacijo...

```
SELECT * INTO Temp
FROM Zaposleni Z, Oddelki O
WHERE Z.oid=O.oid and O.sef='Mara';

SELECT T.oid, AVG(T.placa)
FROM Temp T
GROUP BY T.oid;;
```

```
SELECT T.oid, AVG(T.placa)
FROM Zaposleni Z, Oddelki O
WHERE Z.oid=O.oid and O.sef='Mara'
GROUP BY T.oid;
```

Slaba uporaba COUNT

```
SELECT oime FROM Oddelki O
 WHERE O.st_zap  >
        (SELECT COUNT(*) FROM Zaposleni Z
          WHERE O.stavba = Z.stavba)
```

```
CREATE VIEW Temp (st_zap, stavba) AS
          SELECT COUNT(*), Z.stavba
            FROM Zaposleni Z
          GROUP BY Z.stavba

SELECT oime
  FROM Oddelki O,Temp
 WHERE O.stavba = Temp.stavba
   AND O.st_zap > Temp.st_zap;
```

Smernice za uglaševanje poizvedb

- Minimiziraj uporabo DISTINCT:
  - ne uporabljaj, če so duplikati sprejemljivi ali
  - ne uporabljaj v primeru, da odgovor vsebuje ključ.
- Minimiziraj uporabo GROUP BY in HAVING:

```
SELECT  MIN (Z.star)
FROM  Zaposleni Z
GROUP BY  Z.oid
HAVING  Z.oid=102
```

```
SELECT  MIN (Z.star)
FROM  Zaposleni Z
WHERE  Z.oid=102
```

- Indeksi in aritmetične operacije:
  - Z.star=2*D.star
  - Indeks na *Z.star* pomaga ne pa tudi na *D.star* !

Zaklepanje

- Več nivojev zaklepanja
  - Zapisi, bloki, tabele
- Širjenje zakepanja
  - Zaklepanje se širi hierarhično
- Strategije zaklepanja
  - Uporabnik lahko izbira strategije zaklepanja

Umerjanje načrta PB

- Spremembe vplivajo na logični načrt
- Umerjanje konceptualne sheme
  - Denormalizacija
  - Dekompozicija tabel
  - Združevanje tabel
  - Replikacija

Umerjanje konceptualne sheme

- Končni izbor konceptualne sheme mora biti voden z delovno obremenitvijo sistema.
  - V prejšnji fazi je bila pomembna redundatnost.
  - Uporabimo lahko 3NO namesto BCNO.
  - Izbor alternativne dekompozicije v 3NO oz. BCNO.
  - Shemo v BCNO lahko še naprej razcepimo !
  - Lahko denormaliziramo (spremenimo dekompozicijo) shemo tako, da dodamo atribute relacijam.
  - Horizontalna dekompozicija omogoča porazdelitev večje količine podatkov.
- Paziti je potrebno na potencialno izgubo podatkov ter na ohranitev pomena podatkov.

Primer relacijske sheme

Pogodbe (pid, bid, jid, oid, did, kol, vredn)
Oddelki (oid, vredn, porocilo)
Dobavitelji (bid, naslov)
Produkti (did, cena)
Projekti (jid, vodja)

- Poglejmo si relacijo Pogodbe, katere shemo označimo z PBJODKV. Veljajo naslednje odvisnosti: JD → P, BO → D, P je primarni ključ.
  - Kaj so kandidatni ključi za PBJODKV?
  - V kakšni normalni obliki je shema?

Primer relacijske sheme

Pogodbe(*pid,bid,jid,oid,did,kol,vredn*)

- P je ključ: P → PBJODKV
- Projekt nabavi vsak produkt z eno samo pogodbo: JD → P
- Oddelek kupi največ en produkt od enega dobavitelja: BO → D

3NO vs. BCNO

- PBJODKV lahko razcepimo v BOD and PBJOKV, in obe relaciji sta v BCNO.
  - Katera FO predlaga takšno dekompozicijo ?
  - Brezizgubna dekompozicija, ne ohranja odvisnosti.
  - Dodamo PJD: nova shema ohranja odvisnosti.
- Recimo, da je naslednje vprašanje pomembno:
  - *Poišči število primerkov K produkta D naročenega s pogodbo P.*
  - Zahteva stik novih shem.
  - Originalna shema: enostaven pregled relacije PBJODKV.
  - Lahko vzrok za izbor relacije v 3NO t.j. shema PBJODKV.

Denormalizacija

- Recimo, da je naslednje vprašanje pomembno:
  - *Je vrednost pogodbe večja kot premoženje oddelka?*
- Da bi pohitrili izvajanje vprašanja lahko dodamo *premoženje* (R) v relacijo *Pogodbe*.
  - Nova FO $O \rightarrow R$ v Pogodbah.
  - Relacija *Pogodbe* ni več v 3NO (denormalizacija).
- Kaj narediti?
  - Sprememba relacije Pogodbe, če je vprašanje pomembno in ni moč dobiti primerne hitrosti izvajanja vprašanja z uporabo dodatnih indeksov ali z alternativno shemo v 3NO.

