

Andmebaasi füüsiline disain, ehitamine, andmesiire, varundamine

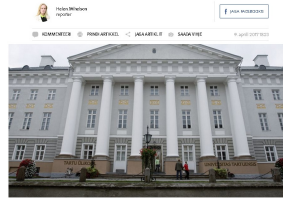
Teema 4

Andmebaasid II 2017

© Erki Eessaar

Jõudlusprobleemide näide

12 kliki jaoks kolm tundi: teadlased on
hädas üliaeglaste infosüsteemiga



Põhjendamatu aja- ja närvikulu.
Aega ei saa kuskilt juurde osta.

- ♦ Eesti Teadusinfosüsteem
<https://www.etis.ee/>
- ♦ Eesti teadlaste ja
õppejõudude kohustuslik
töövahend (nt CVD,
publikatsioonid,
uurimisraha taotlused,
aruanded ning nende
retsensioonid)
- ♦ Töödel on **tähtjad!**

10.01.2018

Teema 4

2

Andmebaasid II 2017

© Erki Eessaar

Andmebaasioperatsioonide töökiiruse parandamine

- ♦ Sellele tuleb mõelda **kõigis** andmebaasi
arendamise etappides/faasides.
 - **Ei ole hõbekuuli!!**
- ♦ Üldiselt – mida **varasemas** etapis/faasis
kasutuselevõetud meede, seda parema
tulemuse see annab.
- ♦ Ennetamine on parem kui probleemiga
võitlemine.

10.01.2018

Teema 4

3

Andmebaasid II 2017

© Erki Eessaar

Andmebaasioperatsioonide töökiiruse parandamine (2)

- ♦ Andmebaasis toimuvate operatsioonide
töökiirus on ainult üks (kuid kindlasti
oluline) komponent, mis mõjutab **süsteemi
üldist töökiirust**.
- ♦ **Süsteem (sild, tarkvarasüsteem ...)** on nii
tugev kui on selle **kõige nõrgem komponent**.
 - Vajalik on tasakaalustatud disain, mille korral
kavandatava süsteemi elemendid jõuavad
taluvuspiirile üheaegselt.

10.01.2018

Teema 4

4

Andmebaasid II 2017

© Erki Eessaar

Optimeerimise meetodid analüüsi käigus

- ♦ Võimalikult täpne **nõuete** kogumine.
 - "Far and away the most money I ever made was by
repeatedly convincing people to drop features and
even a couple of applications entirely. There's way too
much software that influences people who don't have the
same requirement as whoever pushes the software or no
requirement at all. Customers were delighted to discover
they didn't have the requirements in the first place. The
big consultancies didn't like me for that because they had
to start doing real work at those sites for the first time, at
least in the small isolated aspects I could influence."
(Eric J. Schwarzenbach ühes andmebaaside teemalises foorumis)

10.01.2018

Teema 4

5

Andmebaasid II 2017

© Erki Eessaar

Analüüsi vead on tõsised vead

- ♦ Nõuete kogumise (analüüsi) faasis tehtud
vigade parandamiseks kuluv pingutus kasvab
iga järgmise süsteemiarenduse faasiga kolm
korda.
- ♦ Disaini faasis on neid **3**, realiseerimise faasis
9 ja testimise faasis **27** korda kulukam
parandada kui nõuete kogumise faasis.

Jaakkola, H., Henno, J., Welzer-Druzovec, T., Thalheim, B., Mäkelä, J., 2016.
Why Information Systems Modelling Is Difficult. In *SQAMIA*. pp. 29–39.

10.01.2018

Teema 4

6

Optimeerimise meetodid andmebaasi **disaini** käigus

- ♦ Optimeerida **andmestruktuuride** disaini (denormaliseerimine – ETTEVAATUST!).
- ♦ Kasutada maksimaalselt andmebaasisüsteemi **sisseehitatud funktsionaalsusi** – nt Oracles auditeerimine, andmemuudatuste ajaloo säilitamine (Workspace Manager).

Optimeerimise meetodid andmebaasi **disaini** käigus (2)

- ♦ Kasutada maksimaalselt ära **andmebaasisüsteemi** poolt pakutavaid **võimalusi**:
 - andmebaasi programmeerimiseks (nt arvutada generaatorid, andmebaasiserveris talletatud rutiinid),
 - operatsioonide töökiiruse parandamiseks (nt tabelite sektsioonideks jagamine ja klastrisse koondamine).

Optimeerimise meetodid andmebaasi **disaini** käigus (3)

- ♦ Projekteerida **indeksite** kasutus.
- ♦ Projekteerida **hetktõmmiste e materialiseeritud vaadete** kasutus.
- ♦ Viia **võrgu koormus** minimaalseks (andmebaasiserveris talletatud rutiinid).
- ♦ Protseduuride **algoritmide** häälestamine.
 - J. Bentley, *Programming Pearls*, Second Edition, Addison-Wesley, 2000.

Optimeerimise meetodid andmebaasi **disaini** käigus (4)

- ♦ Planeerida andmebaasiobjektide **paigutamist** erinevatele serveritele ja ketastele.
 - Andmed ühel serveril, ühel või mitmel kettal, võivad olla hargsalvestatud **sõltumatute ketaste liiasmassiivile (RAID)**
 - Andmed samas geograafilises asukohas **serverarvutite klastris**
 - *Shared Nothing Partitioning*
 - Andmed erinevates geograafilistes asukohtades, kasutajatele lähedal (**hajus andmebaas**)



Andmete paigutamisest

- ♦ Eelneva korral on erinevad andmeobjektid erinevatel ketastel/serveritel (*sharding*).
- ♦ Parandab **jõudlust**, sest võimalik andmete paralleelne lugemine/kirjutamine erinevate kettaseadmete/serverite poolt.
- ♦ **Töökindlust** saab parandada luues lisaks andmetest koopiaid erinevatel ketastel/serveritel.



Andmete paigutamisest (2)

- ♦ Juhul kui andmebaasisüsteem võimaldab määrata, millistes failides/ketastel peaksid paiknema baastabelite/indeksite andmed.
 - PostgreSQLis ja Oracles selleks tabeliruumid.
 - Võib teha nii, et **baastabelid** on aeglasemal, kuid suurema mahuga **kõvakettal** ja **indeksid** väiksema mahuga, kuid kiiremal **pooljuhtkettal**.

Optimeerimise meetodid andmebaasi **disaini** käigus (4)

- ♦ Projekteerida tabelite **seksioonideks jagamine**.
- ♦ Planeerida andmebaasi **failide** sisemine organisatsioon (nt sorteerimine, pakkimine).
- ♦ Valida sobiv **plokkide** suurus andmefailides.
- ♦ Planeerida **vaba ruumi suurus** plokkides.
- ♦ Planeerida andmebaasisüsteemi **mälukasutust**.

10.01.2018

Teema 4

13

Optimeerimise meetodid andmebaasi **disaini** käigus (5)

- ♦ Planeerida **transaktsioonide** läbiviimist.
 - Transaktsioonid ei tohiks üksteise järel liialt kaua oodata.
- ♦ Planeerida tarkvara **paigutus**.
 - Andmebaasisüsteem ning rakenduste server eraldi füüsilistel arvutitel jätab mõlemale rohkem arvutiressurssi.

10.01.2018

Teema 4

14



Disaini protsess

- ♦ Disaini valikud **mõjutavad** üksteist – mõned rohkem mõned vähem
 - Näiteks mõne indeksi loomine võib kaotada vajaduse luua mingi hetktõmmis või vastupidi
- ♦ Suurendab disainiprotsessi **keerukust!**
 - Raske jõuda esimese korraga ideaalini
 - „Ideaali“ on liikuv sihtmärk, sest süsteemi kasutajate hulk, andmete hulk, kasutamise mustrid muutuvad

10.01.2018

Teema 4

15

Disaini protsess (2)

- ♦ Võimalikud disainilahendused sõltuvad kasutatavast **andmebaasisüsteemist**.
- ♦ Oluline teada põhimõtteid!
 - Ridade sorteerimine, ploki ja nende suurus, erinevat tüüpi indeksid, tabelite seksioonideks jagamine, hetktõmmised, ...

10.01.2018

Teema 4

16

Disaini protsess – soovitusi

- ♦ **Itereeri**
 - Timmi pidevalt olemasolevat disaini
- ♦ Alusta **indeksitest**
 - Üldiselt väga suur mõju operatsioonide töökiirusele
- ♦ Kasuta (**füüsilise**) **disaini nõustamise vahendeid**
 - Oracle SQL Access Advisor
 - MS SQL Server Database Tuning Advisor
 - DB2 Design Advisor

10.01.2018

Teema 4

17

Oracle SQL Advisor – näide

```
BEGIN
  DBMS_ADVISOR.quick_tune(
    advisor_name => DBMS_ADVISOR.SQLACCESS_ADVISOR,
    task_name    => 'emp_quick_tune',
    attr1        => 'SELECT e.* FROM emp e WHERE UPPER(e.ename) =
"SMITH"');
END;
```

10.01.2018

Teema 4

18

Oracle SQL Advisor – näide (2)

```
SET LONG 100000
SET PAGESIZE 50000
SELECT DBMS_ADVISOR.get_task_script('emp_quick_tune') AS script
FROM dual;
SET PAGESIZE 24
```

Tulemus:

SCRIPT

```
Rem SQL Access Advisor: Version 12.1.0.1.0 - Production
Rem
Rem Username: C##TUD1
Rem Task: emp_quick_tune
Rem Execution date:
Rem
```

```
CREATE INDEX "C##SCOTT"."EMP_IDX$$_06DD0000"
ON "C##SCOTT"."EMP"
(UPPER("ENAME"))
COMPUTE STATISTICS;
```

10.01.2018

Teema 4

19

Optimeerimise meetodid andmebaasi ehitamise käigus

- Optimeerida andmebaasi kasutatavat **programmikoodi** (nii SQL andmekäitluskeele lauseid, andmebaasiserveris talletatud rutiine kui ka rakenduse koodi).
 - SELECT * asemel küsige andmeid, mida tegelikult vaja, et süsteem ei oleks koormatud mittevajalike andmete otsimise ja üle võrgu edastamisega.
 - Pascal, F. 1988. SQL redundancy and DBMS performance. *Database Programming and Design*. Vol. 1, No. 12, pp. 22–28.

10.01.2018

Teema 4

20

SQLi keeleline liiasus

- Männil, M., 2014. *Mõnede SQL-andmebaasisüsteemide võimekusest SQLi keelelise liiasuse silumisel*. Magistritöö. TTÜ Informaatikainstituut. [WWW] <https://digi.lib.ttu.ee/i/?1952>
 - Pascali 1988. aasta katse kordus PostgreSQL ja Oracle andmebaasisüsteemide põhjal.

10.01.2018

Teema 4

21

SQLi keeleline liiasus (2)

- Ühte ja sama andmete leidmise ülesannet saab lahendada mitme erineva SELECT lausega, millel erinev süntaks.
- Erinevatel SELECT lausetel kahjuks erinev töökiirus, sest andmebaasisüsteem koostab neile erineva täitmisplaani.
 - Nii nagu seda näitas 1988. aasta uuring, esines seda ka 2014. aastal, kuigi väiksemas matus.

10.01.2018

Teema 4

22

SQLi keeleline liiasus (Männili järeldused) (3)

- Eksperimenti tulemused tõestasid, et **SQL keele liiasus** on endiselt **probleemide allikas**.
- Oli näha, et **optimeerijate efektiivsus** on aastate jooksul selgelt kasvanud.
- Ebatüüpilise** (vähem kasutatava) lause optimeerimisel näitas kommertssüsteem **Oracle** paremaid tulemusid kui PostgreSQL, mis tähendab, et viimase kasutamisel langeb suurem vastutus päringuid koostava inimese õlgadele.

10.01.2018

Teema 4

23

Optimeerimise meetodid andmebaasi hooldamise käigus

- Optimeerida operatsioonisüsteemi ja arvutivõrgu tööd.
- Vajadusel **täiendada/muuta** disaini ja ehitamise käigus rakendatud meetmeid (indeksid, andmete faili tasemel sorteerimine, andmebaasi mälukasutus, plokkide kasutus, andmebaasiobjektide paigutus jne).
 - Hoolduskulud **50–80%** tarkvarasüsteemidega seotud kuludest.

10.01.2018

Teema 4

24

Probleem

- ♦ Olemitüübil O on palju atribuute, mille väärtuseid soovitakse registreerida.
- ♦ Nende atribuutide hulk suureneb ajas, analüüsi käigus pole võimalik seda täpselt ennustada.
- ♦ Iga tüüpi O kuuluva olemitüübi korral on väärtustatud väike hulk nendest atribuutidest.
- ♦ Näited: kaupade omadused, haigla analüüs.

10.01.2018

Teema 4

25

Võimalikke lahendusi PostgreSQLis

- ♦ Lisada olemasolevasse tabelisse uusi veerge
 - PostgreSQL tabelis kuni 1600 veergu
 - Read lähevad liiga suureks, jõudlus kannatab
- ♦ "Universaalse disaini" tabelid
 - Variatsioon – üldiste tabelite *Atribuut*, *Atribuudi_väärtus* jne asemel luuakse olemitüüpide jaoks spetsiifilised tabelid
 - Näide: Tabelis *Kaup* kaupade üldandmed, *Kauba_omaduse_väärtus* spetsiifiliste atribuutide väärtused

10.01.2018

Teema 4

26

Võimalikke lahendusi PostgreSQLis (2)

- ♦ Tabelite pärimine
 - Iga atribuudi kohta alamtabel
 - Piirang välisvõtme kitsenduste jõustamisele ülatüüpi tabeli tasemel
 - Keerulised päringud üle erinevate atribuutide
- ♦ JSONB tüüpi veerg/veerud
 - Variatsioon – "põhilistele" atribuutidele vastavad eraldi veerud, varieeruvate/vähekasutatavate atribuutide väärtused JSONB tüüpi veerus

10.01.2018

Teema 4

27

Võimalikke lahendusi PostgreSQLis (3)

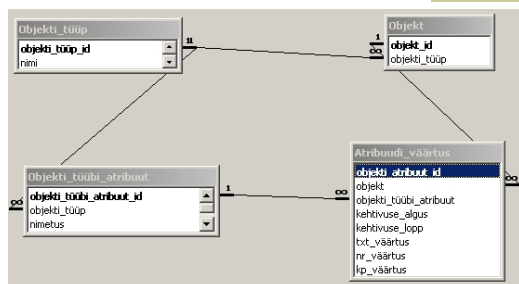
- ♦ Kuuendal normaalkujul tabelid
 - Igale atribuudile vastab eraldi tabel, mis on välisvõtme kaudu seotud olemitüübile vastava tabeliga
 - Kasutab näiteks ankurmodelleerimine: <http://www.anchor modeling.com/>

10.01.2018

Teema 4

28

Ebaotstarbeka struktuuriga SQL-andmebaas – "universaalne disain"



10.01.2018

Teema 4

29

Ebaotstarbeka struktuuriga SQL-andmebaas (2)

- ♦ Objektitüüp:
 - Töötaja
- ♦ Objektitüübi atribuut:
 - perenimi
- ♦ Objekt:
 - 1
- ♦ Atribuudi väärtus:
 - Mets

Sisuliselt ehitatakse üks andmemudel (*atribuut-väärtus paarid*) teise andmemudeli (*SQL andmemudel*) peale.

