Loeng6. Andmebaasi häälestamine (järg), käsufailide genereerimine ja käivitamine

Loengu plaan

- Andmebaasi objektide paigutuse planeerimine (< häälestamine)
- Andmebaasi objektide laadimise optimeerimine (< häälestamine)
- Koodi häälestamine (< häälestamine)
- Käsufailide genereerimine
- Käsufailide käivitamine
- Oracle'i struktuuridest

Andmebaasi objektide paigutuse planeerimine

Andmebaasi objektide paigutuse planeerimisega on võimalik optimeerida sisend / väljund operatsioonide teostamise kiirust, minimeerida võimalus, et kettaseadmete poole pöörduvad serveri protsessid peaksid üksteise taga ootama.

Sisend-väljund operatsioonide optimeerimise aluseks on nende operatsioonide jagamine erinevate füüsiliste kettaseadmete vahel. Selle teostamisel soovitatakse lähtuda järgmistest kriteeriumitest:

1. Võimalusel paigutada erinevatele kettaseadmetele andmefailid ja logfailid, s.o. seda tüüpi failid, mille poole server küllalt sageli pöördub:
    - Vajaduse tekkimisel ja võimaluste olemasolul paigutada iga andmefail eraldi kettale,
    - Paigutada logfailid üksteisest eraldi sellistele ketastele, kus samaaegselt muid sisend-väljundoperatsioone ei teostata. Sellega tagatakse logfailidesse kirjutamise protsessi (LGWR) maksimaalne efektiivsus (logfailidesse kirjutamine toimub jadamisi ning on märksa efektiivsem, kui kettaseadmel muud samaaegset tegevust pole). Erinevatel kettaseadmetel paiknevate sama järjekorranumbriga logfailidesse kirjutamine toimub paralleelselt ning seetõttu ei põhjusta taoline logfailide peegeldamine (dubleerimine) olulist kaotust töökiiruses. Lisaks on logfailide peegeldamine oluline ettevaatusabinõu süsteemi ohutuks

funktsioneerimiseks (ketta rikkega ei kaotata korraga kõiki logfaile).

2. Paigutada võimalusel erinevatel kettaseadmetel paiknevatesse andmefailidesse tabelid ja indeksid (hoida indekseid tabelitest lahus, nii on kergem leida järjestikust vaba ruumi (ekstent)).

3. Vajaduse tekkimisel (on ette näha väga suuri tabeleid) jagada suured tabelid ühe ja sama tabeliruumi sees erinevatel kettaseadmetel paiknevate failide vahel.

Failide paigutamine eraldi kettaseadmetele kirjeldatakse teadmusbaasis ning tegeliku andmebaasi loomise käsufailid genereeritakse sellest loogikast lähtuvalt.

Andmebaasi objektide laadimise optimeerimine

Andmebaasi objektide optimaalse laadimise (hoidmise) andmeblokkidesse peavad ühiselt tagama süsteemi administraator ja projekteerija (disainer). Andmebaasi objektide optimaalse laadimise eelduseks on andmemahtude hinnangu olemasolu.

Vaatame mõningate (olulisemate) laadimisparameetrite mõju objektidega opereerivate rakenduste töökiirusele.

1. PCTFREE (vaikimisi 10) näitab, mitu % andmebloki kogumahust võib veel olla vaba, kui vastav blokk kuulutatakse tegelikult täis olevaks. Kui bloki vaba ruum on saanud väiksemaks, kui PCTFREE, siis taolisse blokki enam kirjeid ei lisata (ei lisata seni, kuni bloki täidetud osa on veel suurem või võrdne parameetri PCTUSED väärtusega).

Antud parameetrile optimaalse väärtuse leidmiseks tuleks lähtuda järgmistest kriteeriumitest:
1)väiksem väärtus (5-10):

- Optimeerib kettaruumi kasutamist (rohkem ridu ühes andmeblokis), kuid tõstab andmebloki poole pöörduvate operatsioonide (lisamine, muutmine) sooritamise aega, kuna:
  - Tõenäosus blokkide reorganiseerimise vajaduse tekkimiseks muutmisoperatsiooni käigus on suurem,

- Võivad tekkida aheldatud read (kogu rida ei mahu ühte blokki, vaid kulgeb läbi mitme andmebloki),
- Võib toimuda ridade migreerimine (muutmisoperatsioon katab kogu vaadeldava andmebloki vaba ruumi, mille tulemusena kogu muudetav rida viiakse üle uude andmeblokki, jättes esialgsesse blokki vaid viida sellele reale).

