Loeng 4. Disaini täpsustamine (Oracle CASE metoodika põhjal)

Taust

Disaini täpsustamise eesmärgiks on andmebaasi objektide kirjelduste täiendamine selliselt, et kogu struktuur võimalikult täpselt vastaks analüüsi etapil spetsifitseeritud nõudmistele.

Disaini täpsustamine hõlmab:

- Indeksite ülevaatamist ja vajadusel uute indeksite lisamist,
- Numbrijada generaatorite (sequence) loomist ja veergude sidumist nendega.
- Andmete denormaliseerimise vajaduse läbikaalumist.
- Olemasolevate kontrollimehhanismide ülevaatamist.
- Keerukama kontrolliva loogika lisamist (kontrollkitsendused, andmebaasi trigerid).

Loengu eesmärgid

1. Anda ülevaade disaini täpsustamise faasi sisust, tegevustest, objektidest ning eduka läbiviimise kriteeriumitest.
2. Käsitleda igat disaini täpsustavat põhitegevust eraldi ning seoses teiste disaini tegevustega.

Loengu plaan

1. Indekseerimine
2. Numbrijada generaatorite loomine
3. Denormaliseerimine
4. Kitsenduste (piirangute) täpsustamine ja lisamine
5. Andmebaasi trigerid
6. Abitabelite lisamine

Indekseerimine

Indekseerimine on üks võtetest, mille abil on võimalik oluliselt tõsta andmetega opereerimise kiirust. Indekseerimise pealt maksimaalse efekti saavutamiseks tuleb indekseeritavaid veerge valida teatavate kriteeriumite alusel:

- Tabelite veergude indekseerimise vajadust kaaluda eelkõige siis, kui:
  - Tabel on suur ning tüüpilised päringud toovad välja vaid väikese osa tabeli ridade koguarvust (tüüpiliselt tõstavad indeksid oluliselt päringutele vastamise kiirust juhul, kui päringud toovad välja 2 – 4 % tabeli ridade koguarvust).
  - Väärtused tabeli indekseeritavates veergudes on suhteliselt unikaalsed (vähe on korduvaid väärtusi).
  - Tabeli indekseeritavates veergudes on küll palju korduvaid väärtusi, kuid tüüpilised päringud kasutavad (WHERE tingimuses) just neid väärtusi, mille esinemissagedus on teiste väärtuste omast väiksem.
  - Veerg sisaldab palju määramatusi (NULL), kuid päringus küsitakse tavaliselt ridu, milles antud veerus on väärtus olemas (WHERE veerg IS NOT NULL).

- Indekseerida kõik primaarsetesse ja unikaalsetesse võtmetesse kuuluvad veerud. Need veerud on seotud Oracle'I serveri poolt loodavate kitsendustega ning tegelikult loob Oracle CASE metoodika kasutamise korral taoliste veergude jaoks indeksid automaatselt (esmase andmete disaini käigus). Selliste veergude indekseerimise põhjus on toodud eelmises punktis – need veerud on kõige selektiivsemad, s.t. neis korduvad väärtused puuduvad.

- Indekseerida veerud, mida kasutatakse sageli päringutes (SELECT käsu WHERE tingimuses) või tabelite sidumisel (tüüpiliselt samuti päringutega seoses).

- Arvestada, et indekseerimine võtab alla lisamiste, muutmiste ja kustutamiste kiirust (kuna koos põhitegevusega tuleb ülal hoida ka vastavaid indekseid), kuid tõstab päringutele vastamise kiirust.
  Seepärast on soovitav:
    - Mõelda veergude indekseerimisele eelkõige tabelites, kus muutmised on küllalt harvad võrreldes päringutega,

- Mitte indekseerida tabelites veerge, mida objekti esinemise eluea jooksul tuleb sageli muuta.

- Indeksite hoidmine nõuab lisaruumi kõvakettal.