Kasutatakse näiteks **meditsiini infosüsteemides**, kus palju eritüübilisi mõõtmisi ja vaatluseid.

10.01.2018

Teema 4

30

Leia töötajate perenimed!

- ♦ SELECT Av.objekt,
Av.txt_väärtus AS perenimi
- ♦ FROM Objekti_tüüp AS Ot
INNER JOIN
(Objekti_tüübi_atribuut AS Ota
INNER JOIN Atribuudi_väärtus
AS Av ON
Ota.objekti_tüübi_atribuut_id =
Av.objekti_tüübi_atribuut) ON
Ot.objekti_tüüp_id =
Ota.objekti_tüüp
- ♦ WHERE
(Ota.nimetus='perenimi') AND
(Ot.nimi='Töötaja');
- ♦ SELECT perenimi
FROM Töötaja;

Soobik, M., 2007.
Andmebaasi disaini mõju
süsteemi töökiirusele:
traditsiooniline ja
universaalne disain.
Bakalaureusetöö. TTÜ
Informaatikainstituut.

Töökiiruse uuring

- ♦ Tellimuste andmebaas "traditsioonilise" ja "universaalse" disaini järgi.
 - Testandmed – nt tellimuste ridu ligi 600000
- ♦ 5 päringut ja 1 andmemuudatustega transaktsioon
- ♦ SELECT lausete täitmine "universaalse" andmebaasi põhjal keskmiselt **1.95** korda aeglasem võrreldes "traditsioonilise" andmebaasiga.
- ♦ Andmemuudatuste täitmine "universaalse" andmebaasi põhjal **1.58** korda aeglasem võrreldes "traditsioonilise" andmebaasiga.

Mõned universaalse disaini probleemid

- ♦ Päringud **keeruka** struktuuriga ja nende täitmine on aeglane. Andmebaasisüsteemi sisse ehitatud võimalused päringute töökiiruse parandamiseks ei ole mõeldud taolistel struktuuridel töötama.
- ♦ Andmemudel ei anna mingit informatsiooni **probleemvaldkonna** kohta, millele andmebaasi luuakse.

Mõned universaalse disaini probleemid (2)

- ♦ Andmetega seotud **kitsendusi** ei saa deklaratiivselt lisada.
- ♦ Kitsenduste kontrolliks loodud **trigerid** on keeruka struktuuriga ja nende haldamine on keeruline.
- ♦ **Õiguste** jagamine ei anna soovitud efekti.

Mõned universaalse disaini probleemid (3)

- ♦ Andmete kirjelduses (tabelid *Objekti_tüüp* ja *Atribuut*) muudatuste tegemiseks tuleks **lukustada** kogu andmebaas.
- ♦ Mistahes tabelis andmete kustumine/riknemine **halvaks kogu** süsteemi töö.
- ♦ Kui **kasutajaliideses** soovida igale atribuudile eraldi kindlat sisestusvälja, siis on selle liidese loomine keeruline.

Kokkuvõte

- ♦ Universaalne disain
 - SQL-andmebaasis üritatakse realiseerida *võtme-väärtuse* paaride andmemudelit
- ♦ JSONB tüüpi veerg/veerud
 - SQL-andmebaasis üritatakse realiseerida *dokumentide* andmemudelit
- ♦ Pärimise kaudu loodud tabelid
 - SQL-andmebaasis üritatakse realiseerida *objekt-orienteeritud* andmemudelit

Andmebaasid II 2017 © Erki Eessaar

Universaalne disain – kas saaks teisiti?

- ♦ Kaalu mõne teise andmemudeli kasutamist:
 - võti-väärtus paaride mudel,
 - RDF mudel.
- ♦ Olemas eraldi andmebaasisüsteemid selliste mudelite alusel esitatud andmetega töötamiseks.
 - <https://db-engines.com/en/ranking/key-value+store>
 - <https://db-engines.com/en/ranking/rdf+store>

10.01.2018 Teema 4 37

Andmebaasid II 2017 © Erki Eessaar

Ressursikirjeldusvorming (RDF)

Kas mõtteliselt või ilmutatult deklareerituna on objektiga seotud *andmetüüp*.

Subjekt	Predikaat	Objekt
töötaja 1	perenimi on	Mets

EAV mudeli alusel loodud andmebaasis olevaid andmeid saab RDF formaadis esitada järgneva vastenduse alusel


- objekt => subjekt
- atribuut => predikaat
- atribuudi väärtus => objekt

- ♦ RDF formaadis andmed esitatakse kolmikutena – *subjekt-predikaat-objekt*.
- ♦ Andmebaasikeel SPARQL sellistest andmetest otsingute tegemiseks, aga ka andmete muutmiseks.

10.01.2018 Teema 4 38

Andmebaasid II 2017 © Erki Eessaar

Veel üldised põhimõtted



- ♦ Mida vähem **lugemise/kirjutamise operatsioone** (muutmälust või kettalt) peab süsteem operatsiooni käigus tegema, seda kiiremini see operatsioon toimub.
 - Analoogia: raamatu lugemine indeksiga või ilma
- ♦ Andmete lugemine/kirjutamine **muutmällu** on palju kiirem kui andmete lugemine/kirjutamine kettale.

10.01.2018 Teema 4 39

Andmebaasid II 2017 © Erki Eessaar

Veel üldised põhimõtted (2)



- ♦ Iga meede, mida mingi operatsiooni kiirendamiseks kasutate, võib muuta mingite teiste operatsioonide läbiviimise **aeglasemaks**.
- ♦ Optimeerige **kõige olulisemate** operatsioonide jaoks – 80/20 reegel.
 - 20% operatsioonidest annab 80% süsteemi koormusest
- ♦ Oluliste operatsioonide hulk muutub ajas ja seega peab ka disain **ajas muutuma** (evolutsioon).



10.01.2018 Teema 4 40

Andmebaasid II 2017 © Erki Eessaar

Veel üldised põhimõtted (3)



- ♦ **Asünkroonne** töö aitab jõudlusele kaasa.
 - Sünkroonse töö näide
 - Töö A kutsub välja töö B ning ootab kuni B lõpetab. Kui B ebaõnnestub, siis ebaõnnestub ka A.
 - Asünkroonse töö näide
 - Töö A paneb B täitmise soovi järjekorda ning jätkab. B täidetakse kui selleks on vaba ressursi. A ei pea B järele ootama.
 - Näide: hetktõmmiste ja andmebaasi koopiate asünkroonne värskendamine.

10.01.2018 Teema 4 41

Andmebaasid II 2017 © Erki Eessaar

Veel üldised põhimõtted (4)

- ♦ Asünkroonne töö **vähendab** A ja B vahelist **sõltuvust**.
- ♦ Sünkroonne muudatuste ülekanne koopiate tegemisel võib ka olla halb selles mõttes, et vigased muudatused/riknenud failid kantakse koheselt koopiatesse, põhjustades koheselt nendes vigu või nende riknemise.
 - Andmete taastamine võtab rohkem aega.

10.01.2018 Teema 4 42

Veel üldised põhimõtted (5)

- ♦ Kasutaja eest **keerukuse peitmine** ja kasutaja töötamine kõrgemal abstraktsiooni tasemel (baastabelid vs. indeksid; vaated vs. baastabelid jne) aitab töökiiruse parandamise meetmeid paremini **hallata**.
- ♦ **Füüsiline andmete sõltumatus** on selle põhimõtte üks näide.

Füüsiline andmete sõltumatus

- ♦ Andmebaasi *sisemises skeemis* tehtud muudatus ei mõjuta seda kuidas paistab kasutajatele andmebaasi *kontseptuaalne skeem*.
- ♦ Selle tulemusena olemasolevad rakendused ja päringud töötavad endiselt (ainult et loodetavasti kiiremini).
- ♦ Paljud järgnevalt kirjeldatavad meetmed tähendavad muudatuste tegemist andmebaasi **sisemisel tasemel**.



Tabeli ridade sorteerimine sisemisel tasemel (failides)

- ♦ Tabeli read on *sorteerimata*.
 - Ridade lisamine **kiire**.
 - Ridade otsimine ilma indeksita **aeglane**.
- ♦ Tabeli read on mingile veergude hulgale vastavate väärtuste järgi *sorteeritud*.
 - Ridade otsimine sorteerimise aluseks olnud veeru/veergude järgi on **kiire**.
 - Ridade lisamine, sorteerimise aluseks olevates veergudes väärtuste muutmine, ridade kustutamine **aeglane**.

MS SQL Serveris sellele analoogiline tabeli klasterdatud indeks

Oracle indeksi alusel organiseeritud tabeli eelised

- ♦ Summaarne **andmemah** võib väheneda – pole vaja eraldi hoida tabelit ja primaarvõtmele loodud indeksit.
- ♦ Read on primaarvõtme väärtuste järgi **sorteeritud** – võimaldab kokku kuuluvad andmed paigutada füüsiliselt lähedikk.
- ♦ Väheneb **plokkide arv**, mida süsteem peab lugema, kui ridu otsitakse konkreetse primaarvõtme väärtuse või primaarvõtme väärtuste vahemiku alusel.
- ♦ Primaarvõtme väärtuste alusel sorteerimist nõudvaid **päringuid** on süsteemil lihtsam täita.

Analoogial põhinev näide

Indeks

Afghanistan – 5
Albania – 4
Algeria – 2
Andorra – 3
Angola – 1

"Tavaline" tabel

1. Angola, Republic of Angola, ÜRO liige
2. Algeria, People's Democratic Republic of Algeria, ÜRO liige
3. Andorra, Principality of Andorra, ÜRO liige
4. Albania, Republic of Albania, ÜRO liige
5. Afghanistan, Islamic Republic of Afghanistan, ÜRO liige

Indeksi alusel organiseeritud tabel

Afghanistan, Islamic Republic of Afghanistan, ÜRO liige
Albania, Republic of Albania, ÜRO liige
Algeria, People's Democratic Republic of Algeria, ÜRO liige
Andorra, Principality of Andorra, ÜRO liige
Angola, Republic of Angola, ÜRO liige

Oracle indeksi alusel organiseeritud tabeli puudused

- ♦ Saab luua vaid **primaarvõtme** alusel.
- ♦ Kui primaarvõtme väärtused monotoonselt kasvavad, siis hakkavad lisamisoperatsioonid **võistlema** samade plokkide pärast ja see aeglustab lisamist.
- ♦ Rea lisamine tabelisse ja võtmeväärtuse muutmine **aeglane**.

IOT – millal kasutada?

- Seostabelid, mis sisaldavad kaks või rohkem välisvõtme veergu.
 - CREATE TABLE Tellimuse_rida (

tellimus_id NUMBER(10),

kaup_id NUMBER(10),

kogus NUMBER(5) NOT NULL,

PRIMARY KEY (tellimus_id, kaup_id))

ORGANIZATION INDEX;
 - CREATE INDEX tellimuse_rida_kaup_idx ON Tellimuse_rida(kaup_id);

IOT – millal kasutada? (2)

- Eraldi indeksit välisvõtme veerule *tellimus_id* pole vaja – selle järgi on loodud IOT indeksi struktuur.
- Lisaks on iga tellimuse kõik read salvestatud füüsiliselt lähedastiku:
 - SELECT * FROM Tellimuse_rida

WHERE tellimus_id=väärtus;
 - Päringu täitmiseks peab lugema vähe plokkide.
 - IOTd võib kasutada ka mitte-seostabeli puhul, kui soovitakse, et tabeli andmed oleksid andmebaasi sisemisel tasemel primaarvõtme väärtuste alusel sorteeritud.

Tabelite klastrisse koondamine (Oracle)

- Koos kasutatavad andmed on ühendatud andmebaasi **sisemisel** tasemel.
- Tabeli ühendamist vajavate päringute täitmiseks peab lugema **vähem plokkide**.
- Räisfunktsioonil põhineva klasterdamise korral pole klasteri võtmele indeksit vaja, sest andmed ise ongi indeksiks (määravad ploki, kuhu nad tuleb salvestada).

Tabelite klastrisse koondamine (Oracle) (2)

Tabelid andmebaasi konseptuaalsel tasemel



Tabelite andmete salvestamine andmebaasi **sisemisel** tasemel

Isikukood	eesnimi	perenimi	auto_nr	mark
12345678	Andres	Mesikapp	123EES	Volvo
23344444	Jüri	Kavalpea	344DDR	Ford
			212WEW	Saab
			312EES	Ford

Klasteri võti (isikukood)

Tabelite klastrisse koondamine (Oracle) – millal kasutada

- Klasterda 1:M suhtes osalevad tabelid, kui tüüpiline päring soovib ridu nii **primaarsest** tabelist kui ka **sõltuvast** tabelist.
- Ära paiguta tabeleid klastrisse, kui klasteri võtme veerus muudetakse sageli andmeid – sageneb ridade **migreerumine**.
- Kasuta klasterit kui tabelite suurus ei muutu või nende maksimaalne suurus on **prognoositav**.

Tabelite klastrisse koondamine (Oracle) – puudused

- Aeglasem andmete **lisamine**.
- Klastrisse kuuluva tabeli andmete hoidmiseks kulub rohkem plokkide võrreldes olukorraga, kui klasterit ei kasutata.
 - Klastrisse kuuluva tabeli plokkide **täielik läbiskaneerimine** tähendab suurema arvu plokkide lugemist.
- Nõuab väga täpset andmete hulga **prognoosimist**.
- Klasterdatud tabelid ei võimalda andmete pakkimist (Oracle 12c Release 1).

Andmebaasid II 2017 © Erki Eessaar

Räsifunktsioonil põhinev klaster, kuhu kuulub üks tabel

- Kasutada **klassifikaatorite** tabelite puhul.
- Andmeid muudetakse harva – ridade migreerumine pole suur probleem.
- SELECT nimi FROM Riik WHERE riigi_kood=ette_antud_kood;
 - (riigi_kood) – klasteri võti.
- Päringu täitmiseks räsifunktsioon(ette_antud_kood) => riigi andmeid sisaldava plokki asukoht.

10.01.2018 Teema 4 55

Andmebaasid II 2017 © Erki Eessaar

Plokkide suurus



- Suhteliselt **suure** plokki **eelised**.
 - Päistele kulub suhteliselt vähem ruumi.
 - Väiksem tõenäosus *ridade migreerumiseks* või *rea jagamiseks mitme plokki vahel (plokkide ahel)*.
- Suhteliselt **suure** plokki **puudused**.
 - Mällu loetakse palju andmeid, mida kellelgi pole parajasti vaja (lugemine käib plokkide kaupa).
 - Palju asjatut *saalimist* mälu ja ketta vahel

10.01.2018 Teema 4 56

Andmebaasid II 2017 © Erki Eessaar

Andmete plokkidesse paigutamine (Oracle)

- PCTFREE (vaikimisi 10) – määrab, mitu % plokki kogumahust võib veel olla vaba, kui vastav plokk kuulutatakse täidetuks ja sellesse ei lubata uusi ridu lisada.
 - Kui PCTFREE=10, siis peab olema **10%**.