Izbira dekompozicije

- Dva načina dekompozicije PBJODKV v BCNO:
  - BOD in PBJOKV; brez-izguben stik, ne ohrani FO.
  - BOD, PBJOKV in PJD; ohrani tudi FO.
- Razlika je samo v ceni za ohranitev odvisnosti JD → P.
  - 2. dekompozicija: Indeks za JD na relaciji PJD.
  - 1. dekompozicija:

```
CREATE ASSERTION   PreveriOddelke
CHECK  ( NOT EXISTS  ( SELECT  *
FROM  Dobavitelji B, Produkti D, Pogodbe P
WHERE  B.bid=P.bid AND D.did=P.did
GROUP BY  P.bid, P.did
HAVING  COUNT (P.pid) > 1  ))
```

Izbira dekompozicije

- Naslednje omejitve so dane:
  - JD → P, BO → D, P je primarni ključ.
  - Dobavitelj vedno računa isto ceno za dan produkt: BDK → V.
- Dekompozicija PBJODKV v BCNO:
  - Začni z dekompozicijo v BDKV in PBJODK.
  - Razcepi PBJODK (ni v BCNO) v BOD, PBJOK.
  - Brez-izgubna dekompozicija: BDKV, BOD, PBJOK.
  - Za ohranitev JD → P lahko dodamo PJD kot prej.
- Izbira: { BDKV, BOD, PBJOK } ali { BOD, PBJOKV} ?

Horizontalna dekompozicija

- Naša definicija dekompozicije: Relacijo zamenjamo z množico relacij, ki so *projekcije* originalne relacije.
  - Najbolj pomembna dekompozicija.
- Včasih želimo zamenjati relacijo z množico relacij, ki so *podmnožice* originalne relacije.
  - Vsaka nova relacija ima enako shemo kot originalna relacija, vendar vsebuje podmnožico vrstic.
  - Skupno nove relacije vsebujejo vse podatke originalne relacije. Preseki med novimi relacijami so prazni.

Horizontalna dekompozicija

- Recimo, da imajo pogodbe z vrednostjo >100000 drugačna pravila od preostalih.
  - Poizvedbe nad tabelo *Pogodbe* bodo pogosto vsebovale *vrednost>100000*.
- En način za delo s tabelo Pogodbe je konstrukcija povezanega B+ dreve-sa na *vrednost* pogodbe.
- Drugi način je zamenjava tabele Pogodbe z dvema tabelama: *VelikePogodbe* in *MajhnePogodbe*.
  - Shema ostane enaka PBJODKV
  - Porazdelitev deluje kot indeks. Ni dejanskega indeksa.
  - Nad tabelami lahko zgradimo različne (povezane) indekse.

Horizontalna dekompozicija

- Porazdeljene podatkovne baze
- Porazdelitev tabele na več različnih strežnikov
  - Razpršilna funkcija nad zapisi
  - Predikat, ki določa porazdelitev
- Pogosta rešitev za pohitritev dostopa
- Klasifikacija razredov v pod-razrede

Vertikalna dekompozicija

- Tabele s podmnožicami atributov
  - Dobimo s projekcijo
  - Originalno tabelo dobimo s stiki
- Velikost zapisa
  - Velik zapis, manj zapisov/blok, hitrejše branje sekvenčnih zapisov
  - Manjši zapisi več jih gre v RAM
  - Nekatera polja so lahko zelo velika
- Uporabimo isti primarni ključ

Združevanje tabel

- Združevanje tabel, ki se praviloma stikajo
  - Hitrejši dostop do podatkov (zapisi so fizično skupaj)
  - Ni potreben stik
  - Ključ je sestavljen iz ključev združenih tabel
  - Lahko imamo null vrednosti

Replikacija

- Kopija tabele na več strežnikih
- Dostop se lahko porazdeli
  - Load-balancing med kopijami
- Problemi
  - Popravljanje/vstavljanje zapisov
  - Kopije se morajo spreminjati sinhrono
  - Spreminja se ena tabela, prožilec širi spremembe na kopije

Pregled načrtovanja relacijskih podatkovnih baz

- Načrtovanje podatkovnih baz vsebuje:
  1) *Analizo zahtev.*
  2) *Konceptualno načrtovanje.*
  3) *Logično načrtovanje.*
  4) *Fizično načrtovanje in umerjanje.*
- Razvojni cikel načrtovanja podatkovnih baz:
  - Iteracija po naštetih opravilih – naprej in nazaj.
  - Odločitev zgoraj vpliva na načrt spodaj – višje je sprememba večji je vpliv.
- Dobro razumevanje delovne obremenitve sistema in ciljev načrtovanja je ključno za dober načrt podatkovne baze.
  - Kaj so pomembna vprašanja? Kateri atributi so vpleteni? …

Povzetek

- Umerjanje konceptualne sheme:
  - Upoštevamo performančne zahteve za poizvedbe.
  - Upoštevamo delovno obremenitev.
- Alternative pri umerjanju shem:
  - Včasih izberemo 3NO ali nižjo NO namesto BCNO.
    - Lahko izbiramo med različnimi dekompozicijami v BCNO.
    - Lahko *denormaliziramo* shemo.
    - Lahko razcepimo shemo v BCNO se naprej.
    - Ohranjanje informacij pri dekompoziciji je najpomembnejše.
    - Ohranjanje FO je pomembno; lahko dosežemo tudi z SQL.

Povzetek

- Indekse je potrebno vzdrževati:
  - S časom se spreminjajo. Nove potrebe, nove tabele, dodajanje zapisov med obratovanjem.
  - Brisanje, kreiranje, ponovno kreiranje (re-build).
- Umerjanje pomembnih poizvedb:
  - Potrebno je poznati plan sistema in potem prilagoditi indekse.
  - Izogibanje napakam.
  - Izogibanje kompleksnim poizvedbam: vgnezdeni bloki, group by, distinct,...
  - Sistem lahko navkljub umerjanju izbere slab plan.