Füüsilise disain – andmebaasisüsteemide võimaluste võrdlus

Teema 2

Füüsiline disain

- Andmebaasi füüsilise disaini käigus toimub andmebaasi kavandamine **konkreetset andmebaasisüsteemi** silmas pidades.
- Erinevate andmebaasisüsteemide pakutav funktsionaalsus erineb detailides üksteisest küllaltki palju (isegi kui põhinevad samal andmemudelil).

Füüsiline disain (2)

- Järgnevalt võrreldakse *PostgreSQL (10)* ja *Oracle (12c Enterprise Edition Release 1)* andmebaasisüsteeme andmebaasi programmeerija seisukohast.
- Erinevuste hulga nägemine aitab muuhulgas mõista **andmebaasi migreerimise** (andmebaasisüsteemi vahetuse) keerukust.

Identifikaatorid

- Identifitseerivad andmebaasiobjekte.
- Sageli kasutatakse terminit "nimi".
- Identifikaator võib koosneda mitmest osast:
  - Näide: *"skeemi nimi"."tabeli nimi"."veeru nimi"*
  - SELECT public.koristaja.perenimi FROM public.koristaja;

Identifikaatorid – pikkus

- PostgreSQL.
  - Vaikimisi kuni 63 baiti.
  - Konfiguratsioonifaili muutes saab suurendada.
- Oracle.
  - Enamasti kuni 30 baiti.
  - Erandid.
    - Andmebaasi identifikaator kuni 8 baiti.
    - Andmebaasi lingi identifikaator kuni 128 baiti.
- Mitmeosalise identifikaatori korral kehtivad need reeglid iga üksiku alamosa kohta.

Identifikaatorid (2)

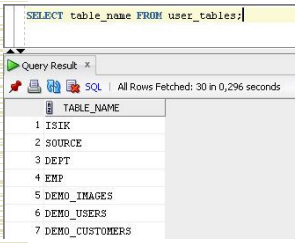
- PostgreSQL
  - CREATE TABLE t0123456789012345678901234567890123456789012345678901234567890123456789(id integer PRIMARY KEY);
  - Andmebaasisüsteem *lühendab identifikaatorit* ja loob tabeli t0123456789012345678901234567890123456789012345678901234567890123456789 01
- Oracle
  - CREATE TABLE T0123456789012345678901234567890(id integer PRIMARY KEY);
  - ERROR at line 1:
  - ORA-00972: identifier is too long

Piiritletud ja regulaarsed identifikaatorid

- PostgreSQL – regulaarsed identifikaatorid salvestatakse süsteemikataloogis *väiketähtedega*.
  - Isik, isik ja "isik" on samaväärsed.
  - isik ja "ISIK" ei ole samaväärsed.
- Oracle – regulaarsed identifikaatorid salvestatakse süsteemikataloogis *suurtähtedega*.
  - Isik, isik ja "ISIK" on samaväärsed.
  - isik ja "isik" ei ole samaväärsed.

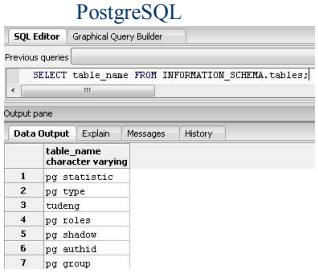
Päringud süsteemikataloogist

Oracle



Päring *Oracle SQL Developer* vahendis

PostgreSQL



Päring *pgAdmin* vahendis

Regulaarses identifikaatoris kasutatavad märgid

- PostgreSQL.
  - Peab algama tähega või alakriipsuga (_).
  - Võib sisaldada tähti, alakriipsu, numbreid (0–9) või dollarimärki ($).
- Oracle.
  - Peab algama tähega, mis kuulub andmebaasis kasutatavasse märgistikku (*character set*).
  - Võib sisaldada tähti, numbreid (0–9), alakriipsu (_), dollarimärki ($), trelle (#).
    - Andmebaasi linkide identifikaatorid ka “at” märke (@) ja punkte.