10.01.2018 Teema 4 57

Andmebaasid II 2017 © Erki Eessaar


Andmete plokkidesse paigutamine (Oracle) (2)

- PCTUSED (vaikimisi **40**) – määrab, mitu % plokki kogumahust on veel täidetud, kui plokk kuulutatakse uuesti vaba olevaks ja lubatakse sellesse uusi ridu lisada
 - Kui PCTUSED=40, siis peab olema **alla 40%**.
- PCTFREE saab määrata nii baastabelite, hetktõmmiste kui ka indeksite korral. PCTUSED ei saa määrata indeksi korral.

10.01.2018 Teema 4 58

Andmebaasid II 2017 © Erki Eessaar

Andmete plokkidesse paigutamine (Oracle) (2)



- PCTFREE väike:
 - salvestusruumi **kokkuhoid** – plokk pakitakse tihedalt täis,
 - tabeli/indeksi täielikul läbiskaneerimisel tuleb lugeda **vähem** plokkide,
 - suurem** ridade migreerumise tõenäosus.
- PCTFREE suur:
 - salvestusruumi **raiskamine** – plokis palju vaba ruumi,
 - tabeli/indeksi täielikul läbiskaneerimisel tuleb lugeda **rohkem** plokkide,
 - väiksem** ridade migreerumise tõenäosus.

Päis

Vaba ruum

Tabelite read


Väiksem loetud plokkide arv – parandab töökiirust

Ridade migreerumine – vähendab töökiirust (nii lugemisel kui andmete muutmisel)

10.01.2018 Teema 4 59

Andmebaasid II 2017 © Erki Eessaar

Andmete plokkidesse paigutamine (Oracle) (3)



- PCTUSED suur:
 - salvestusruumi **kokkuhoid** – ei lasta tekkida plokkidel, kus on palju tühja ruumi,
 - andmete lisamine **aeglasem**, sest erinevad read võib olla vaja panna erinevatesse plokkidesse; vabade plokkide küsimine/lugemine võtab aega,
 - eelnev tekitab **killustumist**, sest koos lisatud read võivad sattuda eraldi plokkides. See võib hiljem päringuid aeglaseks muuta.
- PCTUSED väike:
 - salvestusruumi kulub **rohkem**, kuid andmete lisamine ja muutmine kiirem ning killustumist tekib **vähem**,
 - sobib tabelitele, kus palju INSERT, UPDATE ja DELETE operatsioone.

10.01.2018 Teema 4 60

Andmebaasid II 2017 © Erki Eessaar

Andmete plokkidesse paigutamine (PostgreSQL)



- Tabelite ja indeksitega seotud salvestusparameetri **FILLFACTOR** väärtus (vaikimisi **100**) määrab, kui mitu protsenti plokkist peab olema täidetud, et sinna ei saaks enam lisada **INSERT** operatsiooniga uusi ridu.
 - Muuda **FILLFACTOR** vaikimisi väärtust, kui tabelis **palju UPDATE operatsioone**, mis võivad olemasolevate ridade **andmemahu suurendada**.
 - Võimalik indikatsioon – mittekohustuslikud veerud.

10.01.2018 Teema 4 61

Andmebaasid II 2017 © Erki Eessaar

Veel tabeli parameetreid (Oracle)

- Kui tabel on *süsteemikataloogi kaudu hallatavas tabeliruumis*.
 - INITIAL** – esimese ekstendi suurus
 - Tabeli puhul tuleks see valida nii, et esialgne ekstent mahutaks kõik algselt tabelisse lisatavad read.
 - See garanteeriks nende ridade paiknemise järjestikustes plokkides.
 - NEXT** – teise eraldatava ekstendi suurus.
 - PCTINCREASE** (vaikimisi 50) – alates kolmandast ekstandist, kui mitu protsenti on järgmine eelmisest suurem.
 - INITIAL=NEXT & PCTINCREASE=0** – kõik ekstandid on ühesuurused.

10.01.2018 Teema 4 62

Andmebaasid II 2017 © Erki Eessaar

Veel tabeli parameetreid (Oracle)(2)

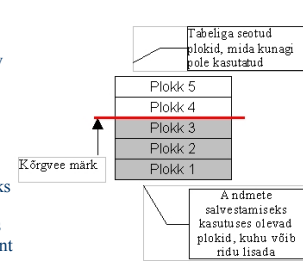
- LOGGING (vaikimisi) | NOLOGGING** – kas genereerida teatud olukorras (nt andmete laadimine tabelisse kasutades **SQL*Loaderit**) redo logi.
- FREELIST GROUPS** (vaikimisi 1) – vabade plokkide nimekirjade gruppide arv (kasutusel Oracle Real Application Clusters).
- FREELIST** (vaikimisi 1) – vabade plokkide nimekirjade arv.

10.01.2018 Teema 4 63

Andmebaasid II 2017 © Erki Eessaar

Freelist (Oracle)

- FREELIST** – plokkide nimekiri kuhu saab lisada ridu.
- Tabeliga seotud **FREELIST**ide arv võiks olla sama, mis võimalik andmeid samaaegselt tabelisse lisavate transaktsioonide arv.
- Eelis** – lisamine võib kiireneeda.
- Probleem** – salvestusruumi raiskamine. Iga transaktsiooni jaoks otsitakse vabu plokkide vaid ühest nimekirjast ja kui sealt ei leita, siis lisatakse tabeli segmenti uus ekstent (defragmenteerumine, tabeliga seotud plokkide arvu suurenemine).



10.01.2018 Teema 4 64

Andmebaasid II 2017 © Erki Eessaar

Veel tabeli parameetreid (Oracle)(3)

- INITRANS** (tabelitel vaikimisi 1, indeksitel 2) – esialgne transaktsioonide andmete pesade arv.
 - Määrab *esialgse* samaaegselt plokki kasutada saavate transaktsioonide arvu.
 - Oracle saab pesade arvu suurendada dünaamiliselt kuni **MAXTRANS** väärtuseni.
 - Probleem** – plokk on andmeid täis, kuid seda soovib paralleelselt kasutada palju transaktsioone. Transaktsioonid peavad üksteise järgi ootama, sest pole vabu pesasid.
- MAXTRANS** – maksimaalne transaktsioonide andmete pesade arv.
 - Määrab *maksimaalse* samaaegselt plokki kasutada saavate transaktsioonide arvu.
 - Viimastes Oracle versioonides väärtus alati 255.

10.01.2018 Teema 4 65

Andmebaasid II 2017 © Erki Eessaar

Veel tabeli parameetreid (Oracle)(4)

- CACHE | NOCACHE (vaikimisi)** – kas andmebaasisüsteem üritab hoida tabeli täieliku läbiskaneerimise käigus loetud plokkide *võimalikult kaua* muutmälus (**CACHE**) või mitte (**NOCACHE**).
 - Ka **CACHE** puhul kustutatakse plokkide muutmälust.
- PARALLEL | NOPARALLEL (vaikimisi)** – kas tabeli andmete töötlemisel kasutada paralleeltööd (mitme CPU olemasolu korral) või mitte.
- BUFFER_POOL** – saab määrata millises andmepuhvri alamosas (*default, keep, recycle*) hoitakse muutmällu lugemise järel tabeli plokkide.

10.01.2018 Teema 4 66

BUFFER_POOL

- ♦ Puulide nimed on lihtsalt nimed, mitte garantii süsteemi käitumise kohta.
 - Näide: Kui tahan, et mingite tabelite/indeksite andmed oleksid mälus (loodetavasti) pidevalt, siis loon piisavalt suure hoiupuuli, kuhu kõik need andmed ära mahuks ning määran tabeli/indeksi spetsifikatsioonis, et nende plokid tuleks mällu lugedes panna just hoiupuuli.

Tabeli loomise lause genereerimine (Oracle)

- ♦ `SELECT dbms_metadata.get_ddl('TABLE', 'YLIOPILANE', 'C##TUD1') FROM Dual;`
- ♦ Kasulik meetod, et vaadata, mis omadusi üldse saab tabeli juures määrata ja millised on vaikinisi omadused.

Tabeli loomise lause (Oracle)

```
CREATE TABLE "C##TUD1"."YLIOPILANE"
("MATRIKLI_NR" CHAR(6),
"YLIOPILASE_SEISUNDI_LIIK_R_ID" NUMBER(4,0)
DEFAULT 1 NOT NULL ENABLE,
"ISIKUKOOD" CHAR(11) NOT NULL ENABLE,
"EESNIMI" VARCHAR2(1000) NOT NULL ENABLE,
"PERENIMI" VARCHAR2(1000) NOT NULL ENABLE,
"ELUKOHT" VARCHAR2(1000),
"PILT" BLOB,
"E_MAIL" VARCHAR2(254) NOT NULL ENABLE,
```

Tabeli loomise lause (Oracle) (2)

```
CONSTRAINT "CHK_YLIOPILANE_E_MAIL" CHECK
(e_mail LIKE '%@%') ENABLE,
CONSTRAINT "CHK_YLIOPILANE_PERENIMI" CHECK
(NOT REGEXP_LIKE(perenimi, '^[[:space:]]*$'))
ENABLE,
CONSTRAINT "CHK_YLIOPILANE_EESNIMI" CHECK
(NOT REGEXP_LIKE(eesnimi, '^[[:space:]]*$')) ENABLE,
```

Märkus Oracle kohta (võib lugeda puuduseks)

- ♦ Oracle jaoks " (tühi string) = NULL
- ♦ `CREATE TABLE Yliopilane (yliopilane_id NUMBER(10) PRIMARY KEY, perenimi VARCHAR2(1000) NOT NULL);`
- ♦ `INSERT INTO Yliopilane(yliopilane_id, perenimi) VALUES(1,");`
- ♦ ERROR at line 1:
- ♦ ORA-01400: cannot insert NULL into ("C##TUD1"."YLIOPILANE"."PERENIMI")

Märkus Oracle kohta (2)

- ♦ Seega ei saa kasutada järgnevat kitsendust, et vältida selliste ridade lisamist, kus perenimi koosneb tühikutest.
 - `CONSTRAINT chk_yliopilane_perenimi CHECK (trim(perenimi)!='')`
- ♦ Põhjendus:
 - Kitsenduse kontrollimisel:
 - `trim(tühikutest koosnev string) => NULL`
 - `NULL != NULL => unknown`
 - Seega saan lisada NOT NULL veergu tühikutest koosneva stringi

Trim – süsteemi-definieeritud funktsioon stringi algusest ja/või lõpust märkide eemaldamiseks. Vaikimisi eemaldab tühikud nii stringi algusest kui lõpust.

Märkus Oracle kohta (3)

- ♦ CREATE TABLE Yliopilane (yliopilane_id NUMBER(10) PRIMARY KEY, perenimi VARCHAR2(1000) NOT NULL, CONSTRAINT chk_yliopilane_perenimi CHECK (trim(perenimi) IS NOT NULL));
 - Ei saa registreerida üliõpilasi, kelle perenimi on tühi string või tühikutest koosnev string.
 - Kuna PostgreSQL ei tee teisendust "=> NULL, siis seal selline kitsendus tulemust ei anna.

Tabeli loomise lause (Oracle) (3)

```
CONSTRAINT "PK_YLIOPILANE" PRIMARY KEY
("MATRIKLI_NR")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
TABLESPACE "USERS" ENABLE,

CONSTRAINT "AK_YLIOPILANE_ISIKUKOOD" UNIQUE
("ISIKUKOOD")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
TABLESPACE "USERS" ENABLE
```

Indeks on andmebaasi sisemise taseme objekt, kuhu salvestatakse andmeid. Salvestusomaduste määramiseks saab kasutada **STORAGE** klauslit.

Tabeli loomise lause (Oracle) (4)

```
CONSTRAINT "AK_YLIOPILANE_E_MAIL" UNIQUE
("E_MAIL")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS
255 TABLESPACE "USERS" ENABLE
```

Tabeli loomise lause (Oracle) (5)

```
SEGMENT CREATION DEFERRED
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
TABLESPACE "USERS"
```

```
LOB ("PILT") STORE AS SECUREFILE (
TABLESPACE "USERS" ENABLE STORAGE IN ROW CHUNK 8192
NOCACHE LOGGING NOCOMPRESS KEEP_DUPLICATES )
```

- ♦ **ENABLE STORAGE IN ROW** – kui LOB väärtus väiksem kui ~4000 baiti, siis salvestatakse koos ülejäänud reaga. Kui suurem kui ~4000 baiti, siis salvestatakse LOB väärtus eraldi segmendis ning rea juures ainult viide LOBi asukohale.

Salvestusruumi säästmise võimalus (Oracle)

- ♦ SEGMENT CREATION DEFERRED:
 - tabeli segment,
 - tabeliga seotud indeksite segmendid,
 - tabelis olevate LOB tüüpi andmete hoidmiseks mõeldud segmendid
- ♦ luuakse ning esimene ekstant eraldatakse alles esimese rea lisamisel tabelisse.
- ♦ Võimalik alates Oracle 11g Release2 ning seal ka vaikimisi käitumine.

Traditsiooniline andmete sisemisel tasemel salvestamise viis

- ♦ Andmebaasisüsteemid, nagu Oracle ja PostgreSQL, salvestavad **sisemisel** tasemel andmeid *ridade* kaupa – ühes *plokis* on ühe või mitme rea andmed.
- ♦ Näide:
 - Isik (isikukood, esinimi, perenimi)
Primaarvõti (isikukood)
 - Plokis olevate andmete näide:
 - 38101020123, Andres, Mets;
 - 38405100123, Kairi, Kask;
 - 38611123210, Mati, Metsis;

Alternatiivne andmete sisemisel tasemel salvestamise viis

- ♦ SQL-andmebaasisüsteemid nagu *C-store*, *Vertica*, *MonetDB*, *CitusDB* jne, aga ka MS SQL server (*columnstore index*) kasutavad **sisemisel** tasemel **veerupõhist** salvestamist.
- ♦ Ühes plokis on koos andmed, mis vastavad kõik ühele tabeli veerule.
- ♦ Plokis olevate andmete näide:
 - 38101020123, 38405100123, 38611123210

10.01.2018

Teema 4

79

Alternatiivne andmete sisemisel tasemel salvestamise viis (2)

- ♦ Selleks, et süsteem saaks panna kokku tabelite read, peavad plokkides olema andmed **ühel põhimõttel sorteeritud**.
 - Selleks, et koostada tabeli X n-s rida, peab süsteem lugema igale X veerule vastavate plokkide hulgast n-nda väärtuse.
 - Sorteerimine võib toimuda ridade lisamise järjekorras, aga ka mingite tabeli X veergude alusel.

10.01.2018

Teema 4

80

Ei mõjuta seda, kuidas kasutaja andmeid näeb

Veerupõhine vs. reapõhine salvestamine

- ♦ Veerupõhise salvestamise eelised.
 - Kuna päringud küsivad andmeid enamasti vaid **mõnedest** tabeli **veergudest**, siis päringule vastamiseks ei pea andmebaasisüsteem vaatama läbi kõiki tabeli ridade andmeid sisaldavaid plokke.
 - Mida vähem veerge päringus, seda kiiremini see täidetakse.
 - Ühte tüüpi andmeid on lihtsam ja efektiivsem **pakkida**.
- ♦ Reapõhise salvestamise eelised.
 - Ridade **lisamine**, **muutmine** ja **kustutamine** kiirem.
 - **Üksiku rea otsimine** kiirem.