- Kasutada mitmekomponendilisi (mitmest veerust koosnevaid) indekseid, kui:
  - Veergude kombinatsioon annab oluliselt parema selektiivsuse (või koguni garanteerib unikaalse väärtuse) kui iga veerg eraldi.
  - Tüüpilised päringud toovad välja ühe ja sama veergude komplekti vaadeldavast tabelist. Sellise veergude komplekti koondamisel indeksisse saavutatakse, et päringud saavad vastused kätte otse indeksist ilma tabeli poole pöördumata.

Mitmekomponendiliste indeksite kasutamisel on oluline komponendiks olevate veergude järjestus. Selle valimisel tuleks lähtuda järgmistest põhimõtetest (toodud tähtsuse kahanemise järjekorras):

1) Paigutada veerud indeksisse vastavalt nende tabelite ühendamisel (WHERE tingimus) kasutamise sagedusele.
2) Võrdse kasutamissageduse korral paigutada ettepoole veerud, mille selektiivsus on suurem (vähem korduvaid väärtusi).
3) Kui andmed on tabelis füüsiliselt järjestatud mingite veergude põhjal, siis paigutada ettepoole need veerud.

Numbrijada generaatorite kasutamine

Numbrijada generaator on mehhanism, mida Oracle kasutab järjestatud numbrite genereerimiseks. Generaator peab tagama selle, et iga järgneva pöördumisega väljastatav number oleks ette antud arvu võrra suurem eelmisest.

Sarnaselt indekseerimisele on ka numbrijada generaatorite kasutamisega võimalik kaasa aidata optimaalse jõudluse saavutamisele olemasoleva andmete struktuuri juures.

```
SQL>  select sequence_name NUMBRIJADA_NIMI, increment_by
2         KASVUKIIRUS, cycle_flag TSÜKLILINE, last_number
3         JÄRGMINE_NR from user_sequences
4         where sequence_name like 'TÖÖK%';

NUMBRIJADA_NIMI  KASVUKIIRUS  T  JÄRGMINE_NR
---------------------------- ---------- - ----------
TÖÖKÄSU_NR                    1          N          69
```

Selliselt genereeritud numbrite tüüpiliseks kasutuskohaks on väärtuste andmine primaarsetele ja unikaalsetele võtmetele. Oracle'i andmebaasis on võimalik genereerida 2 erinevat tüüpi numbrijadasid:

1. nn. Oracle'i numbrijadasid (Oracle Sequence), mille ülalhoidmise eest hoolitseb Oracle'I server ise.
2. nn. Koodiga juhitavaid jadasid (Code Control Sequence), mille ülalhoidmise loogika tuleb kirjutada projekteerijal.

Oracle'i numbrijada iseloomustus:

- tema poole võib korraga pöörduda mitu kasutajat
- tema kasutamisel võivad järjestatud numbritesse sisse jääda tühikud, kuna alustatud tegevuse tagasivõtmise korral mingi vea tõttu läheb number vahelt kaduma.
- Seepärast on niisugused jadad kõige paremini kasutatavad väärtuste andmisel veergudele, mis on vajalikud eelkõige andmebaasi enese sisemise loogika jaoks. Need on veerud, mida:
  - Andmebaas kasutab tabelite sidumisel,
  - Mille olemasolust kasutaja ei pea olema teadlik
  - Mille väärtused ei ole tuletatud mingi range reegli alusel
  - Väärtused ei pea tulema järjest (võivad tühikud vahele jääda).

- Ta luuakse koos teiste andmebaasi objektidega genereeritud käsufailide käivitamisel,
- Numbrite genereerimise, numbrijada ülalpidamise eest hoolitseb Oracle ise,
- Pöördumine jada poole toimub läbi Oracle'I pseudoveeru NEXTVAL.

```
NEXTVAL
----------
        69
```

Koodiga juhitava jada omadused:

- Tema poole saab korraga pöörduda vaid 1 kasutaja
- Kasutaja pöördumisel jada lukustatakse ning järgmine kasutaja saab sama jada poole pöörduda alles pärast seda, kui eelmise kasutaja tegevus on täielikult lõpetatud (salvestatud või tagasi võetud).
- Seda tüüpi jada generaator garanteerib tühikute puudumise genereeritud järjestatud numbrites, kuna tegevuse tagasivõtmisel annulleeritakse ka numbrite jada kasvatamine ning sama numbrit saab kasutada järgmise tegevuse poolt uuesti.
- Eelmistest punktidest lähtuvalt on seda tüüpi numbrijadasid soovitav kasutada väärtuste genereerimiseks veergudele, kus on oluline numbrite järjepidevus,
- Numbrijada ülalpidamise loogika tuleb kirjutada projekteerijal,
- Numbrijada andmeid hoitakse 1 reana koodi juhtimise tabelis (tabel CG_CODE_CONTROLS, antud juhul on tegemist ühise tabeliga kõigi rakenduste peale, kuid olenevalt paika pandud generaatori määratlusest on võimalik nõuda ka eraldi tabelite genereerimist iga rakenduse jaoks).

```
SQL>  select cc_domain, cc_next_value from cg_code_controls where
2         cc_domain like 'ARVE%';

CC_DOMAIN                CC_NEXT_VALUE
------------------------ --------------------------------
ARVENUMBER                     97000021

REM
REM       Teostab tegevused uue arve numbriga.
REM
PROMPT  Creating Prcedure  uus_arve
CREATE  OR  REPLACE  PROCEDURE  uus_arve (
     arve_nr  IN  OUT  NUMBER  ,
     eelm_arve  IN  OUT  NUMBER  ,
     lepingu_nr  IN  NUMBER  )
IS
BEGIN
    /*1.  Genereerib koodikontrolli tabeli alusel unikaalse */
    /*      arve numbri  lisatavale arvele*/
    /*2.  Genereerib viida eelmisele arvele */
```

```
/*3.  Paneb  lisatava  arve  nr-i  aktiivse lepingu viimaseks */
/*      arve nr-iks*/
DECLARE
    aasta  VARCHAR2 (2);
BEGIN
  /* Leiab jooksva aastanumbri  2  viimast kohta*/
  SELECT  TO_CHAR (sysdate, 'YY') INTO aasta
  FROM  dual;

  /*Leiab potentsiaalse järgmise arve numbri*/
  SELECT  cc_next_value  INTO arve_nr
  FROM  cg_code_controls
  WHERE cc_domain = 'ARVENUMBER'  FOR  UPDATE;

  /*Kui  vahepeal  on aasta muutunud, siis genereeritakse */
  /*uus nr, mille 2 esimest kohta vastavad uue aasta */
  /* 2-le viimasele nr-le ja lõpp on 000001  */
  IF aasta  <> SUBSTR(TO_CHAR(arve_nr), 1, 2) THEN
     UPDATE cg_code_controls  SET
       cc_next_value =
      TO_NUMBER(SUBSTR(TO_CHAR(cc_next_value+
      1000000), 1, 2)  ||  '000001')
     WHERE cc_domain = 'ARVENUMBER';

     SELECT  cc_next_value INTO arve_nr
     WHERE  cc_domain = 'ARVENUMBER' FOR UPDATE;
  END IF;

  /*Suurendab nr 1 võrra, et järgmine pöördumine saaks kätte */
  /* 1 võrra suurema unikaalse nr */
  UPDATE cg_code_controls SET cc_next_value =
  cc_next_value+1
  WHERE cc_domain = 'ARVENUMBER';

  /* Genereerib viida eelmisele arvele */
  SELECT viimase_arve_nr  INTO eelm_arve
  FROM lepingud  WHERE  lep_number = lepingu_nr;

  /*Paneb lisatava arve nr-i aktiivse lepingu viimaseks arve */
  /* nr-iks  */
  UPDATE  lepingud  SET  viimase_arve_nr = arve_nr
  WHERE  lep_number = lepingu_nr;
END;
```

```
END uus_arve;


REM
REM      Genereerib  unikaalsed  arve numbrid
REM
PROMPT
PROMPT  Creating TriggerGEN_ARVE_NR
CREATE  OR  REPLACE  TRIGGER  gen_arve_nr
BEFORE  INSERT
ON  ARVED
FOR  EACH  ROW
DECLARE
BEGIN
    /*Kutsub  välja mooduli, milline :*/
    /*1.  Genereerib koodikontrolli tabeli alusel unikaalse arve numbri */
    /* lisatavale arvele*/
    /*2.  Genereerib viida eelmisele arvele */
    /*3.   Paneb lisatava arve nr-i aktiivse lepingu viimaseks arve nr-iks*/
    BEGIN
        Uus_arve (:NEW.arve_number, :NEW.arve_arve_number,
        :NEW.lep_lep_number);
    END;
END;
```