- Sobib eelmiste punktide alusel staatilisi andmeid (muutmised harvad, muutused ei suurenda rea pikkust) sisaldavate tabelite jaoks.

2)Suurem väärtus (üle 20):
- Kulutab rohkem kettaruumi (vähem kirjeid ühes blokis),
- Optimeerib andmebloki poole pöörduvate operatsioonide töökiirust,
- Sobib dünaamilisi andmeid (eriti siis, kui muutmisoperatsioonid kipuvad suurendama ridade pikkust (muutmisoperatsiooniga antakse väärtus selle ajani määramata olnud mittekohustuslikku tüüpi veerule)) sisaldavate tabelite jaoks.

Tüüpiliselt antakse rakendustes antud paramrrtrile suurem väärtus valitud tabelites, milles on palju mittekohustuslikke veerge, millised võidakse täita hiljem muutmiste käigus (mitte esialgsel rea lisamisel).

2. PCUSED (vaikimisi 40 %) määrab ära, mitu % andmebloki kogumahust on veel täidetud, kui blokk kuulutatakse uuesti vaba olevaks. Kui korra täidetud olnud andmebloki täitumusaste langeb alla parameetris PCTUSED sisalduva väärtuse, siis andmeblokk kuulutatakse vabaks (uuesti täis olevaks kuulutatakse blokk siis, kui vaba ruum blokis saab väiksemaks kui parameetri PCTFREE väärtus.

Parameetrile PCUSED optimaalse väärtuse leidmisel tuleks lähtuda järgnevast:
1) Suurem väärtus:
   - Kasutab ökonoomsemalt kettaruumi,
   - Vähendab töökiirust (lisamised, muutmised), kuna:
     - On suurem tõenäosus fragmenteerumiseks (ridade kustutamisel vabanev ruum ei pruugi paikneda bloki sees järjest),

- Andmeblokki lülitatakse tihedamini ümbervabade blokkide nimistusse ja sealt ära

- Sobib staatilisi andmeid (eriti juhul, kui ridade kustutamisi on vähe) sisaldavate tabelite jaoks.

2) Väiksem väärtus:
   - Nõuab rohkem kettaruumi,
   - Optimeerib lisamis-, muutmisoperatsioonide töökiirust bloki ridade peal,
   - Sobib dünaamilisi andmeid (palju ridade kustutamisi) sisaldavate tabelite jaoks.

3. Parameetrid MINTRANS ja MAXTRANS on seotud sellega, kui mitu samaaegset tegevust võib korraga antud andmebloki andmete poole pöörduda.

Töökiirust silmas pidades on siin oht valida liiga väike parameetri MAXTRANS väärtus, mis sunniks suurema arvu samaaegsete pöördumiste korral osasid tegevusi teiste taga ootama. Suurem väärtus aga vähendab andmetele jäetavat ruumi andmebloki sees ning suurendab bloki päise osa (transaktsioonide jaoks reserveeritud osa paikneb andmebloki päises).

4. PCINCREASE näitab, mitu % järgnevana eraldatud ekstent on suurem eelmisest.

Kasutades siin suuremaid väärtusi, on võimalik võita töökiiruses, kuna suurem ekstent garanteerib suurema hulga andmete paiknemise järjestikustes andmeblokkides kettal (sellise ekstendi eraldamiseks on aga vaja leida kettalt piisava suurusega järjestikku paiknev vaba ruum).

Kõige suuremat efekti annab suurema väärtuse andmine parameetrile PCTINCREASE just väga suurte andmetabelite puhul. Väga suurte andmetabelite puhul tulekski kinni pidada loogikast: eraldada vähem ekstente, kuid teha ekstendid suuremad.

5. Parameeter INITIAL (esimese andmebaasi objektile (eelkõige puudutab see andmetabeleid) eraldatud ekstendi suurus) on mõistlik valida tabeli suurushinnangu põhjal, see tähendab nii, et esialgne ekstent mahutaks kõik algselt tabelisse laetavad read. See garanteeriks nende ridade

paiknemise järjestikustes andmeblokkides (ning võimaldaks sellega optimeerida antud ridade poole pöörduvate operatsioonide töökiirust).

Taoline suurushinnang sisaldab tabeli keskmise rea pikkust.