10.01.2018

Teema 4

81

Veerupõhine vs. reapõhine salvestamine (2)

- ♦ Veerupõhine salvestamine.
 - Sobib kasutada andmebaasides, kus põhiliselt toimub suure hulga andmete lugemine, kuid päringud ei puuduta kõiki tabeli veerge (**andmeid**, **OLAP** süsteemid).
- ♦ Reapõhine salvestamine.
 - Sobib operatiivandmete andmebaasides, kus palju operatsioone üksikute ridadega.

10.01.2018

Teema 4

82

PostgreSQL laiendus

- ♦ `cstore_fdw` (https://citusdata.github.io/cstore_fdw/)
- ♦ Installeerimine andmebaasi CREATE EXTENSION lausega.
- ♦ Saab luua väliseid tabeleid, mille andmed on salvestatud veerupõhiselt.
- ♦ Samas andmebaasis nii rea- kui veerupõhise salvestusega tabelid.
- ♦ Annab eelise andmete kettalt lugemisel.

10.01.2018

Teema 4

83

Lisalugemine


- ♦ Puustusmaa, S., 2016. *Reapõhise ja veerupõhise andmete salvestamise võrdlus kahe SQL-andmebaasisüsteemi näitel*. Magistritöö. TTÜ Informaatikainstituut [WWW] <https://digi.lib.ttu.ee/i/?4101>
 - Microsoft SQL Server 2014 ja MonetDB 5
 - MS SQL 2014 salvestab andmeid reapõhiselt. Selles on indeksi tüüp, mis salvestab andmed veerupõhiselt. MonetDB kasutab veerupõhist salvestamist.

10.01.2018

Teema 4

84

Andmebaasid II 2017 © Erki Eessaar




Hetktõmmised

- ♦ Igal hetktõmmisel on **alampäring**, mis defineerib hetktõmmisesse kuuluvad andmed.
- ♦ Päringu tulemus leitakse ja **salvestatakse** juba **enne**, kui keegi soovib seda päringut käivitada.
- ♦ Piira kasutatavat **salvestusruumi** (mitte üle 20% andmebaasi mahust).

10.01.2018 Teema 4 85

Andmebaasid II 2017 © Erki Eessaar




Hetktõmmised (2)

- ♦ Oracle võib **päringuid ümber kirjutada**, et need kasutaks hetktõmmist.
 - ENABLE QUERY REWRITE hetktõmmise definitsioonis.
- ♦ Liiga suur hetktõmmiste hulk
 - kulutab liigselt **salvestusruumi**,
 - Oracle korral teeb andmebaasisüsteemile raskeks päringu **täitmisplaani valimise**.

10.01.2018 Teema 4 86


Andmebaasid II 2017 © Erki Eessaar



Hetktõmmised (3)

- ♦ Loo **vaadete** põhjal, mida kasutatakse sageli päringutes.
- ♦ **Indekseeri**.
- ♦ Defineeri **kitsendused**, et anda andmebaasisüsteemile rohkem infot.
 - Võimalik Oracles, kuid mitte PostgreSQLis.


Andmebaasid II 2017 © Erki Eessaar



Hetktõmmised (4)

- ♦ Välti Oracle hetktõmmise alampäringus korreleeruvaid alampäringuid ja SELECT DISTINCT kasutamist
 - Andmebaasisüsteemil raskem otsustada, kas päringut saab hetktõmmise põhjal täita või mitte.

Andmebaasid II 2017 © Erki Eessaar




Tabeli, indeksi pakkimine (Oracle)

- ♦ Eelised.
 - Kulub vähem salvestusruumi.
 - Rohkem andmeid mahub mällu. Otsimine kiirem.
- ♦ Puudused.
 - Andmete muutmine aeglasem.
 - Andmete muutmine võib tingida *ridade migreerumise*.

10.01.2018 Teema 4 89

Andmebaasid II 2017 © Erki Eessaar



Tabeli, indeksi pakkimine (Oracle) (2)

- ♦ Kasuta andmete pakkimist tabelite juures, milles **muudatused on harvad** (nt ajaloolised andmed).
- ♦ Kui suures tabelis nii aktiivsed kui ajaloolised andmed, siis jaga tabel selle alusel **sektsoonideks** ja paki sektsioon/sektsoonid, kus on ajaloolised andmed.

10.01.2018 Teema 4 90

Indeksid

- Andmebaasi päringute täitmise kiiruse parandamiseks saab kasutada indekseid.
- Vaatleme näitena *bitmap indeksit*, mida saab kasutada Oracle andmebaasisüsteemis, aga ei saa kasutada PostgreSQLis.

10.01.2018

Teema 4

91

Bitmap indeks – näide (Oracle)

Tabel

Arve	arve_nr	arve_kuupaev	tasumise_viis	summa	ostja_tuup
1	01.04.2002	sularahas	1000	eraisik	
2	01.04.2002	ülekandega	2000	firma	
3	05.04.2002	ülekandega	5000	firma	
4	08.04.2002	arvega	3000	eraisik	
5	10.04.2002	ülekandega	1500	eraisik	
6	12.04.2002	arvega	2200	firma	
7	12.04.2002		2500	eraisik	

Bitmap indeks, mis on loodud veerule *tasumise_viis*

rida	tasumise_viis= "sularahas"	tasumise_viis= "ülekandega"	tasumise_viis= "arvega"	tasumise_viis= NULL
1	1	0	0	0
2	0	1	0	0
3	0	1	0	0
4	0	0	1	0
5	0	1	0	0
6	0	0	1	0
7	0	0	0	1

Bitivektorid

10.01.2018

Teema 4

92

Bitmap indeks – päringu töökiiruse näide (1) (Oracle)

- Tabelis *customer* on **12 011 201** rida

```
SELECT cst_type, count(*) AS amt
FROM Eshop.customer
GROUP BY cst_type;
```

10.01.2018

Teema 4

93

Bitmap indeks – päringu töökiiruse näide (2) (Oracle)

Tulemus:

CST_TYPE	AMT

1	8245655
2	3765546

Elapsed: **00:00:48.81**

10.01.2018

Teema 4

94

Bitmap indeks – päringu töökiiruse näide (3) (Oracle)

Execution Plan

```
0  SELECT STATEMENT Optimizer=CHOOSE (Cost=24986 Card=2
   Bytes=6)
1  0  SORT (GROUP BY) (Cost=24986 Card=2 Bytes=6)
2  1  TABLE ACCESS (FULL) OF 'CUSTOMER' (Cost=9544
   Card=12011201 Bytes=36033603)
```

Statistics

99300 consistent gets
99022 physical reads

Füüsiline täitmisplaan,
 mille alusel toimus
 päringu täitmine.

10.01.2018

Teema 4

95

Bitmap indeks – päringu töökiiruse näide (4) (Oracle)

- CREATE BITMAP INDEX bidx_customer_cst_type ON customer(cst_type);

- Päring uuesti

CST_TYPE	AMT

1	8245655
2	3765546

Elapsed: **00:00:00.30**

10.01.2018

Teema 4

96

Bitmap indeks – päringu töökiiruse näide (5) (Oracle)

Execution Plan

```

0  SELECT STATEMENT Optimizer=CHOOSE (Cost=29 Card=2
   Bytes=6)
1  0  SORT (GROUP BY NOSORT) (Cost=29 Card=2 Bytes=6)
2    1  BITMAP CONVERSION (COUNT)
3    2  BITMAP INDEX (FULL SCAN) OF
       'BIDX_CUSTOMER_CST_TYPE'

```

Statistics

460 consistent gets
426 physical reads

10.01.2018

Teema 4

97

Bitmap indeks aeglustab andmete muutmist (Oracle)

- ♦ UPDATE Arve SET tasumise_viis='sularahas' WHERE arve_nr=4; /*Vana tasumisviis oli 'arvega'*/
- ♦ Bitivektorid, mille see SQL lause **lukustab**, on näidatud järgmisel joonisel rasvaselt esile tõstetuna.

rida	tasumise_viis="sularahas"	tasumise_viis="ulekandega"	tasumise_viis="arvega"	tasumise_viis=NULL
1	1	0	0	0
2	0	1	0	0
3	0	1	0	0
4	0	0	1	0
5	0	1	0	0
6	0	0	1	0
7	0	0	0	1

10.01.2018

Teema 4

98

Bitmap indeks aeglustab andmete muutmist (2) (Oracle)

- ♦ Näiteid teiste sessioonide UPDATE lausetest, mille täitmine **jääb** eelnimetatud lukustamise tulemusel ootele:
 - UPDATE Arve SET tasumise_viis='sularahas' WHERE arve_nr=6;
 - UPDATE Arve SET tasumise_viis='ulekandega' WHERE arve_nr=1;
- ♦ Näiteid teiste sessioonide UPDATE lausetest, mille täitmine **ei jää** eelnimetatud lukustamise tulemusel ootele:
 - UPDATE Arve SET summa=2000 WHERE arve_nr=6;
 - UPDATE Arve SET tasumise_viis=NULL WHERE arve_nr=2;

10.01.2018

Teema 4

99

Bitmap indeks aeglustab andmete muutmist (3) (Oracle)

- ♦ Töökiirust halvendab ka see, et ridade lisamisel/kustutamisel või bitmap indeksiga indekseeritud veerus andmete muutmisel, tuleb muuta vastavat bitmap indeksit.

10.01.2018

Teema 4

100

Soovitusi bitmap indeksi kasutamiseks (Oracle)

- ♦ Kasuta bitmap indeksi *andmeaitades* ja *andmevakkades*, sest neid andmebaase iseloomustavad:
 - suured andmemahud,
 - sagedased ja keerukad päringud koondandmete leidmiseks,
 - ei toimu olemasolevate andmete muutmist.

10.01.2018

Teema 4

101

Üldiseid soovitusi indeksi valikuks

- ♦ Indeksid võtavad andmemahust **10–20%**. Üle 25% nõuaks juba põhjalikku indeksite ülevaataust.
- ♦ Indeksid
 - kiirendavad mõningaid päringuid,
 - aitavad kiiremini leida ridu, mida tuleb muuta või kustutada,
 - kuid aeglustavad ridade lisamist ja indekseeritud veergudes andmete muutmist.

10.01.2018

Teema 4

102

Andmebaasid II 2017 © Erki Eessaar

Üldiseid soovitusi indeksi valikuks (2)

- ♦ Jälgi indeksite kasutamist ja eemalda indeksid, mida päringute täitmiseks ei kasutata.
- ♦ Välti üldjuhul üksteist **dubleerivad** indeksid.
 - Mis kasu on raamatu lõpus olevast kahest ühesugusest indeksist?

10.01.2018 Teema 4 103

Andmebaasid II 2017 © Erki Eessaar

Mitu indeksit ühesugusel veergude hulgal?

- ♦ Oracle lubab alates versioonist 12c Release 1
- ♦ Milline võiks olla sellest saadav kasu?
 - Testida töökiirust erinevate indeksite korral
 - Üks indeks on andmebaasisüsteemile nähtav, ülejäänud „nähtamatud“ (andmebaasisüsteem ei kasuta neid; indeksi nähtamatuks muutmine võimalik alates Oracle 11g).
 - Tagada käideldavust
 - Indeksi asendamisel luua uus indeks „nähtamatuks“ ning siis asendada uus indeks vanaga. Lüheneb üleminekuperiood vana ja uue indeksi vahel, millal kumbagi indeksit ei saa kasutada.

10.01.2018 Teema 4 104

Andmebaasid II 2017 © Erki Eessaar

Miks indekseerida (B-puu indeksiga) *Mark_ID* tabelis *Auto*?

- ♦ DELETE FROM Mark WHERE Mark_ID=1;
 - Kiirendab kontrolli, kas kustutataval real on seotud ridu tabelis *Auto*.
 - Kui tabelis *Auto* ei ole veerule *Mark_ID* loodud indeksit, siis soovib Oracle rea kustutamisel tabelist *Mark* lukustada terve tabeli *Auto*, et vältida uute autode lisamist, mis on kustutatavat marki – pole võimalik, kui mõni transaktsioon juba lukustab tabelit *Auto* või mõnda selle rida.
- ♦ Võivad kiireneda päringud nagu
 - SELECT * FROM Auto WHERE Mark_ID=:X;
 - SELECT * FROM Auto WHERE Mark_ID BETWEEN :X AND :Y;
 - SELECT * FROM Auto, Mark WHERE Auto.Mark_ID = Mark.Mark_ID AND Mark.Mark_ID = :X;

10.01.2018 Teema 4 105

Andmebaasid II 2017 © Erki Eessaar

Näide (Oracle)

Stsenaarium (1) – *Auto.Mark_ID* ei ole indekseeritud

Aeg	Transaktsioon 1	Transaktsioon 2
1	DELETE FROM Auto WHERE auto_id=2; Lukustatakse rida tabelis <i>Auto</i> . Kustutamine õnnestub.	
2		DELETE FROM Mark WHERE mark_id=1; Jäab ootele, kuna tabelit <i>Auto</i> ei saa lukustada.

Kui *Auto.Mark_ID* ei ole indekseeritud, siis PostgreSQL lubab ikkagi mõlemad kustutamisoperatsioonid samaaegselt läbi viia.

Stsenaarium (2) – *Auto.Mark_ID* on indekseeritud

```
CREATE INDEX idx_auto_mark ON Auto(mark_id);
```

Aeg	Transaktsioon 1	Transaktsioon 2
1	DELETE FROM Auto WHERE auto_id=2; Lukustatakse rida tabelis <i>Auto</i> . Kustutamine õnnestub.	
2		DELETE FROM Mark WHERE mark_id=1; Kustutamine õnnestub, sest read tabelis <i>Auto</i> , mida on vaja kustutada, ei ole lukustatud.

10.01.2018 Teema 4 106

Andmebaasid II 2017 © Erki Eessaar

Millal kaaluda tabelis *Auto* *Mark_ID* indekseerimata jätmist?

- ♦ Indeksit läheb harva vaja.
 - Tabelist *Mark* kustutatakse ridu või muudetakse primaarvõtme väärtuseid väga harva ja/või väljaspool andmebaasi aktiivse kasutamise aega.
 - Tabeleid *Auto* ja *Mark* ühendatakse päringutes harva.
 - Tabeli *Auto* veergu *Mark_ID* kasutatakse harva päringute tingimustes.
- ♦ Indeks aeglustab teatud andmemuudatusi.
 - Registreeritud autode marke muudetakse sageli.

10.01.2018 Teema 4 107

Andmebaasid II 2017 © Erki Eessaar

Millal välisvõtit mitte indekseerida?

- ♦ Tellimuse_rida(tellimus_id, kaup_id, kogus)
 - Primaarvõti (tellimus_id, kaup_id)
 - Välisvõti (tellimus_id)
 - Viitab Tellimus(tellimus_id)
 - Välisvõti (kaup_id) Viitab Kaup(kaup_id)
- ♦ Enamik andmebaasisüsteeme loob automaatselt liitindeksi (tellimus_id, kaup_id):
 - tellimus_id – esimene veerg liitindeksis,
 - saab alati kasutada, kui on vaja leida konkreetse tellimuse read.
- ♦ Järelikult mitte luua täiendavat indeksit (tellimus_id)!!
- ♦ Tuleb luua indeks (kaup_id).

10.01.2018 Teema 4 108

Andmebaasid II 2017 © Erki Eessaar

Miks on vaja *kaup_id* täiendavalt indekseerida?