See oli A) Väljavõte koodi juhtimise tabelist ning B) vastavat koodi juhtimise tabeli rida ülal pidav andmebaasi triger koos trigeri poolt välja kutsutava protseduuriga, mis sisaldab tegelikult rea ülalpidamisega seotud koodi (maksimaalne osa koodist on viidud andmebaasi trigerist ära eraldi seisvasse moodulisse).

- Numbrijada luuakse (rida koodi juhtimise tabelis) spetsiaalselt selleks ette nähtud utiliidiga (Update Code Control Tables).

Rakenduse loomise disaini etapil on soovitav unikaalsete ja primaarsete võtmete väärtuste genereerimiseks maksimaalselt ära kasutada numbrijada generaatorite võimalusi, et:

- Mitte kasvatada süsteemi keerukust (lisakoodi kirjutamine),
- Mitte kaotada töökiiruses. Numbrijada generaatoritega samaväärsete mehhanismide kodeerimine nõuab suurte tabelite lukustamist teiste kasutajate jaoks, et teha kindlaks senine suurim väärtus primaarse (või unikaalse) võtme veerus. See sunnib kasutajaprotsesse ootama

üksteise taga ning tõmbab alla rakenduste töökiirust vaadeldava tabeli peal
(ka koodiga juhitavate jadade kasutamine ei sunni suurt põhitabelit lukustama. Uus väärtus tehakse kindlaks väikese, üksikuid ridu sisaldava abitabeli (koodi juhtimise tabeli) kasutamisel).

Seejuures tuleks lähtuda järgmistest põhimõtetest:

- Kui võtme (unikaalne või primaarne) väärtus ei oma kasutaja jaoks olulist tähendust, siduda võtmeveeruga Oracle'i numbrijada (võtmeveeru väärtused on olulised vaid süsteemi enese jaoks unikaalsuse tagamisel, tabelite sidumisel).
- Kui võtme väärtused on kindla algoritmi alusel arvutatavad (tuletatavad) ning samal ajal peavad olema rangelt järjestatud ilma vahepealsete tühikuteta, siis siduda võtmeveeruga koodi juhtimise jada.
- Kui võtme väärtus omab tähendust süsteemi potentsiaalse kasutaja jaoks või laiemas mõttes, jätta võtmeveeru väärtuse sisestamine kasutaja jaoks lahtiseks ning võimalusel lisada rakendusse kood vaikimise väärtuste (, mida kasutaja saaks vajadusel muuta) genereerimiseks, lubatud väärtuste kontrollimiseks.

Denormaliseerimine

Analüüsi etapi lõpuks peaksid kõik andmed olema viidud kolmandale normaalkujule (iga olemi iga atribuudi väärtus sõltub ainult vastava olemi unikaalsest identifikaatorist ja ei millestki muust). Denormaliseerimise eesmärk on aga taolisse andmemudelisse sõltuvate andmete sissetoomine. Taoline tegevus omab mõtet juhul, kui see annab arvestatava võidu töökiirust ja jõudlust silmas pidades.
Denormaliseerimise võimalust tuleks kaaluda vaid tabelite puhul:

- , mida kasutatakse sageli päringutes
- , milles andmete muutmised toimuvad harva.

Tuletatud veergude sissetoomise võimalust tuleks kaaluda näiteks arve ridade tabelis, kus iga rea summa moodustub sõltuvates tabelites sisalduvate andmete alusel. On võimalik, et võit töökiiruses (aruannete koostamise juures) kaalub üles trigerite lisamisega (vastava summa arvutamine) juurde tuleva keerukuse ning arvete moodustamise juures summa arvutamisele kulutatava aja.