Lisaks laadimisparameetrite häälestamisega on võimalik optimeerida andmete paigutamist ka veergude õige järjestamisega. Mõistlik on paigutada (defineerida juba enne käsufailide genereerimist veergude õige järjekord teadmusbaasis) mittekohustuslikud veerud tabeli lõppu, et:

- Nende pikkus oleks 0 juhul, kui nad väärtust ei sisalda,
- Suurendada tõenäosust, et muutmisoperatsioonid toimuvad ainult rea lõpus.

Koodi häälestamine

Puudutame mõningaid reegleid, mida silmas pidada SQL lausete häälestamisel optimaalse töökiiruse saavutamise huvides. SQL võimaldab sama funktsionaalsust saavutada erineva süntaksiga kirjutatud käskude korral. Mõningatel juhtudel võimaldavad õigesti (optimaalse süntaksiga) formuleeritud SQL laused saavutada töökiiruses väga olulise võidu. Seetõttu tuleks optimeerida SQL lausete süntaksit nii andmete disaini objektide (serveris hoitavad protseduurid ja funktsioonid, andmebaasi trigerid), kui ka rakenduse disaini objektide (ekraanivormid, aruanded) juures.
Sobivaima süntaksi saab kindlaks teha katseliselt - erinevate variantide töökiirust analüüsides (käivitusplaani analüüs EXPLAIN PLAN käsuga).

SQL lausete häälestamise puhul on tegemist Oracle'isse sisse ehitatud optimeerija (mehhanism, mille ülesandeks on SQL lausete täitmiseks kõige efektiivsema tee leidmine temale teada oleva informatsiooni põhjal) toetamise (garanteerida, et optimeerija saaks valida kasutatavate käivitusplaanide hulgast kõige efektiivsema) ja juhtimisega (sundida optimeerijat tegema tavalisest erinevat otsust info põhjal, mida projekteerija teab andmete kohta rohkem kui optimeerija).

Koodi (SQL lausete) häälestamise juures tuleks silmas pidada järgnevat:

1. Indeksil põhinevad otsingud on üldjuhul madalama maksumusega (suurema töökiirusega, suurema efektiivsusega) kui terve tabeli lugemine otsimiseks (välja arvatud juhul, kui kasutatava indekseeritud veeru selektiivsus on madal - palju korduvaid väärtusi indekseeritud veerus).

2. Eelnevast lähtudes peaks indekseeritud otsingul baseeruv käivitusplaan olema optimeerija jaoks valitav (kätte saadav). Viimase garanteerimiseks:
    - Peab vastav veerg olema indekseeritud
    - Peab käsus sisalduma konstruktsioon, mis teeb indeksi kasutamise võimalikuks
    - Peab olemas olema ajakohane statistika koodis kasutatavate andmebaasi objektide kohta. Statistika genereeritakse objektide peale rakendatava ANALIZE käsuga.Tegelikkusele mittevastava jaotusstatistika kasutamine võib sundida optimeerijat tegema vale otsust käivitusplaani valikul.
    - Või peab olema optimeerijale antud vihje (INDEX) indeksi kasutamiseks (sel juhul võib statistika ka puududa, ainuke nõue on see, et kasutatav veerg peab olema indekseeritud).
3. Tuleks ära kasutada optimeerija juhtimiseks (vihjetega) andmete kohta teada olevat infot, mis võiks viia optimeerija poolt valitavast parema käivitusplaani valimiseni. Näiteks halva selektiivsusega veeru puhul soovitada indeksi kasutamist terve tabeli lugemise asemel (milline on halva selektiivsuse korral optimeerija valik), kui otsingus kasutatava väärtuse esinemissagedus antud veerus on madal.
4. Kasutada süntaksit, milline lubaks kasutada võimalikult madala maksumusastmega käivitusteed (15 raskusastet, alustades rea leidmisest pseudoveeru ROWID abil, läbi ühte rida ja ridade vahemikku määravate indeksite kasutamise kuni kogu tabeli lugemiseni välja).
5. Keerukamate käskude puhul proovida läbi mitu alternatiivset süntaksit (kontrollida nende täitmise aegu, käivitusplaanide valikut) ning valida nende hulgast kõige efektiivsem.