Ulo Mets	1184
Valga	
Hannes Mets	1184
Inju, Vinni Vald, Lääne-Viru Maakond	
Ergo Mets	1184
Sõstra, Rakvere	
Martin Mets	1184
Kalda, Kadima, Kadima Vald, Lääne-Viru Maakond	
Natalje Mets	1184
Ravi 4, Väike-Maarja, Väike-Maarja Vald, Lääne-Viru Maakond	
Hennu Mets	1184
Rooel, Roela, Vinni Vald, Lääne-Viru Maakond	
Vive Mets	1184
Puru Tee 24, Jõhvi Vald, Jõhvi	
Vilma Mets	1184
Liva, Tolla, Tolla Vald, Ida-Viru Maakond	
Ullar Mets	1184
Puhkuse 9, Tolla, Tolla Vald, Ida-Viru Maakond	
Linda Mets	1184
Pik, Tolla, Tolla Vald, Ida-Viru Maakond	

- Analoogiaks telefoniraamat
 - Liitindeks (perenimi, eesnimi).
 - Kui otsin isikut perenime või perenime + eesnime järgi, siis on otsimine kiire, sest isikute andmed on perenime ja selle sees eesnime järgi sorteeritud.
 - Kui otsin isikuid **ainult** eesnime järgi, siis pean vaatama läbi kogu raamatu.
 - antud näites eesnimi = kaup_id

10.01.2018 Teema 4 109

Andmebaasid II 2017 © Erki Eessaar

Indeks ja surrogaatvõti (PostgreSQL näitel)

- CREATE TABLE Isik (isik_id SERIAL PRIMARY KEY, perenimi VARCHAR(1000) NOT NULL);
- Automaatselt luuakse indeks veerule *isik_id*.
- Veergu väärtuste genereerimiseks arvutada generaator sammuga 1 – iga järgmine väljastatud väärtus eelmisest ühe võrra suurem.

10.01.2018 Teema 4 110

Andmebaasid II 2017 © Erki Eessaar

Indeks ja surrogaatvõti (PostgreSQL näitel)(2)

Rea lisamisel on vaja värskendada indeksit.
Kõik muudatused koonduvad ühte plokki – tekib *kuumkoht*.
Kui palju paralleelseid lisamisi, siis ridade lisamise operatsioonid peavad hakkama üksteise järel ootama.
Kasutage rohkem sisulise tähendusega võtmeid – muudatused jaotuvad erinevate indeksi plokkide vahel.

10.01.2018 Teema 4 111

Andmebaasid II 2017 © Erki Eessaar

Üldised probleemid seoses indeksitega

- Indeksit ei looda üldse, või luuakse neid liiga **vähe**.
- Indeksit luuakse liiga **palju** või luuakse selliseid, millest tüüpiliste operatsioonide läbiviimisel pole **kasu**.
- Kasutatakse päringuid, mille puhul ühestki loodud indeksist pole kasu (koos päringute struktuuriga peavad muutuma ka indeksid).

10.01.2018 Teema 4 112

Andmebaasid II 2017 © Erki Eessaar

Andmemuudatuste kiirendamine (PostgreSQL)

- UNLOGGED tabel – andmemuudatuste kohta ei genereerita logi.
 - Andmete lisamine/muutmine/kustutamine tabelis kiirem, kuid peale süsteemi krahhi kustutatakse tabelis olevad andmed automaatselt, sest puudub logi, mille alusel tabelis olevad andmed korrektseks seisundis taastada.

10.01.2018 Teema 4 113

Andmebaasid II 2017 © Erki Eessaar

Andmemuudatuste kiirendamine (PostgreSQL) (2)

<ul style="list-style-type: none"> CREATE TABLE Temp_logged (temp INTEGER); INSERT INTO Temp_logged (temp) SELECT temp FROM Temp; --Lisan 100000 rida Aeg: 459.922 ms DELETE FROM Temp_logged; Aeg: 414.333 ms 	<ul style="list-style-type: none"> CREATE UNLOGGED TABLE Temp_unlogged (temp INTEGER); INSERT INTO Temp_unlogged (temp) SELECT temp FROM Temp; --Lisan 100000 rida Aeg: 80.742 ms DELETE FROM Temp_unlogged; Aeg: 44.611 ms
---	---

10.01.2018 Teema 4 114

Andmebaasiserveris talletatud rutiinide kasutamine

- ♦ **Rutiin** – protseduur või funktsioon, mingit kindlat tegumit täitev programmi osa.
- ♦ Andmebaasi kasutava rakenduste töökiiruse tõstmise huvides on kasulik koondada maksimaalne osa *andmete kasutamisega seotud* programmikoodist andmebaasiserveris talletatud rutiinidesse.
- ♦ Järgnev jutt Oracle põhjal, kuid üldiselt universaalsed põhimõtted.

10.01.2018

Teema 4

115

Andmebaasiserveris talletatud rutiinide kasutamine (2)

- ♦ *Protseduuridesse* võiks koondada andmetöötlusoperatsioonid.
 - Protseduurid võiks luua **andmebaasioperatsioonide lepingute** baasil.
- ♦ Samuti võib andmebaasiserveris luua *funktsioone*, mida saab kasutada andmetöötlusoperatsioonides.
 - `SELECT f_Fahrenheit_Celsius(temp) AS temp_Celsius, aeg FROM Mootmine WHERE f_Fahrenheit_Celsius(temp) > 25 ORDER BY aeg;`

10.01.2018

Teema 4

116

Andmebaasiserveris talletatud rutiinide kasutamine (3)

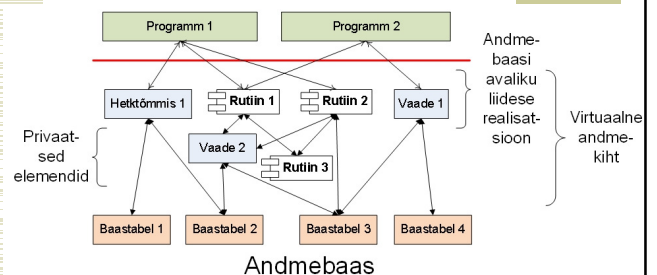
- ♦ PostgreSQL **lubab** kasutada CHECK kitsendustes kasutaja-defineeritud funktsioone, **Oracle ei luba**.
 - `CREATE TABLE Klient (klient_id SERIAL PRIMARY KEY, perenimi VARCHAR(1000) NOT NULL);`
 - `ALTER TABLE Klient ADD CONSTRAINT chk_perenimi_tyhikuid_liiga_palju CHECK(f_tyhikute_arv(perenimi) <= 3);`

10.01.2018

Teema 4

117

Andmebaasi avalik liides (välised skeemid ANSI/SPARC arhitektuuris)



10.01.2018

Teema 4

118

API vs. Tabeli API (TAPI)

API – programmi liides

[Aadress]-0..1----1-[Töötaja]-0..*-----1-[Osakond]

- | | |
|---|-------------------------------|
| ♦ API rutiini näide | ♦ TAPI rutiinide näide |
| ■ Operatsioon: Lisa töötaja | ■ Lisa aadress |
| ■ Eeltingimused. | ■ Loe aadress |
| • <i>Osakond</i> on registreeritud | ■ Muuda aadressi |
| ■ Järelingimused. | ■ Kustuta aadressi |
| • On loodud <i>Töötaja</i> | ■ Lisa töötaja |
| • On loodud seos <i>Töötaja</i> ja <i>Osakond</i> vahel | ■ Loe töötaja |
| • On loodud <i>Aadress</i> | ■ Muuda töötaja |
| • On loodud seos <i>Töötaja</i> ja <i>Aadress</i> vahel | ■ Kustuta töötaja |

10.01.2018

Teema 4

119

API vs. TAPI (2)


- ♦ Soovitan eelistada API rutiine
 - Igaüks aitab saavutada mingit kasutaja **eesmärki**, mis võib olla laiem kui üksiku rea tabelisse lisamine/lugemine/muutmine/kustutamine.
 - Iga rutiin moodustab eraldi **transaktsiooni**.
 - Võimaldavad töötada **ridade hulkadega**, kasutada paremini SQL võimalusi.
 - Looge andmebaasioperatsioonide **lepingute** põhjal.

10.01.2018

Teema 4

120

Andmebaasid II 2017 © Erki Eessaar



Operatsiooni leping

- ♦ **Eeltingimus** on lause, mis ütleb, millise eeldame olevat maailma enne operatsiooni sooritamist.
 - Täidetuse kontroll on väljakutsuja kohustus.
- ♦ **Järeltingimus** on lause, mis ütleb, milline peaks maailm olema pärast operatsiooni sooritamist.
 - Täidetuse kontroll on operatsiooni kohustus.
- ♦ **Erand** avaldub selles, et operatsiooni käivitamisel on eeltingimused täidetud, kuid pärast operatsiooni sooritamist pole täidetud järeltingimused.

10.01.2018 Teema 4 121

Andmebaasid II 2017 © Erki Eessaar

Invariantid – tingimused, mis peavad olema alati täidetud – andmebaasi korral **kitsendused!**

Andmebaasioperatsiooni leping

- ♦ Eeltingimused.
 - Millised andmed peavad olema andmebaasis registreeritud, et operatsiooni läbi viia.
- ♦ Järeltingimused.
 - Andmete muutmisega tegelevad operatsioonid – millised andmemuudatused on andmebaasis operatsiooni tulemusena toimunud.
 - Kui operatsioon peab tagastama andmeid – millised andmed operatsiooni tulemusena leiti.

10.01.2018 Teema 4 122

Andmebaasid II 2017 © Erki Eessaar

Miks tuleb rutiin töökiirusele kasuks?

- ♦ Rutiini käivitamiseks rakendusest kantakse üle võrgu vaid **üks** käsk.
- ♦ Täidetakse serverarvutis, mis tüüpiliselt kliendi arvutist palju **võimsam**.
- ♦ Rutiin on serveris **kompileeritud** kujul.

10.01.2018 Teema 4 123

Andmebaasid II 2017 © Erki Eessaar

Miks tuleb rutiin töökiirusele kasuks? (2)

- ♦ Mitme kasutaja poolt käivitamisel tuleb mällu lugeda vaid üks koopia – kui protseduur on juba jagatud puhvris (*shared pool*), saab selle täitmine kohe alata.
 - Väheneb **kettalt lugemiste** arv.
 - Pole vaja korduvalt kontrollida, kas rutiin on seisundis "**valid**".
 - Hoitakse kokku **mäluressursse**.

10.01.2018 Teema 4 124

Andmebaasid II 2017 © Erki Eessaar

Rutiinide muud eelised

- ♦ SQL laused **koonduvad** ühte kohta.
- ♦ Lihtsustab täienduste ja **paranduste** sisseviimist.
- ♦ Sarnase koodi **koondamine** parandab taaskasutuse võimalusi.
- ♦ Võimaldab kasutaja eest **peita** muutuseid andmebaasi struktuuris.

10.01.2018 Teema 4 125

Andmebaasid II 2017 © Erki Eessaar

Rutiinide muud eelised (2)

- ♦ Võimaldab kasutajatele anda **õiguse** käivitada rutiini, kuid mitte pöörduda otse tabelite poole.
- ♦ Aitab vähendada **SQL süstamise rünnaku** võimalusi (eeldusel, et rutiinis pole kasutatud ebaturvalisel viisil dünaamilist SQLi).
- ♦ Andmebaasisüsteem kontrollib ja võimaldab süsteemikataloogi põhjal kontrollida **kooskõllalisust** teiste andmebaasiobjektidega.

10.01.2018 Teema 4 126

Rutiinide muud eelised (3)

- ♦ Rutiinides saab *enamasti* kasutada vastava andmebaasisüsteemi SQL dialekti toetatud andmetüüpe, operaatoreid ja funktsioone.
- ♦ Andmekäitluskeele lausetes saab kasutada funktsioone.
- ♦ Rutiine saab koondada **pakettidesse** (Oracle näitel).

10.01.2018

Teema 4

127



Miks on paketid kasulikud? (Oracle näitel)

- ♦ Lihtsustab **õiguste** andmist – saab anda käivitamise õiguse kogu pakstile.
- ♦ On võimalik parandada rakenduse **jõudlust** – kõik pakettis sisalduvad rutiinid loetakse korraga mälli, kui pöördutakse neist ühe poole.
- ♦ Pakettis sisalduva **rutiini muutmine** (seni kuni ei muudeta paketi päises sisalduvat vastava rutiini spetsifikatsiooni) ei sunni Oraclet uuesti kompileerima seda rutiini välja kutsuvaid alamprogramme (erinevalt paketiväliste rutiinide muutmisest).

10.01.2018

Teema 4

128

Miks on paketid kasulikud? (Oracle näitel) (2)

- ♦ Erinevates pakettides võib olla sama **nimega** rutiine.
- ♦ Pakettis võib olla mitu ühe nime, kuid erinevate parameetritega rutiini – **ülelaadimine**.
 - **Polümorfism** – rutiin omab erinevates kontekstides erinevat tähendust või kasutusviisi.
- ♦ Paketiga võib siduda koodi, mis täidetakse paketi **esmakordsel** mälli lugemisel.

10.01.2018

Teema 4

129



Pakett ja tarkvaraarenduse parimad praktikad

- ♦ Madal sõltuvus (ingl *low coupling*):
 - Paketti kuuluvad rutiinid peaksid võimalikult **vähe lootma** teistes pakettides paiknevale rutiinidele.
- ♦ Kõrge kokkukuuluvus (ingl *high cohesion*):
 - Paketti kuuluvate rutiinide ülesanded (vastutused) peaksid olema üksteisega **tihedalt seotud**.
- ♦ Miks järgida?
 - Lihtsam aru saada, taaskasutada. Koos kasutatavad rutiinid loetakse koos mälli. Koos kasutatavatele rutiinidele lihtsam anda õiguseid. ...