Identifikaatorid

- PostgreSQL ja Oracle lubavad identifikaatorites "täpitähti" (õäöüÕÄÖÜ).
- Kõik rakendused ei pruugi nendega hakkama saada.
- Kindlam on nende kasutamisest identifikaatorites loobuda või teha seda vähemasti andmebaasi avaliku liidese moodustavate elementide puhul.

Andmetüüp e tüüp

- Nime omav lõplik väärtuste hulk.
- Määrab andmed, mida *võib* andmebaasis hoida.
- Igal tüübil on üks või rohkem võimalikku *kasutajale mõeldud esitusviisi*.
- Igal tüübil on üks või rohkem võimalikku *sisemist esitusviisi*.

Andmetüüp e tüüp (2)

- Iga tüübi nimel on null või rohkem *sünonüümi*.
- Iga tüübi jaoks on olemas üks või rohkem *operaatorit*, mis määravad võimalikud operatsioonid seda tüüpi andmetega.

Stringitüübid

SQL:2016	Oracle (siin ja edaspidi ver 12c Rel. 1)	PostgreSQL (siin ja edaspidi ver 10)
character (n) ***char(n)*** 'Martin'	***char (n [byte|char])*** n: baitide arv (vaikimisi) või märkide arv n: max kuni 2000 baiti; vaikimisi väärtus 1. Võib kasutada Unicode kodeeringut.	***character(n)*** ***char(n)*** n: märkide arv; vaikimisi väärtus 1. Väärtuse suurus kuni 1 GB.
character varying(n) ***char varying(n)*** ***varchar (n)*** 'Martin'	***varchar2 (n [byte|char])*** n: baitide arv (vaikimisi) või märkide arv n: max kuni 4000 baiti (vaikimisi). Muutes juhtparameetri väärtust võib n olla kuni 32767 baiti. Võib kasutada Unicode kodeeringut.	***character varying(n)*** ***varchar(n)*** n: märkide arv; vaikimisi max suurus. Väärtuse suurus kuni 1 GB.

CHAR või VARCHAR?

- Veerg tüüpi CHAR(10)
  - Kui salvestan stringi e sõne, mis on lühem kui 10 (baiti/märki – sõltub andmebaasisüsteemist), siis pannakse maksimumpikkuse saavutamiseks sõne lõppu *tühikud*.
- Veerg tüüpi VARCHAR(10)
  - Kui salvestan stringi, mis on lühem kui 10 (baiti/märki – sõltub andmebaasisüsteemist), siis tühikuid lõppu ei lisata.

CHAR või VARCHAR? (2)

- CHAR eelised.
  - Reserveeritakse kindel salvestusruum. Väldib ridade migreerumist (andmebaasisüsteemi poolset rea uude plokki paigutamist).
- CHAR puudused.
  - Stringi lõpus olevad tühikud "lähevad kaotsi", sest pole selge, millised nendest kasutaja pandud ja millised süsteemi lisatud.
  - Konkateneerimine, otsimine.
- VARCHAR eelised.
  - Hoiab salvestusruumi kokku.
- VARCHAR puudused.
  - Seda tüüpi andmeid sisaldava rea muutmine *võib* olla aeglasem, sest võib toimuda rea migreerumine (kui VARCHAR tüüpi väärtus suureneb).

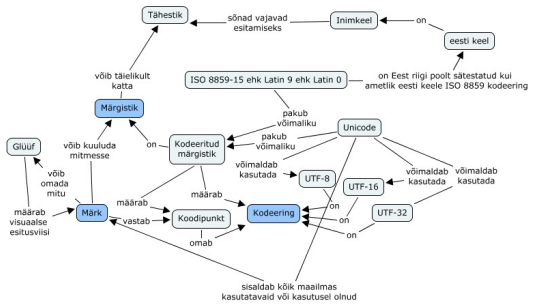
CHAR – konkateneerimine

- CREATE TABLE Isik(eesnimi CHAR(20), perenimi CHAR(20));
- INSERT INTO Isik (eesnimi, perenimi) VALUES ('Mart','Mets');
- SELECT eesnimi || perenimi AS nimi FROM Isik;
- PostgreSQL tulemus: MartMets
- Oracle tulemus: Mart Mets

CHAR – otsimine

- Leia isikud, kelle perenimi lõpeb tähtedega *ts*.
- PostgreSQL, Oracle
  - SELECT * FROM Isik WHERE perenimi LIKE '%ts';
  - Päringu tulemuses **0 rida**, sest salvestatud perenime lõpus on tühikud.