<u>Näide koodi häälestamise rakendamisest</u>

Vaatame programmilõiku, mis leiab ette antud ajavahemikul arvetele laekunud summad, vastavate arvete summad, samad summad teenuse liikide kaupa ning vastavate teenuste kogused

```
SELECT   RAJA.SUMMA SUMMA
         ARVE.SUMMA SUMMA1
         ,teenuse_liigi_summa(to_char(ARVE.ARVE_NUMBER),'V','2')
          TEENUSE_SUMMA
         ,teenuse_liigi_summa(to_char(ARVE.ARVE_NUMBER),'V','2')/
          '2' TEENUSE_KOGUS
FROM   RAHA_JAOTAMISED  RAJA,  ARVED  ARVE
```

```
WHERE  ARVE.ARVE_NUMBER = RAJA.ARVE_ARVE_NUMBER
AND  RAJA.KASUTAMISE_KP  IS  NOT  NULL
AND    TRUNC(RAJA.KASUTAMISE_KP,'DD')>=
TRUNC(NVL(to_date(:ALGUSKP,'DD.MM.YYYY'),SYSDATE),'DD')
AND  TRUNC(RAJA.KASUTAMISE_KP,'DD')<=
TRUNC(NVL(to_date(:LOPUKP,'DD.MM.YYYY'),SYSDATE),'DD')
AND  RAJA.ARVE_ARVE_NUMBER IS NOT NULL
AND NOT NVL(RAJA.MÄRKUS,:DUMMY) LIKE '%VIIVIS%';
```

SUMMA	SUMMA1	TEENUSE_SUMMA	TEENUSE_KOGUS
600000	1105030.2	49308	24654
400000	1019145.8	0	0
5000	5734.8	0	0
734.8	5734.8	0	0

Kulutatav protsessori aeg : 2 min 10 sek.

Vaadeldavas lõigus sisaldub serveri funktsiooni teenuse_liigi_summa väljakutse, milline omakorda pöördub korduvalt (tsüklis) teise serveri funktsiooni leia_tariif poole:

```
CREATE  OR  REPLACE  FUNCTION leia_tariif(
  Liigi_kood  IN  NUMBER,
  grupi_nr IN NUMBER,
  loppkp IN DATE DEFAULT SYSDATE)
RETURN NUMBER IS
BEGIN
 /*Leiab viimase (siiamaani kehtestatute hulgast) seni kehtiva tariifi*/
 /*ette antud teenuse liigi ja tariifigrupi kohta, milline on kehtestatud*/
 /*enne ette antud võrdluskuupäeva või sellel kuupäeval*/
DECLARE
  summa tariifid.tariif%type;
BEGIN
   SELECT tariif INTO summa FROM tariifid
   WHERE tariifid.tagr_teli_kood = liigi_kood
   AND tariifid.tagr_grupp = grupi_nr
   AND tariifid.kuupäev <= loppkp
   AND ROWNUM = 1
   Order BY kuupäev desc;
```

```
   RETURN summa;
 END;
END leia_tariif;
/
```

Funktsioonis sisalduv WHERE tingimus sisaldab tegelikult vastava tabeli (TARIIFID) unikaalsesse võtmesse kuuluvaid veerge (ka veergude järjestus on sobiv indeksi kasutamiseks). Seetõttu sai järgnevas sunnitud optimeerijat kasutama tabeli TARIIFID primaarse võtme indeksit (indekseeritud otsing vahemiku alusel, kuna kuupäeva võrdlemise tingimus lubab ainult vahemiku alusel otsida). Seega sai funktsiooni sisse viidud järgmine muutus:

```
SELECT /*+INDEX_DESC (tariifid tari_pk)*/
   tariif INTO summa FROM tariifid   WHERE …
```

Indeksi peale sundimise tulemusena otsingule tabelil TARIIFID oli kogu vaadeldava programmilõigu täitmisele kulunud aeg 1 min 17,7 sek.

Teema: Käsufailide genereerimine

Kui andmete kirjelduste lihvimine on lõpetatud, siis järgmiseks sammuks andmete disaini käigus on käsufailide genereerimine. Käsufailide genereerimine on vajalik reaalse andmebaasi ja selles sisalduvate objektide (tabelid, indeksid, kitsendused, kasutajagrupid, kasutajad, ..) loomiseks. (Reaalne andmebaas ja selle objektid luuakse genereeritud käsufailide käivitamise tagajärjel.) Käsufailide genereerimine toimub automaatselt selleks ette nähtud utiliidi (Generate DDL (Data Definition Language) Utility) käivitamise teel. Käsufailide genereerimise aluseks on andmebaasi enese, andmebaasi failide, andmebaasi objektide kirjeldused teadmusbaasis.