10.01.2018

Teema 4

130

PL/SQL funktsioon SQL lauses – näide 1 (Oracle)

```

/*Loon Oracle (12c Enterprise Edition Release 1)
andmebaasis tabeli temperatuur andmete hoidmiseks.*/
CREATE TABLE Temp(temp NUMBER(9));

/*Lisan tabelisse testandmed – iga reas on juhuslikult
leitud väärtus vahemikus -100 ja 2000. Tabelisse lisan 77594 rida.*/
INSERT INTO Temp(temp)
SELECT dbms_random.value(-100,2000) num FROM all_objects;

/*Loon funktsiooni, mis teisendab Fahrenheiti temperatuuriskaalal oleva temperatuuri Celsiusi skaalale.*/
CREATE OR REPLACE FUNCTION f_Fahrenheit_Celsius(sisend IN NUMBER) RETURN NUMBER
DETERMINISTIC IS
  tulemus NUMBER(9,3);
BEGIN
  tulemus:=round(((sisend - 32) * 5 / 9),3);
  RETURN tulemus;
END f_Fahrenheit_Celsius;
/

/*Kontrollin tabeli kaudu käivit atariitikat.*/
EXEC dbms_stats.gather_table_stats(ownname=> 'C##TUD1', tablename=> 'TEMP',
cascade => true);

/*Käivitan SQL*Plus ajastu.*/
set timing on;

```

10.01.2018

Teema 4

131

PL/SQL funktsioon SQL lauses – näide 1 (Oracle)(2)

```

/*Päring, kus pöördutakse PL/SQL funktsiooni poole.*/
SELECT Count(*) arv FROM Temp
WHERE f_Fahrenheit_Celsius(temp)>100;

ARV
-----
66001
Elapsed: 00:00:00.33

/*Samaväärse tulemuse andev päring, kus pöördutakse ainult
süsteemi-definitsioonid funktsioonide poole.*/
SELECT Count(*) arv FROM Temp
WHERE round(((temp - 32) * 5 / 9),3)>100;

ARV
-----
66001
Elapsed: 00:00:00.11

```

10.01.2018

Teema 4

132

PL/SQL funktsioon SQL lauses – näide 2 (Oracle)

```
CREATE OR REPLACE FUNCTION f_oppeasutus_nimi
(id IN NUMBER) RETURN VARCHAR2
IS
  m_nimi VARCHAR2(100);
BEGIN
  SELECT nimi INTO m_nimi FROM Oppeasutus WHERE
    oppeasutus_id=id;
  RETURN m_nimi;
END f_oppeasutus_nimi;
/
```

10.01.2018

Teema 4

133

PL/SQL funktsioon SQL lauses – näide 2 (Oracle)(2)

```
SELECT oppekava_id, f_oppeasutus_nimi(oppeasutus_id)
       oppeasutus FROM Oppekava;
```

Elapsed: 00:00:00.38

```
SELECT Ok.oppekava_id, O.nimi oppeasutus
       FROM Oppekava Ok INNER JOIN Oppeasutus O ON
         O.oppeasutus_id=Ok.oppeasutus_id;
```

Elapsed: 00:00:00.20

Tabel *Oppeasutus* – 135 ridaTabel *Oppekava* – 2361 rida

10.01.2018

Teema 4

134

PL/SQL funktsioon SQL lauses – kokkuvõte (Oracle)

- Kui Oracle SQL lausest pöördutakse PL/SQL funktsiooni poole, siis toimub **kontekstkommutatsioon** (ingl *context switch*) SQL ja PL/SQL mootori vahel.
- "Kontekstkommutatsioon on arvuti keskprotsessori olekut (konteksti) salvestav ja taastav protsess, mis võimaldab mitmel protsessil ühiselt kasutada üht ja sama protsessorit" (<http://www.vallaste.ee/>).

10.01.2018

Teema 4

135

PL/SQL funktsioon SQL lauses – kokkuvõte (Oracle)(2)

- Kui SQL lause täitmise käigus tehakse palju **PL/SQL** keeles kirjutatud funktsiooni väljakutseid, siis see vähendab selle lause täitmise kiirust.
- Kui Oracle SQL lausest pöördutakse **süsteemi-definieeritud funktsiooni** poole, siis kontekstkommutatsiooni ei toimu ja võrreldes SQL lauses PL/SQL funktsiooni kasutamisega täidetakse lause kiiremini.
 - Süsteemi-definieeritud funktsioonid on realiseeritud C keeles. Selles keeles on kirjutatud ka suur osa Oracle andmebaasisüsteemist.

10.01.2018

Teema 4

136

Funktsiooni väljakutse PostgreSQL SQL lauses

- Kordasin temperatuuriteisenduse eksperimenti PostgreSQL andmebaasis.
- Kolm lahendust.
 - a) Päring kasutab ainult süsteemi-definieeritud funktsioone (need on kirjutatud C või SQL keeles)
 - b) Päring kasutab SQL keeles kirjutatud kasutaja-definieeritud funktsiooni
 - c) Päring kasutab PL/pgSQL keeles kirjutatud kasutaja-definieeritud funktsiooni
- Variandi c) korral võttis päringu täitmine **üle kuue korra** rohkem aega, kui variantide a) ja b) korral.
- Variantid a) ja b) olid umbes ühesuguse töökiirusega.

10.01.2018

Teema 4

137

Rutiinide kasutamise probleeme


- Andmebaasiserveris talletatud rutiinide väljakutsumine rakendustest üle arvutivõrgu on näide **kaugprotseduuride väljakutsete** kasutamisest (*remote procedure call*).
- Järgnev pole spetsiifiline rutiinidele, vaid on võimalik ka näiteks siis, kui rakendus teeks üle arvutivõrgu andmemuudatusi otse baastabelites.

10.01.2018

Teema 4

138

Andmebaasid II 2017 © Erki Eessaar



Rutiinide kasutamise probleeme (2)

- ◆ Kui arendajad ise midagi muud ei realiseeri, siis infovahetus on **sünkroonne** – klient kutsub rutiini välja ja kliendi tööjärg peatub kuni saab vastuse.
- ◆ Väljakutsuja ja väljakutsutava vahel **ennustamatu arvutivõrk**.

10.01.2018 Teema 4 139

Andmebaasid II 2017 © Erki Eessaar

Rutiinide kasutamise probleeme (3)

- ◆ Kui **vastust ei tule**, siis põhjus võib olla
 - käivitamise palve läks kaotsi või hilineb,
 - täitmine ebaõnnestus (rutiini viga),
 - täitmine ebaõnnestus või aeglane (serverarvuti viga, aga ka ebaotstarbekalt kodeeritud ja seega aeglaselt töötav rutiin),
 - tagastatud vastus läks kaotsi või hilineb.

10.01.2018 Teema 4 140

Andmebaasid II 2017 © Erki Eessaar

Rutiinide kasutamise probleeme (4)

- ◆ Erinevate põhjuste korral on vajalik erinev **reaktsioon**.
 - Näiteks kui õnnestunud olemi lisamise rutiini uuesti samade argumentidega välja kutsuda, siis võib tulemuseks olla olemi andmete korduv registreerimine.
 - Vältimiseks kasutage tabelis sisulisi võtmeid.
- ◆ **Täitmisaeg** sõltub ka arvutivõrgu seisust ja võib tänu sellele kõikuda.

10.01.2018 Teema 4 141

Andmebaasid II 2017 © Erki Eessaar

Rutiinide kasutamise probleeme (5)

- ◆ Kuna erinevad rakenduse loomise vahendid toetavad erinevat hulka **andmetüüpe**, siis võib olla vaja eri rakenduste jaoks olla vaja luua sama sisuga rutiine, mille sisendparameetrite tüübid on erinevad.

10.01.2018 Teema 4 142

Andmebaasid II 2017 © Erki Eessaar



SQL lausete optimeerimine

- ◆ Selleks, et SQL lause täidetaks kiiremini, peab vähenema:
 - plokkide arv, mida tuleb lugeda **kettalt** (ingl *physical reads*),
 - plokkide arv, mida tuleb lugeda **muutmälust** (ingl *logical reads*).

10.01.2018 Teema 4 143

Andmebaasid II 2017 © Erki Eessaar

Soovitusi programmeerijale

- ◆ Küsi vaid selliseid andmeid, mida tõesti kasutad.
 - **SELECT * FROM T ...** – halb stiil
 - Näiteks, kui tulevikus lisatakse tabelisse T uusi veerge, siis päring küsib andmeid ka nendest veergudest, kuigi rakendus neid ei vaja. Lisaks ka koodi lugeja ei näe, milliseid andmeid loetakse.
- ◆ Erineva süntaksiga kuid samaväärsete päringute täitmise aeg võib olla oluliselt erinev.
- ◆ Kasuta SQL lauseid (mis töötavad ridade hulkadega), selle asemel, et kirjutada protseduure, mis loevad ja töötlevad andmeid ühe rea kaupa.

10.01.2018 Teema 4 144

Andmebaasid II 2017 © Erki Eessaar

2361 rea tabelisse lisamine (2 lahendust) (Oracle)

```

CREATE OR REPLACE PROCEDURE load_oppekava IS
  CURSOR oppekava_kursor IS SELECT oppekava_id, nimetus FROM Oppekava;
  n_rec oppekava_kursor%ROWTYPE;
BEGIN
  --Kursori avamine
  OPEN oppekava_kursor;
  LOOP
    --Iga rea lisamine
    FETCH oppekava_kursor INTO n_rec;
    EXIT WHEN oppekava_kursor%NOTFOUND;
    INSERT INTO Oppekava_koopia(oppekava_id, nimetus)
      VALUES (n_rec.oppekava_id, Upper(n_rec.nimetus));
  END LOOP;
  --Kursori sulgemine
  CLOSE oppekava_kursor;
  RETURN;
END;
/
/*lisamine üksikute
ridade kaupa*/
/*ridade hulga lisamine*/

```

Elapsed: 00:00:00.57 Elapsed: 00:00:00.01

10.01.2018 Teema 4 145

Andmebaasid II 2017 © Erki Eessaar

Soovitusi programmeerijale (2)

- Eelista *staatilist* SQLi dünaamilisele SQLile (dünaamiline SQL – SQL lause pannakse kokku programmi töö käigus).
- Koonda *mitu* andmemuudatust ühte *transaktsiooni*.
- Kasuta andmebaasisüsteemi pakutavaid *paralleeltöö* võimalusi.

10.01.2018 Teema 4 146

Andmebaasid II 2017 © Erki Eessaar

Andmekäitluskeele lausete töötlemine Oracles

- Mida peab andmebaasisüsteem tegema lause täitmiseks?
 - Soft parse** – lause süntaktiline ja semantiline analüüs.
 - Hard parse** – lause süntaktiline ja semantiline analüüs, lause täitmiskava koostamine ja optimeerimine ning täitmiskava jagatud puhvriss laadimine.
- Andmekäitlusoperatsioonide töökiiruse parandamiseks peaks võimalikult väikese hulga lausete puhul toimuma *hard parse*.
 - Kui lause täitmiskava on *jagatud puhvriss*, siis Oracle saab seda taaskasutada!

10.01.2018 Teema 4 147

Andmebaasid II 2017 © Erki Eessaar

Bind variables (Oracle)

- CREATE OR REPLACE PROCEDURE pr_aine_oppimine_mitteakt(ak IN VARCHAR2) IS BEGIN UPDATE Oppimine SET on_aktuaalne=0 WHERE ainekood=ak; END pr_aine_oppimine_mitteakt; /
- ak** – bind variable
- Käivitades protseduuri korduvalt, saab andmebaasisüsteem taaskasutada lause täitmiskava
 - UPDATE Oppimine SET on_aktuaalne=0 WHERE ainekood=ak;
- Täitmiskava leitakse kasutades lause põhjal genereeritud *räsväärtust*
- Järgnevad laused loeb Oracle erinevateks lauseteks ning ühe täitmiskava ei saa teise puhul kasutada
 - SELECT * FROM Oppimine;
 - select * from oppimine;

10.01.2018 Teema 4 148

Andmebaasid II 2017 © Erki Eessaar

Staatiline SQL (Oracle)

```

begin
for i in 1..1000 loop
  insert into t1 values(i);
end loop;
commit;
end;
/

```

Staatilises SQL lauses pole bind variables kasutust vaja eraldi deklareerida. Antud näites on i bind variable.

hard parse – 1 kord

Elapsed: 00:00:00.13

10.01.2018 Teema 4 149

Andmebaasid II 2017 © Erki Eessaar

Dünaamiline SQL koos bind variables kasutamisega (Oracle)

```

begin
for i in 1..1000 loop
  execute immediate 'insert into t1 values(:x)'
    using i;
end loop;
commit;
end;
/

```

hard parse – 1 kord

Elapsed: 00:00:00.08

10.01.2018 Teema 4 150

Andmebaasid II 2017 © Erki Eessaar

Dünaamiline SQL ilma *bind variables* kasutamisetä (Oracle)

```

begin
for i in 1..1000 loop
    execute immediate 'insert into t1 values('||i||')';
end loop;
commit;
end;
/

```

Elapsed: **00:00:00.74**

hard parse – 1000 korda

10.01.2018 Teema 4 151

Andmebaasid II 2017 © Erki Eessaar

Staatiline SQL koos sagedase kinnitamisega (Oracle)

```

begin
for i in 1..1000 loop
    insert into t1 values(i);
    commit;
end loop;
end;
/

```

Elapsed: **00:00:00.18**

10.01.2018 Teema 4 152

Andmebaasid II 2017 © Erki Eessaar

Soovitusi programmeerijale (3)

- Vältige *varjatud* tüübiteisendusi.
- CREATE TABLE Tellimus (tellimuse_kood VARCHAR2(5) PRIMARY KEY);
- SELECT * FROM Tellimus WHERE tellimuse_kood=1;
- Tegelikult täidetakse:
 - SELECT * FROM Tellimus WHERE to_number(tellimuse_kood)=1;
 - Veerule tellimuse_kood loodud indeksit ei kasutata!

Näide Oracles

10.01.2018 Teema 4 153

Andmebaasid II 2017 © Erki Eessaar

Kitsendused aitavad ka parandada töökiirust

- CREATE TABLE Isik (synni_aeg DATE NOT NULL);
- CREATE INDEX idx_isik_synni_aeg ON Isik(synni_aeg);
- INSERT INTO Isik(synni_aeg) VALUES (to_date('1999-01-01 11:00','YYYY-MM-DD HH24:MI'));
- SELECT * FROM Isik WHERE synni_aeg=to_date('1999-01-01','YYYY-MM-DD');
- Päringu tulemus pole ühtegi rida!

Näide Oracles

10.01.2018 Teema 4 154

Andmebaasid II 2017 © Erki Eessaar

Kitsendused aitavad ka parandada töökiirust (2)

- SELECT * FROM Isik WHERE trunc(synni_aeg)=to_date('1999-01-01','YYYY-MM-DD');
- Päringu tulemus üks rida!
- Tänu trunc funktsiooni rakendamisele veerule synni_aeg ei kasuta andmebaasist süsteem sellise päringu täitmiseks kunagi indeksit idx_isik_synni_aeg.
- Lahendus.
 - Jõustada andmebaasis kitsendus, et vältida hiljem ebasobivat päringut – sünniaja kellaeg peab olema 00:00.
 - ALTER TABLE Isik ADD CONSTRAINT chk_isik_synni_aeg CHECK (trunc(synni_aeg)=synni_aeg);

10.01.2018 Teema 4 155

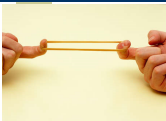
Andmebaasid II 2017 © Erki Eessaar

Skaleerimine e mastaapimine



- Skaleeritavus – võime kasvada järk-järgult.
- Skaleerimine – süsteemi suurendamine.
- Riistvara skaleerimine võib olla:
 - vertikaalne – suurendatakse serverarvuti võimsust (protsessor, muutmälu) ja kettamahtu,
 - horisontaalne – serverarvutite kobarasse lisatakse uusi arvuteid.
 - Saab kasutada väiksemaid tarbeservereid, suure ja kalli eriotstarbelise serveri asemel.

10.01.2018 Teema 4 156



Elastsus

- ♦ **Elastsus** – võime olukorraga dündamiliselt kohaneda. Kui **nõudlus** suureneb, siis *kasvada*, kui nõudlus väheneb, siis *kahaneda*.
 - Keerukam kui ainult skaleerimine – andmete korral, kuidas dündamiliselt andmed väiksemale ketaste või serverite hulgale tagasi kokku tõmmata.