Märgid, märgistikud, kodeeringud



Stringitüübid (2)

SQL:2016	Oracle	PostgreSQL
national char (n) ***nchar (n)*** 内心平静	***nchar(n)*** n: märkide arv Max suurus 2000 baiti. Unicode.	***char(n)*** koos andmebaasi märgistikuga UTF8
national char varying (n) ***nvarchar(n)***	***nvarchar2(n)*** n: märkide arv Max suurus 4000 baiti. Unicode. Muutes juhtparameetri väärtust võib n olla kuni 32767 baiti.	***varchar(n)*** koos andmebaasi märgistikuga UTF8

Stringitüübid (2)

SQL:2016	Oracle	PostgreSQL
puudub	***long*** Max suurus 2GB (tabelis saab olla vaid üks seda tüüpi veerg; **selle asemel soovitatakse kasutada CLOB**)	***varchar***, ***text*** Väärtuse suurus kuni 1 GB

CHAR käitumine – PostgreSQL

- CREATE TABLE test(txt CHAR(3));
- INSERT INTO test (txt) VALUES (**'ab '**);
- SELECT * FROM test WHERE txt=**'ab'**;
  - Enne võrdluse tegemist 'ab'=>'ab '
  - Result: 'ab ' (1 row)
- SELECT * FROM test WHERE length(txt)=length('ab');
  - Result: 'ab ' (1 row)

  (sest 2=2 => TRUE)

length – süsteemi-defineeritud funktsioon.

CHAR käitumine – Oracle

- CREATE TABLE test(txt CHAR(3));
- INSERT INTO test (txt) VALUES (**'ab '**);
- SELECT * FROM test WHERE txt=**'ab'**;
  - Result: 'ab ' (1 row)
- SELECT * FROM test WHERE length(txt)=length('ab');
  - Result: (0 rows)

  (sest 3=2 => FALSE)

CHAR käitumine – PostgreSQL (2)

- INSERT INTO test (txt) VALUES (**'ab '**);
  - Lisatava stringi lõpus **kaks** tühikut
  - String lisatakse tabelisse, üks tühik "lõigatakse" lõpust ära
- INSERT INTO test (txt) VALUES (**'abcd'**);
  - ERROR: value too long for type character(3)

CHAR käitumine – Oracle (2)

- INSERT INTO test (txt) VALUES (**'ab '**);
  - ERROR at line 1: ORA-12899: value too large for column „C##TUD1"."TEST"."TXT" (actual: 4, maximum: 3)
- INSERT INTO test (txt) VALUES (**'abcd'**);
  - ERROR at line 1: ORA-12899: value too large for column „C##TUD1"."TEST"."TXT" (actual: 4, maximum: 3)

NULL käitumine – PostgreSQL

- UPDATE test SET txt='';
  - Muudeti 2 rida
- SELECT Count(*) AS arv FROM test WHERE txt='';
  - Arv: 2
- SELECT Count(*) AS arv FROM test WHERE txt IS NULL;
  - Arv: 0

NULL käitumine – Oracle

- UPDATE test SET txt='';
  - Muudeti 1 rida
- SELECT Count(*) AS arv FROM test WHERE txt='';
  - Arv: 0
- SELECT Count(*) AS arv FROM test WHERE txt IS NULL;
  - Arv: 1