Käsufailide käivitamine

Käsufailide käivitamine loob füüsilise andmebaasi koos vajalike failidega, milledeks on:

- Parameetrite fail,
- Andmebaasi juhtfailid,
- Andmefailid
- Logfailid

ning objektid selles andmebaasis.

Sageli on kasulik käsufailide alusel esialgu enne lõplikku produktsioonilise andmebaasi loomist tekitada testandmebaas (prototüüplähenemine),kus:

- On vähendatud andmefailide jaoks ette nähtud parameetreid (failide suurusi),
- On vähendatud mõnede suuremate andmetabelite jaoks ette nähtud mahtusid,
- Muus osas säilitatakse kirjeldustes ette nähtud andmebaasi struktuur, et oleks võimalik esialgseks testimiseks sisse tuua testandmete komplekt andmebaasi rakendustest.

Taoline lähenemine jätab ülemineku etapi jaoks lahtiseks mitu võimalust:

1) Luua tegelik andmebaas uuesti juba eelnevalt olemas olevate käsufailide käivitamise teel (pärast seda, kui andmebaasi vajadustele vastav riistvara on olemas),
2) Täiendada (Upgrade) testbaasi serveri riistvara nii, et see vastab tegelikele nõudmistele ning suurendada andmefailide suurusi (ja andmetabelite mahtusid) jooksvalt Oracle'i vahenditega. (Tegelikult on osade parameetrite dünaamiline muutmine problemaatiline, efekt võib andmebaasi häälestatuse seisukohalt olla oodatule vastupidine.)

Taolise lähenemise puhul on võimalik:

- Vigade leidmisel, parandamise või täiendamise vajaduse tekkimisel tagasi pöörduda hilisematest faasidest (testimise käigus ilmnenud puudused jne.) disaini etapi objektide juurde ja viia sisse vastavad parandused teadmusbaasis ning seejärel genereerida tegelik produktsiooniline andmebaas kõiki parandusi sisaldavate käsufailide alusel.
- Lihtsalt läbi proovida erinevad võimalikud lahendusvariandid (erinevad süntaksid).

Lisa. Oracle'i struktuuridest

1. Oracle'I failid :

1.1. Andmefailid

1.2. Logfailid. Neisse failidesse salvestatakse kõik andmebaasis tehtud muutused. Nad on vajalikud andmebaasi taastamise juures (kuna nad sisaldavad infot kõigi andmebaasi sisse viidud muutuste kohta, siis toimub andmebaasi taastamine neis failides sisalduva informatsiooni põhjal).

1.3. Juhtfail:

- Kirjeldab andmebaasi struktuuri (andme- ja logfailid),
- Sisaldab taastamiseks vajalikku sünkroniseerimise infot (garanteerib et kõik failid taastatakse sama ajahetke seisuga, see tähendab, et taastatud failides sisalduv info on kooskõlaline ja täielik). See sünkroniseerimise info on sisuliselt logfaili number, mis kirjutatakse juhtfaili logfaili täitumisel (uue logfaili kasutusele võtmisel). Sama number kirjutatakse samal ajal ka kõigi töös olevate andmefailide päistesse.
- Hoiab andmebaasi nime,
- Salvestab muutused andmebaasi struktuuris (andmefaili lisamine, kustutamine, logfaili lisamine, kustutamine, …)

1.4. Parameetrite fail:

- Määrab ära kasutatava juhtfaili nime ja asukoha (raja juhtfailini),
- Paneb paika andmebaasi iseloomustavate parameetrite väärtused (erinevate mälustruktuuride kasutamine jpm.).

2. Oracle'i andmestruktuurid:

2.1. Blokk - füüsiline andmefaili blokk (kettal).

2.2. Ekstent - järjestikuste (kettal järjest paiknevate, nende aadressid tulevad järjest) andmeblokkide kogum.

2.3. Segment - 1 või mitu ekstenti, millised sisaldavad vaadeldava objekti (näiteks mingi tabel andmebaasis) kõiki andmeid antud andmeruumi raames. Mõnes muus andmeruumis paiknevad sama objekti andmed kuuluvad teise segmenti.

2.4. Andmeruum - loogiline (mitte füüsiline) struktuur andmete kokku koondamiseks (hoidmiseks). Füüsiliselt võib andmeruum paikneda kettal ühes või mitmes andmefailis.

2.5. Andmefail - andmete paiknemine füüsilisel kettal. Andmefail saab kuuluda 1 ja ainult 1 andmeruumi koosseisu (selle andmeruumi andmete hoidmiseks füüsilisel kettaseadmel).