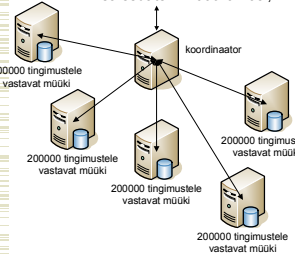
10.01.2018

Teema 4

157

Shared Nothing Partitioning (SNP)

```
SELECT Sum(sales) AS total FROM
Sale WHERE salesdate < '2008-02-04'
AND salesdate > '2009-04-05';
```



- ♦ Igal *serverarvutil* (serveril) oma andmed.
- ♦ Serverid suhtlevad kiire arvutivõrgu vahendusel.
- ♦ *Koordinaator* võtab vastu päringu, edastab teistele serveritele.
- ♦ Iga server leiab tulemise *oma andmete* põhjal.
- ♦ Tulemused edastatakse koordinaatorile, mis paneb nendest kokku lõpptulemuse.
- ♦ Andmete salvestamisel leitakse nende asukoht (server) *räsifunktsiooni* ja vastavustabeli abil.

10.01.2018

Teema 4

158

SNP eeliseid

- ♦ Serverite **paralleeltöö** ülesande lahendamisel.
- ♦ Hea **horisontaalne skaleeritavus**.
- ♦ Andmebaas jaotunud serverite vahel väiksemateks alamosadeks, mida saab **paindlikult** ja **kiiremine** hallata (nt indeksi loomine võtab vähem aega).
- ♦ Sobilik arhitektuur **analüüsi-** ja **otsustussüsteemidele**.

10.01.2018

Teema 4

159

SNP probleeme

- ♦ **Keerukam disainida** kui ühe serveriga süsteeme.
 - Kuidas genereerida võtmeväärtuseid?
 - Kuidas leida server, kuhu andmed paigutada?
 - Näiteks reas olevate väärtuste alusel arvutatud räsiväärtuse järgi.
 - Millistest andmetest teha koopia kõigisse serveritesse?
 - Näiteks klassifikaatorid.
- ♦ Onlain-tehingutöötlemise süsteemid töötavad efektiivselt vaid siis, kui pöörduvad otse vajalike andmetega serveri poole. See suurendab rakenduste loomise keerukust.

10.01.2018

Teema 4

160

Andmebaasisüsteemid ja SNP

- ♦ Paljud **NoSQL** süsteemid on loodud spetsiaalselt tööks sellisel arhitektuuril.
 - Sellised süsteemid salvestavad seotud andmed *agregaatidena* (nt JSON dokumendid), mida on hea erinevate serverite vahel jagada.
- ♦ Samas leidub ka SQL-andmebaasisüsteeme, mis on spetsiaalselt sellise arhitektuuri jaoks *timmitud*.
 - PostgreSQL-XL, Timescale

10.01.2018

Teema 4

161

Andmebaasisüsteemid ja SNP – Pinterest näide

- ♦ <https://engineering.pinterest.com/blog/sharding-pinterest-how-we-scaled-our-mysql-fleet/>
- ♦ Suur hulk servereid, igas üks või mitu **MySQL** andmebaasi
 - Pin: <https://www.pinterest.com/pin/241294492511762325/>
 - Pini ID'sse on kodeeritud andmebaasi, objekti tüübi (tabeli) ja rea identifikaator.
 - Süsteemis info andmebaaside serveris paiknemise kohta.
 - Selle info alusel koostab tarkvara **SELECT** lause.
 - Rakendus pöördub otse vajalike andmetega serveri poole.

10.01.2018

Teema 4

162

Andmebaasid II 2017 © Erki Eessaar

Töökiiruse parandamisest Oracles

- ♦ Rodionov, D., 2015. *Andmebaasi optimeerimine ning selles toimuvate operatsioonide jõudluse analüüs ühe Oracle andmebaasi näitel*. Bakalaureusetöö. TTÜ Informaatikainstituut. [WWW] <https://digi.lib.ttu.ee/i/?3433>

10.01.2018 Teema 4 163

Andmebaasid II 2017 © Erki Eessaar

Töökiiruse parandamisest Oracles (2)

- ♦ Kaks sama andmebaasi realiseerimist
 - Tavaline: "tavalised" baastabelid, indeksid, vaated
 - Optimeeritud: indeks-orienditud ja klastrisse koondatud tabelid, tabelite sektsioonideks jagamine, lokaalsed indeksid ning hetktõmmised
- ♦ 8 päringut, 5 andmemuudatust
- ♦ Oracle Database 12.1 Enterprise Edition

10.01.2018 Teema 4 164

Andmebaasid II 2017 © Erki Eessaar

Töökiiruse parandamisest Oracles (3)

- ♦ Optimeeritud disain
 - ♦ *parandas* üldiselt **päringute** täitmise kiirust
 - ♦ 5 päringut 8-st olid optimeeritud disaini korral kiiremad
 - ♦ *aeglustas* üldiselt **andmemuudatusi**
 - ♦ 3 andmemuudatust 5-st olid optimeeritud disaini korral aeglasemad
 - ♦ *suurendas* **andmemahte** 2.7 korda.

10.01.2018 Teema 4 165

Andmebaasid II 2017 © Erki Eessaar

Mõned vead töökiiruse optimeerimisel

- ♦ Alustatakse süsteemiarenduse **hilises** faasis.
- ♦ Mõeldakse ainult **riistvara** parandamisele.
- ♦ **Lemmikidee** rakendamine igas olukorras.
 - Näiteks igale veerule indeks ("kui ma olen haamer, siis kõik probleemid paistavad naeltena").
- ♦ Töökiiruse parandamiseks mõeldud lahenduste **mitte testimine**.
- ♦ Andmebaasi administraatori ja arendajate vastastikused **süüdistused**.

10.01.2018 Teema 4 166

Andmebaasid II 2017 © Erki Eessaar

Andmebaasi tegelik loomine

- ♦ Disaini käigus loodud mudelite põhjal käsufailide genereerimine.
- ♦ Käsufailide käivitamine.
 - *Koskstiilis* arenduse korral luuakse sellega korraga terve andmebaas.
 - *Iteratiivse* arenduse korral lisatakse täiendusi olemasolevasse andmebaasi.
 - Peaks olemasolevate andmebaasiobjektide kasutamist võimalikult vähe häirima.

10.01.2018 Teema 4 167

Andmebaasid II 2017 © Erki Eessaar

Käsufailide käivitamisest PostgreSQLis

- ♦ Andmekirjelduskeele laused saab koondada üheks loogiliseks tervikuks (transaktsiooni).
- ♦ CREATE, ALTER ja DROP lausetes IF EXISTS või IF NOT EXISTS tingimused.
 - Vältib veateateid, ei katkesta transaktsiooni.
- ♦ CREATE **OR REPLACE** FUNCTION / VIEW ...
 - Asendab kehandi, kui päis/veerud ei muutu.

10.01.2018 Teema 4 168

Käsufailide käivitamisest PostgreSQLis (2)

- ♦ Mõningate muudatuste tegemiseks tuleb andmebaasiobjekt kustutada ja uuesti luua. Teiste muudatuste jaoks on võimalik kasutada ALTER lauseid.
 - Näiteks ei saa ALTER lausega muuta domeeni baastüüpi, ega vaate alampäringut.

10.01.2018

Teema 4

169

Käsufailide käivitamisest PostgreSQLis (3)

- ♦ Tabelite struktuuri muutmine või indeksite muutmine lukustab tabelleid ja blokeerib samal ajal andmete lugemist/muutmist – ning vastupidi.
- ♦ Andmebaas tuleb hooldustöödeks sulgeda.
 - Üks põhjus, miks on kasulikud andmebaasi disainid, mille korral ei tule uute andmete registreerimise hakkamiseks muuta olemasolevate tabelite struktuuri.

10.01.2018

Teema 4

170

Olemasolevate CASE vahendite puudus

- ♦ Piiratud võimalused **kitsenduste disainiks** ja nende jõustamise koodi **genereerimiseks**.
- ♦ Andmebaasi disainer peab töötama **madalal abstraktsioonitasemel** (kirjeldama triggerite protseduure), selle asemel, et deklareerida kitsenduse olemasolu ning lasta süsteemil genereerida selle jõustamise kood.
 - Projekt, mis üritab olukorda leevendada: http://staff.ttu.ee/~eessaar/SQL_profile/

10.01.2018

Teema 4

171



Andmete uude süsteemi ülekandmine / andmesiire

- ♦ Kui andmebaas ei teki tühjale kohale, siis tuleb uude andmebaasi kanda andmed vana(de)st andmebaasi(de)st.
- ♦ Toimub **ülemineku** käigus.
- ♦ Kontseptuaalsed andmemudelid aitavad aru saada, mis on andmete *tähendus* vanas ja uues baasis.
- ♦ Järgnev käsitus SQL-andmebaaside kohta.

10.01.2018

Teema 4

172

Probleemid andmesiirdel

- ♦ Puuduvad vana või uue andmebaasi **mudelid**.
- ♦ Uue ja vana andmebaasi erinev **ulatus** ja **struktuur**.
- ♦ Andmeallikatest, kust andmed üle kanda, on **korduvad** ja **vastuolulised** andmed.
- ♦ Uues andmebaasis võib olla rohkem **kitsendusi** kui vanas.
- ♦ **Mittesoovitavate väärtuste** genereerimine väljadesse seoses andmebaasi triggerite käivitumisega või vaikimisi väärtustega.

10.01.2018

Teema 4

173

Sisukate definitsioonidega kontseptuaalne andmemudel

- ♦ Abistab *andmete integratsiooni* protsessis – andmesiire selle üks erijuhus.
 - Näiteks tuleb välja, et vana süsteemi tabel *Subjekt* ja uue süsteemi tabel *Isik* sisaldavad põhimõtteliselt samu andmeid. Teisalt tuleb välja, et vana andmebaasi tabeli *Subjekt* veerus *pikkus* on isiku pikkus sentimeetrites, samas kui uue andmebaasi tabeli *Isik* veerus *pikkus* on tema maksimaalse lubatud ooteaja pikkus minutites.

10.01.2018

Teema 4

174

Vigadega toimunud andmesiirde näide

- Eesti Energia sõlmis lepingu surnuga (tarbija24.ee, 12.01.2013 08:56)
- Vigaselt sisestatud isikukoodi tõttu pidi Tallinnas Nõmmel elav 76-aastane Illi Lill sõlmima elektrilepingu Eesti Energiaga oma 2000. aastal surnud abikaasa nimele. **Viga sai ilmselt alguse mullu septembris, mil toimus võrguettevõtete süsteemis andmete ülekandmine.**
- Eleringi kommunikatsioonijuht Ain Köster nentis, et seesuguseid juhtumeid on veel. «Viimasel päeval on välja tulnud veel juhtumeid vale isikukoodiga. Inimestel on samasugune probleem ja nad ei saa sellega midagi teha,» märkis ta.
- Lill soovis 29. novembril sõlmida endale internetis soovitud elektrikpaketti, kuid veebilehel tuli ette kiri «Teil ei ole liitumiskohta». «Mõtlesin, milles nüüd asi on, kogu aeg olen arved korralikult ära maksnud,» rääkis naine, kes käis avaneva elektrituru kohta kuulamas ka loengut Nõmme kultuurikeskuses.
- <http://www.tarbija24.ee/1100950/est-energia-solmis-lepingu-surnuga>

10.01.2018

Teema 4

175

Näide (PostgreSQL)

- CREATE TABLE Isik (isik_id SERIAL CONSTRAINT isik_pk PRIMARY KEY, eesnimi VARCHAR(1000) NOT NULL, perenimi VARCHAR (1000) NOT NULL);
- INSERT INTO Isik VALUES (1, 'Andres','Mets');
- INSERT INTO Isik VALUES (2, 'Teet','Tee');
- ...
- INSERT INTO Isik VALUES (500, 'Kaarel','Kask');
- INSERT INTO Isik(eesnimi, perenimi) VALUES ('Lembit','Mets');
- ERROR: duplicate key value violates unique constraint "isik_pk"
- DETAIL: Key (isik_id)=(1) already exists.

10.01.2018

Teema 4

176

Andmesiire – andmete ülekandmise järjekord

- Kõigepealt ilma välisvõtmeteta baastabelid (edaspidi *tabelid*).
- Seejärel (tsüklis) ainult täidetud tabelitele viitavate välisvõtmetega tabelid seni, kuni kõik tabelid on täidetud.

10.01.2018

Teema 4

177

Andmesiire – probleemid välisvõtmega

- Probleem:** Tabelis olev välisvõti viitab sama tabeli kandidaatvõtmele.
 - Töötaja (isikukood, perenimi, ülemus)
 - Primaarvõti (isikukood)
 - Välisvõti (ülemus) Viitab Isik (isikukood)
- Võimalik lahendus:** Andmesiirde ajaks vastav välisvõtme kitsendus eemaldada ja peale siiret tagasi lisada.

10.01.2018

Teema 4

178

Andmesiire – probleemid välisvõtmega (2)

- Võimalik lahendus:** Kui välisvõtme veerg on *mittekohustuslik*, siis kõigepealt kanda andmed mitte-välisvõtme veergudesse ja seejärel eraldi lausega välisvõtme veergu.
- Võimalik lahendus:** Välisvõtme kitsenduse deklareerimine *deferred* kitsendusena, mille kontroll lükatakse transaktsiooni lõppu.

10.01.2018

Teema 4

179

Andmesiire – ajalised parameetrid

- Staatilisemaid** (aeglaselt muutuvaid) andmeid võib hakata üle kandma juba paar nädalat enne tegelikku planeeritavat üleminekut.
 - Klassifikaatorid, põhiaandmed (*master data*).
- Dünaamilisemaid** (sagedasti muutuvad/täienevad) andmeid (näiteks andmed protsesside/sündmuste kohta) tuleks üle kanda vahetult enne uue süsteemi tegelikku tööle panemist, kasutades selleks aega, mil andmeid ei uuendata.
 - Otsustada, kui vanu andmeid on vaja üle kanda.
- Kanda üle andmed ja alles siis luua **indeksid**.

10.01.2018

Teema 4

180

Andmebaasid II 2017 © Erki Eessaar

Klassifikaatorid

- Klassifikaatorid on mistahes andmed, mida kasutatakse andmebaasis teiste andmete liigitamiseks või andmete seostamiseks väljaspool organisatsiooni vastutusalala olevates andmebaasides olevate andmetega.
 - ISO 4217 Currency Codes.** Nt Euro alfabeetiline kood on EUR ja numbriline kood 978.
 - ISO 3166 Country Codes.** Nt Eesti puhul on riigi nimi "Estonia" ja riigi kood "EE".
 - ISO/IEC 5218 Information technology -- Codes for the representation of human sexes.** (0, 1, 2, 9)

10.01.2018 Teema 4 181

Andmebaasid II 2017 © Erki Eessaar

Andmesiire – ajalised parameetrid (2)

- Kõiki andmeid ei saa ilma töötlemata üle kanda.
- Üheetapiline** ülekandmine.
 - Kanda andmed üle neid vajadusel käigupealt teisendades.
- Kaheetapiline** ülekandmine.
 - Kõigepealt kanda üle töötlemist mitte vajavad andmed.
 - Osta või kirjutada konverteerimisprogrammid ning kanda üle töötlemist vajavad andmed.