Oracle teeb teisenduse ''=>NULL

Kuupäevatüüp

SQL:2016	Oracle	PostgreSQL
date '0001-01-01 AD' – '9999-12-31 AD' '2008-03-02'	***date*** '4712-01-01 BC' – '9999-12-31 AD' (v.a aasta 0) SELECT to_char(loomise_aeg, 'YYYY-MM-DD') AS aeg FROM Aine; Seda tüüpi veerus võib registreerida nii kuupäeva kui ka kellaaja (ei saa registreerida sekundi murdosi, ega ajavööndit).	***date*** '4713-01-01 BC' – '5874897-12-31 AD' (v.a aasta 0) BC – Enne Kristust AD – Kristuse Aastal 4713-01-01 BC – Juliuse päevade algus (Kuupäeva saab esitada kui täisarvu, mis näitab päevade arvu sellest kp)

Kuupäev – PostgreSQL

- DROP TABLE IF EXISTS Test;
- CREATE TABLE Test (kp DATE);
- INSERT INTO Test(kp) VALUES ('2016-06-09 23:00');
- SELECT to_char(kp, 'YYYY-DD-MM HH24:MI') AS aeg FROM Test;
  - Tulemus: 2016-09-06 **00:00**

Kuupäev – Oracle

- DROP TABLE Test;
- CREATE TABLE Test (kp DATE);
- INSERT INTO Test(kp) VALUES (**to_date**('2014-06-09 23:00', 'YYYY-MM-DD HH24:MI'));
- SELECT to_char(kp, 'YYYY-DD-MM HH24:MI') aeg FROM Test;
  - Tulemus: 2014-09-06 **23:00**

Kuupäeva registreerimine

- PostgreSQL.
  - CREATE TABLE Isik (.... synni_kp DATE ...);
- Oracle.
  - CREATE TABLE Isik (... synni_kp DATE .... CONSTRAINT chk_isik_synni_kp CHECK(**trunc**(synni_kp)=synni_kp));
  - Kitsendus tagab, et andmebaasis registreeritakse sünniaeg kuupäeva täpsusega.

Kuupäeva registreerimine – eksperiment (Oracle)

- CREATE TABLE Isik (isik_id NUMBER(10) PRIMARY KEY, synni_kp DATE NOT NULL);
- INSERT INTO Isik(isik_id, synni_kp) VALUES (1, to_date('2000-04-03 11:00', 'YYYY-MM-DD HH24:MI'));
- Tüüpilise päringuga otsitakse isikut ainult sünnikuupäeva järgi. Järgmise päringu tulemuseks ei ole ühtegi rida.
  - SELECT * FROM Isik WHERE synni_kp= to_date('2000-04-03', 'YYYY-MM-DD');
- Mis on selle põhjus?
  - Väide: 2000-04-03 11:00 = 2000-04-03 00:00 => FALSE

Kellaajatüüp

SQL:2016	Oracle	PostgreSQL
time '13:24:36.1'	Kellaaja hoidmiseks tuleb kasutada veergu, mis on tüüpi ***date*** või ***timestamp***.	***time [(p)] [with time zone | without timezone]*** p – sekundi murdosa täpsus – 0 ... 10

Timestamp tüüp

SQL:2016	Oracle	PostgreSQL
timestamp (kuupäev+ kellaaeg+ ajavöönd) '2008-04-02 12:31:11+3'	***timestamp [(p)] [with time zone]*** Kuupäevad samas vahemikus kui DATE korral. p – sekundi murdosa täpsus – 0..9, vaikimisi 6.	***timestamp [(p)] [with time zone | without time zone]*** Vaikimisi ajavööndita. 1. jaanuar 4713 BC – 31. detsember 294276 AD (v.a aasta 0) p – sekundi murdosa täpsus – 0..6, vaikimisi 6.

Intervallitüüp

SQL:2016	Oracle	PostgreSQL
interval '4' YEAR '50' MONTH '94' DAY '45 23:16' DAY TO MINUTE '23:16' HOUR TO MINUTE	***interval year (aasta täpsus) to month (kuu täpsus)*** ***interval day (päeva täpsus) to second (sekundi murdosa täpsus)***	***interval [(p)]*** p – sekundi_murdosa _täpsus – 0..6

Täisarvutüübid

SQL:2016	Oracle	PostgreSQL
smallint 11	***number(t)*** t – täpsus – numbrite koguarv, vahemikus 1 ja 38	***smallint*** -32768 kuni +32767
integer, int 334	***number(t)***	***integer*** -2147483648 kuni +2147483647
bigint 23344	***number(t)***	***bigint*** -9223372036854775808 kuni 9223372036854775807

Täisarvutüübid – unikaalsete väärtuste genereerimine

SQL:2016	Oracle	PostgreSQL
puudub	puudub	***smallserial*** (baastüüp SMALLINT) ***serial*** (baastüüp INTEGER) ***bigserial*** (baastüüp BIGINT) Pole päris tüübid, vaid notatsioon arvujada generaatorite loomiseks.