10.01.2018 Teema 4 182

Andmebaasid II 2017 © Erki Eessaar

Andmesiire – meetodid ja vahendid

- Laadida** olemasoleva andmebaasi baastabelid uude andmebaasi ja kirjutada programm andmete ülekandmiseks.
 - Need tabelid võiksid olla eraldi skeemis. Siis on neid ka lihtne skeemi kustutamise abil korrigeerida.
- Hankida spetsiaalne **konverteerimisprogramm** andmete ülekandmiseks.
 - Mõttekam, kui andmesiiret tuleb läbi viia korduvalt.
 - Näide: REVER

10.01.2018 Teema 4 183

Andmebaasid II 2017 © Erki Eessaar

Mudelipõhine andmesiirde vahend

10.01.2018 Teema 4 184

Andmebaasid II 2017 © Erki Eessaar

Andmesiirde vahendeid Oracle näitel

10.01.2018 Teema 4 185

Andmebaasid II 2017 © Erki Eessaar

Andmesiirde näide – vanad ja uued tabelid

<p><i>Isik1</i>(isikukood, perenimi, aadress, telefon, kommentaarA, kommentaarB) Primaarvõti (isikukood)</p> <p><i>Auto1</i>(omanik, auto_reg_nr, mark, kommentaar) Primaarvõti (auto_reg_nr) Välisvõti (omanik) Viitab Isik2(isik2_id);</p> <p><i>Kommentaari2</i>(isik, tekst, tüüp) Primaarvõti(isik, tüüp) Välisvõti (isik) Viitab Isik2(isik2_id);</p> <p>Vanad tabelid</p>	<p><i>Isik2</i>(isik_id, perenimi, aadress, telefon, laadimise_aeg) Primaarvõti (isik_id)</p> <p><i>Auto2</i>(omanik, auto_reg_nr, kommentaar) Primaarvõti (auto_reg_nr) Välisvõti (omanik) Viitab Isik2(isik2_id);</p> <p><i>Kommentaari2</i>(isik, tekst, tüüp) Primaarvõti(isik, tüüp) Välisvõti (isik) Viitab Isik2(isik2_id);</p> <p>Uued tabelid</p>
--	--

10.01.2018 Teema 4 186

Andmesiirde näide (PostgreSQL baasil) – ettevalmistus

- ♦ Loo uues andmebaasis vana andmebaasi struktuurile vastavad tabelid ja kanna andmed sinna üle (need võiks panna eraldi *skeemi*, kuid näite lihtsustamiseks seda hetkel ei tehta).

```
START TRANSACTION;
ALTER TABLE Isik2 ADD COLUMN isikukood
CHAR(11);
```

10.01.2018

Teema 4

187

Andmesiirde näide – laadimine

```
INSERT INTO Isik2(isikukood, perenimi, aadress, telefon, laadimise_aeg)
SELECT isikukood,
/*Uues andmebaasis on perenimi kohustuslik, kuid vanas andmebaasis ei
olnud. Kui perenimi määramata, siis väärtus 'N/A'.*/
Coalesce(perenimi,'N/A'),
aadress,
/*Uues andmebaasis on telefoni nr täisarvu tüüpi vanas aga tekstitüüpi.
Enne tüübiteendust tuleb telefoninumbri kustutada
tühikud ja "+" märgid.*/
Cast(Translate(telefon,'+ ','') AS integer),
CURRENT_TIMESTAMP(0)
FROM Isik1;
```

10.01.2018

Teema 4

188

Andmesiirde näide – laadimine (2)

```
INSERT INTO Auto2(omanik, auto_reg_nr, kommentaar)
/*Vanas andmebaasis ei kontrollitud, kas auto numbris on
suurtähed või mitte. Uues andmebaasis peab auto
registreerimisnumber olema suurtähtedega.*/
SELECT Isik2.isik2_id, Upper(Auto1.auto_reg_nr),
Auto1.kommentaar || ' ' || Auto1.mark as kom
/*Auto margi andmed lähevad kommentaari veergu.*/
FROM Isik2 INNER JOIN Auto1 ON
Isik2.isikukood=Auto1.omanik;
```

10.01.2018

Teema 4

189

Andmesiirde näide – laadimine (3)

```
INSERT INTO Kommentaar2(isik, tekst, tüüp)
SELECT Isik2.isik2_id, Isik1.kommentaarA, 'A'
FROM Isik2, Isik1
WHERE Isik1.isikukood=Isik2.isikukood AND
Isik1.kommentaarA IS NOT NULL;
```

10.01.2018

Teema 4

190

Andmesiirde näide – laadimine (4)

```
INSERT INTO Kommentaar2(isik, tekst, tüüp)
SELECT Isik2.isik2_id, Isik1.kommentaarB, 'B'
FROM Isik2, Isik1
WHERE Isik1.isikukood=Isik2.isikukood AND
Isik1.kommentaarB IS NOT NULL;
```

10.01.2018

Teema 4

191

Andmesiirde näide – abistruktuuride eemaldamine

```
ALTER TABLE Isik2 DROP COLUMN isikukood;
/*Transaktsiooni lõpetamine*/
COMMIT;
ANALYZE; /*Statistika värskendamine*/
```

- ♦ Andmesiirde kiirendamiseks võib kaaluda andmesiirde sihttabelites *indeksite* kustutamist enne andmesiiret ning uuesti loomist peale andmesiiret (kiirem kui indeksite jooksev värskendamine).

10.01.2018

Teema 4

192

Andmebaasid II 2017 © Erki Eessaar




Süsteemi (sealhulgas andmebaasi) käikuandmise strateegiad

- ♦ **Suur pauk (nt ID-kaarti sertifikaatide värskendamise tarkvara 2017. a. sügisel)**
 - Kõik kasutajad hakkavad korraga uut süsteemi kasutama.
 - Eelnevalt peab olema süsteemi tööd suure töökoormuse juures piisavalt testitud.
 - Vana süsteem pannakse kinni.
 - Ei teki probleeme seoses vana ja uue süsteemi vahel andmete sünkroniseerimisega.

10.01.2018 Teema 4 193

Andmebaasid II 2017 © Erki Eessaar




Süsteemi (sealhulgas andmebaasi) käikuandmise strateegiad (2)

- ♦ **Tilkumine (nt TTÜ ÖIS2)**
 - Kasutajad lähevad järk-järgult uue süsteemi kasutamisele üle. Süsteemi koormus kasvab järk-järgult.
 - Vana ja uus süsteem töötavad mõnda aega paralleelselt.
 - Vaja on sünkroniseerida andmeid vana ja uue süsteemi vahel.
 - Kui uues süsteemis jõudluse või muud probleemid, siis mõjutab vaid uue süsteemi kasutajaid.
 - Sisuliselt on varased uue süsteemi kasutajad katsejāneste ja testijate rollis.

10.01.2018 Teema 4 194

Andmebaasid II 2017 © Erki Eessaar



Andmetest varukoopiate tegemise strateegiad

- ♦ **Füüsiline varukoopiate tegemine.**
 - Koopiate tegemine failidest ilma logifailide arhiveerimiseta (külm, offline) – andmebaasisüsteem peab olema seisatud.
 - Taastamine võimalik koopia tegemise seisuga.
 - Koopiate tegemine failidest koos logifailide arhiveerimisega (kuum, online) – andmebaasisüsteemi ei pea seiskama.
 - Taastamine võimalik ka mingi ajahetke seisuga, mis jääb koopia tegemise ja tõrke vahele.

10.01.2018 Teema 4 195

Andmebaasid II 2017 © Erki Eessaar

Andmetest varukoopiate tegemise strateegiad (2)

- ♦ **Loogiline varukoopiate tegemine.**
 - Andmete eksportimine (arhiveerimiseks) ja importimine (taastamiseks).
 - Taastamine võimalik koopia tegemise seisuga.
- ♦ **Andmete replikeerimine (тираžeerimine, kopeerimine, dubleerimine) (vt teema 12).**
 - Põhiandmebaasis toimuvate andmemuudatuste sünkroonne või asünkroonne ülekanne varuandmebaasi.

10.01.2018 Teema 4 196

Andmebaasid II 2017 © Erki Eessaar

Tõrgete tüübid (Oracle)

- ♦ **SQL lause tõrge**
 - *Lahendus:* Automaatne transaktsiooni alguse eelse olukorra taastamine
- ♦ **Kasutaja tehtud viga**
 - *Lahendus:* Tagasiulatuva päring (*flashback query*)
- ♦ **Serveriprotsessi tõrge**
 - *Lahendus:* Taustprotsessid rullivad lõpetamata transaktsioonid automaatselt tagasi
- ♦ **Eksemplari tõrge**
- ♦ **Meediumi tõrge**

10.01.2018 Teema 4 197

Andmebaasid II 2017 © Erki Eessaar

Varukoopiate tegemine – varukoopiate tüübid (Oracle)

- ♦ **Terviklik varukoopia.**
 - Koopia tegemise ajaks on andmebaas suletud.
- ♦ **Mitteterviklik varukoopia.**
 - Koopia tegemise ajal on andmebaas kasutuses (ainuvõimalik variant 24/7 süsteemide puhul).
 - Andmebaasisüsteem peab töötama ARCHIVELOG režiimis – logifailidest tehakse enne nende ülekirjutamist koopia, mis paigutatakse ettenähtud asukohta.
 - Andmebaasisüsteem kirjutab vanad logifailid üle.

10.01.2018 Teema 4 198

Varukoopiate tegemine – varukoopiate tegemise vahendid (Oracle)

- ♦ RMAN (Recovery Manager).
 - andmete taastamine ploki, tabeli, tabeliruumi, andmefaili, andmebaasi tasemel; tabeliruumi taastamine mingi ajahetke seisuga; varukoopia pakkimine; varukoopia krüpteerimine, paralleeltöö pakkimisel; andmebaasi kloni loomine üle võrgu; ...
- ♦ Operatsioonisüsteemi utiliitid.
 - ALTER TABLESPACE ... BEGIN BACKUP;
 - Failide kopeerimine;
 - ALTER TABLESPACE ... END BACKUP;
- ♦ Export ja Import utiliitid (Data Pump Oracle 12c).

Varukoopiate tegemine (PostgreSQL)

- ♦ Terviklik füüsiline varukoopia.
 - Varukoopia tegemine andmefailidest koos andmebaasisüsteemi seiskamisega.
- ♦ Mitteterviklik füüsiline varukoopia.
 - Varukoopia tegemine andmefailidest ja sejärel ainult logifailide arhiveerimine (ilma andmebaasisüsteemi seiskamiseta).
- ♦ Loogiline varukoopia: pg_dump ja pg_dumpall
 - pg_dump -C -f varukoopia_faili_nimi andmebaasi_nimi
 - pg_dump -C -f t990999.sql t990999

Üksikute tabelite sisu hetkeseisu talletamine (PostgreSQL)

- ♦ COPY ... TO – üksikute tabelite sisu hetkeseisu talletamine
 - COPY Aine TO '/tmp/aine_koopia.out';
- ♦ COPY ... FROM – andmete kopeerimine tabelisse
 - COPY Aine FROM '/tmp/aine_koopia.out';
 - Andmete kopeerimisel tabelisse rakenduvad INSERT trigerid, aga ei rakendu INSERT reeglid
- ♦ Loogilise varukoopia erijuhus.

Võimalik varukoopiate tegemise strateegia

- ♦ Kasutada failidest koopiate tegemist koos logifailide arhiveerimisega.
- ♦ *Sagedus*: keskmiselt üks kord ööpäevas (tööaja väliselt) kõigist andmebaasi failidest.
- ♦ Dubleerida andmebaasi juhtfail ja logifailid ning parameetrite fail. Duplikaadid paigutada erinevatele kettaseadmetele.

Võimalik varukoopiate tegemise strateegia (2)

- ♦ Hoida samast seisust mitut koopiat erinevatel seadmetel (*kõiki mune ei panda ühte korvi*):
 - vabal kõvakettal (kiireks taastamiseks),
 - aeglasemal arhiveerimiseseadmel,
 - pilves (nt Oracle pakub integratsiooni Amazoni *Simple Storage Service* (S3) teenusega).
- ♦ Kriitilistes süsteemides – *kuumvaru*.
 - Andmebaas, kuhu kantakse sünkroonselt või väikese viiteajaga üle kõik andmemuudatused ja millega saab asendada rivist välja langenud andmebaasi.

Magnetlintsalvesti (lintmäluseadme e lindiajam)

- ♦ Järjestikpöördusega seadmed.
 - Vajaliku ploki lugemiseks tuleb enne läbi lugeda kõik eelnevad plokiid.
- ♦ Aeglased kasutamaks üldotstarbelise mäluseadmena.
- ♦ Odavaim andmekandja suurte andmemahtude jaoks.
- ♦ Näiteid (<http://www.lascon.co.uk/Tape-drive-comparisons.php>):
 - Mahutavus: 75GB–8TB.
 - Andmete edastamise kiirus: 246–1256 GB andmeid tunnis.

Reeglid varukoopiate tegemisel

- ◆ Andmebaasis **pole ainsad** andmed.
- ◆ Optimeeri **väikese hulga** andmete taastamise jaoks.
- ◆ Taastamise regulaarne **testimine**.
- ◆ Ära ületa tootja määratud andmekandjale **kirjutamiste arvu**.
- ◆ Andmekandjad **vananevad**.

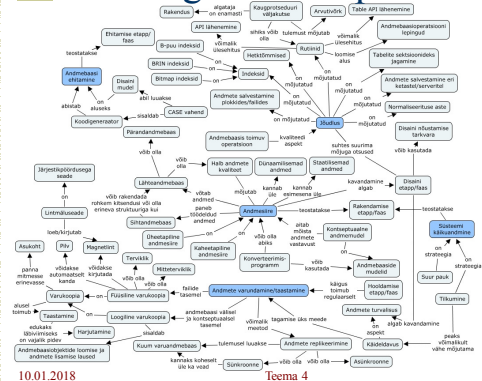
Kuidas säilitada varukoopiat pikaajaliselt?

- ♦ Koos andmekandjatega säilita **seadmed** ning **programmid** andmekandjatel olevate varukoopiate **lugemiseks**.
- ♦ Kanna varukoopiaid vanema põlvkonna andmekandjalt **uuema põlvkonna** andmekandjatele ning teisendada varukoopiaid nii, et ka uue põlvkonna programmid oskaksid neid lugeda.

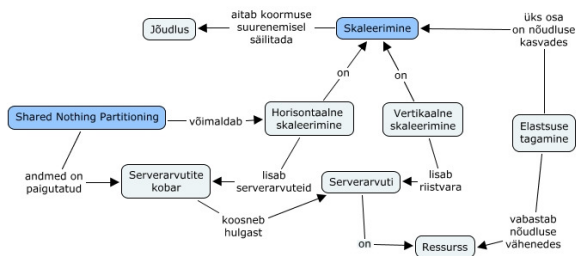
Varukoopiate turvalisuse tagamine

- ◆ Võrgus liikuvate andmete **krüpteerimine**.
- ◆ Piira füüsilist **juurdepääsu** andmekandjatele.
- ◆ Andmekandjad tuleb **märgistada**, et kogemata mitte üle kirjutada.
- ◆ Andmete **krüpteerimine** andmekandjal. Taastamisel parooli küsimine.
- ◆ Enne andmekandja ära viskamist muuda see **loetamatuks**.

Mõningad teema põhimõisted



Mõningad teema põhimõisted – skaleerimine



Mõningad teema põhimõisted – andmebaasi käikuandmine