Reaaltüübid

SQL:2016	Oracle	PostgreSQL
float(t) *(ujukomatüüp)*	***float(t)*** t – täpsus, maksimaalne numbrikohtade arv kahendarvus (1–126). Väärtused salvestatakse kümnendsüsteemis (numbrid 0-9). Näide: 12345.65789 Ümardamisel, arvutustes, võrdlustes ei teki ebatäpsusi.	***float(t)*** t – täpsus, maksimaalne numbrikohtade arv kahendarvus (1–53) (IEEE 754, kahend)
real *(ujukomatüüp)* -.27982E+015	***float(63)*** ***binary_float*** (IEEE Standard for Floating-Point Arithmetic (IEEE 754)). Väärtused salvestatakse kahendsüsteemis (0,1). Näide: 10001.10010110011 **Osasid arve ei saa niimoodi täpselt salvestada.** Ümardamisel, arvutustes, võrdlustes tekib ebatäpsusi.	***float(t)*** t – täpsus, (1–24) (IEEE 754) ***real*** (IEEE 754, kahend)

Reaaltüübid (2)

SQL:2016	Oracle	PostgreSQL
double precision *(ujukomatüüp)* 0.465E-007	***float(126)*** ***binary_double*** (IEEE 754, kahend)	***float(t)*** t – täpsus, (25–53) (IEEE 754 kahend) ***double precision*** (IEEE 754, kahend)
numeric (p, s) ***decimal (p, s)*** *(püsikomatüüp)* -999.3	***number (t, s)*** t – täpsus – numbrite koguarv s – skaala – numbrite arv paremal pool koma	***decimal (t, s)*** ***numeric (t, s)***

Ümardamise vigade vältimiseks DECIMAL või NUMBER!

SQL antimuster – ümardamise vead

- Karwin, B., 2010. *SQL antipatterns : avoiding the pitfalls of database programming*, Pragmatic Bookshelf. 333 p.
  - Peatükk *Rounding Errors*
  - Kui ujukomatüüpi väärtuseid salvestatakse **IEEE 754** määratud kahendformaadis (nt 10001.1001011001), siis võib juhtuda, et salvestavat väärtust ei saa süsteemis täpselt sellisena esitada ja salvestatakse sellele lähedane väärtus. **Tekivad ümardamise ja võrdlemise vead!**
  - https://www.h-schmidt.net/FloatConverter/IEEE754.html

Arvutüübid – PostgreSQL

- Iga arvutüübi T korral kulub sellesse tüüpi kuuluvate väärtuste salvestamiseks mingi fikseeritud hulk salvestusruumi.
  - SMALLINT – 2 baiti
  - REAL – 4 baiti
  - ...

Arvutüübid – Oracle

- Kasutusel tüüp NUMBER
- Kõik NUMBER tüüpi väärtused salvestatakse ühel viisil
- Oracle NUMBER tüüpi väärtused on muutuva pikkusega – erinevate arvude salvestamiseks kulutatakse erinev hulk salvestusruumi
- Arv **523** salvestatakse kui arv **5.23*10^2**
- Salvestamiseks 3 baiti:
  - Astendaja baidis #0: 2
  - Mantiss (arvu tüvi) baidis #1: 5
  - Mantiss baidis #1: 2
  - Mantiss baidis #2: 3
  - Lisaks on baidis #2 ka tühik, et see bait oleks täiesti täidetud.

Arvutüübid – Oracle (2)

- Arvude salvestamisest tulevad soovitused.
  - Korrutamise ja jagamise operatsioonid on aeglased.
  - Jätke plokkidesse vaba ruumi, sest arvud on muutuva pikkusega ja reas oleva arvu muutumine võib tingida *rea migreerumise*.
    - Vaikimisi jätab Oracle igasse tabeli plokki sellel otstarbel 10% vaba ruumi.

Bitistringitüübid ja tõeväärtustüüp (Boole'i tüüp)

SQL:2016	Oracle	PostgreSQL
puudub (bit ja bit varying – olid SQL:1999, SQL:2003-st eemaldatud)	puudub	***bit(n)*** n – positiivne täisarv. Kui määramata, siis n=1.
puudub	puudub	***bit varying(n)***
boolean TRUE	puudub (ei saa kasutada veeru tüübina, kuid saab kasutada PL/SQL rutiinides)	***boolean***

Tõeväärtustüüp SQLis

- SQL kasutab *kolmevalentset* loogikat.
- SQLi tõeväärtustüüpi kuuluvad väärtused:
  - TRUE,
  - FALSE,
  - UNKNOWN – selle esitamiseks kasutatakse NULLi.

Tüübid suurte objektide salvestamiseks

SQL:2016	Oracle	PostgreSQL
blob (binary large object)	***blob*** Max kuni (2^{32}-1 baiti)*(ploki suurus).	***bytea***
clob (character large object)	***clob*** Max kuni (2^{32}-1 baiti)*(ploki suurus).	***text*** ***varchar*** Max kuni umbes 1 GB
nclob (national character large object)	***nclob*** Max kuni (2^{32}-1 baiti)*(ploki suurus).	***text*** koos andmebaasi märgistikuga UTF8
puudub	***bfile*** ("external Large Object") (viide välisele failile)	***oid***

Sümbioos dokumendipõhise ja võti-väärtus viisil andmete esitusega

SQL:2016	Oracle	PostgreSQL
XML	**XMLType**	**XML**
puudub JSON formaadis dokumendid salvestatakse tekstina Alates SQL:2016 ette nähtud operaatorid ja funktsioonid seda tüüpi andmete töötlemiseks.	puudub JSON formaadis dokumendid salvestatakse VARCHAR2, BLOB või CLOB tüüpi veergu. Alates Oracle 12.1 ette nähtud operaatorid ja funktsioonid seda tüüpi andmete töötlemiseks.	**JSON** (*JavaScript Object Notation*) ja **JSONB** JSON korral salvestatakse andmed tekstina – täpselt sellisena nagu oli sisend, JSONB korral binaarformaadis. JSONB tüüpi väärtuseid on veidi aeglasem salvestada kui JSON väärtuseid, kuid nendest on palju kiirem otsida. JSONB tüüpi väärtuseid saab indekseerida.
puudub	puudub	Hstore tüüp võti-väärtus paaride registreerimiseks

Varia

Oracle	PostgreSQL
URI tüübid Geomeetrilised tüübid Tüübid multimeedia andmete hoidmiseks ROWID	Võrguaadresside tüübid Geomeetrilised tüübid (point, line, circle, ...) Money tüüp (sisaldab valuuta tähist, mis määratud serveri seadetega; arvutused nõuavad tüübiteisendust) Vahemike tüübid (nt kuupäevade vahemiku tüüp)

Oracle ja SQL standardis ettenähtud tüübid

- Tabelite deklareerimisel võib kasutada SQL standardis ettenähtud tüüpide nimesid.
- Oracle võtab vastavustabeli alusel kasutusele enda andmetüübid.
- Näide
  - INTEGER, SMALLINT => NUMBER(38)

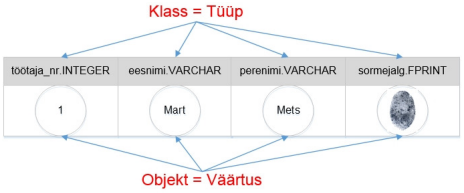
Oracle ja SQL standardis ettenähtud tüübid (2)

```
C##TUD1@ORCL > CREATE TABLE X(a SMALLINT, b INTEGER);

Table created.

C##TUD1@ORCL > DESC X
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------
 A                                                  NUMBER(38)
 B                                                  NUMBER(38)
```


Milline on vastavus objektorienteeritud ja relatsiooniliste mõistete vahel?




```
SELECT töötaja_nr, Upper(perenimi) AS nimi
FROM Töötaja
WHERE eesnimi = perenimi;
```

Meetod = Operaator, Funktsioon

Miks tabel<>klass ja rida<>objekt?

Amet

ameti_kood	nimetus	max_tootur	min_palk
1	administraator	50	700,00 €
2	liini operaator	40	800,00 €

Objektorienteeritud lähenemises on objektil peidetud muutujad, mille väärtustele pääseb ligi avalike meetodite kaudu (kapseldamise omadus). Kui **tabel=klass** ja **rida=objekt**, siis oleks objekti kõik muutujad avalikud – neile pääseb ligi otse, ilma mingite meetoditeta. **See on ju vastuolu objektorienteerituse põhimõtetega!**

Miks tabel<>klass ja rida<>objekt? (2)

Query1

```
SELECT Amet.ameti_kood, Amet.nimetus, Tootaja.perenimi, Tootaja.synni_aeg
FROM Amet INNER JOIN Tootaja ON Amet.ameti_kood = Tootaja.ameti_kood
WHERE on_aktuaalne=FALSE;
```

Query1

ameti_kood	nimetus	perenimi	synni_aeg
1	administraator	Mets	3.01.1981

Mis klassi kuulub punasega tähistatud "objekt"?

Oracle ja PostgreSQL kui objekt-relatsioonilised andmebaasisüsteemid

- Objekt-relatsioonilisuse üks põhitunnus on võimalus luua uusi **andmetüüpe**.
- Nii PostgreSQL kui Oracle andmebaasisüsteemis on kasutusel *CREATE TYPE* lause ja lisaks võimalus kasutada spetsiifilisi operaatoreid – **tüübikonstruktoreid** (nt ARRAY, ROW tüüpide loomiseks) (vt teema 10).
- PostgreSQLis saab luua loendustüüpe.
  - CREATE TYPE lipp AS ENUM ('sinine', 'must', 'valge');
    - Kaaluge kasutamist, kui väärtuste hulk tüübis on ajas muutumatu!

PostgreSQLis tüüpide loomine (2)

- PostgreSQLis on hulk süsteemi-defineeritud vahemiku tüüpe ja saab luua uusi vahemiku tüüpe.
  - CREATE TYPE timerange AS RANGE (subtype = time);
  - CREATE TABLE vastuvotuaeg (aeg timerange PRIMARY KEY);
  - INSERT INTO vastuvotuaeg (aeg) VALUES ('[10:00,10:15)');
  - INSERT INTO vastuvotuaeg (aeg) VALUES ('(**11:00**,**10:15**]');
  - ERROR: range lower bound must be less than or equal to range upper bound

"[" ja "]" – vahemiku otspunkt **on** väärtusesse kaasa arvatud.
"(" ja ")" – vahemiku otspunkt **ei ole** väärtusesse kaasa arvatud.

Millist andmetüüpi kasutada?

veerg	baastüüp (lisaks veel vaja kitsendusi)
on_ aktuaalne	char(1) või number(1) – Oracle (lisaks kitsendus, mis piirab võimalikke väärtusi) boolean – PostgreSQL
kooli hinne (0–5)	number(1) – Oracle smallint – PostgreSQL. Tegemist võiks olla välisvõtme veeruga, mis viitab tabelile *Hinne*. PostgreSQLis võimalik luua ka loendustüüp.
nimi	varchar2(n) või nvarchar2(n) – Oracle varchar(n) või text+CHECK – PostgreSQL

Milline n valida VARCHAR puhul?

- Pikim nimi piiblis *Maher-shalal-hash-baz* on 21 märki pikk (Isaiah 8:1)
- Pikim taksonoomiline nimi *Archaeosphaerodiniopsidaceae* on 28 märki pikk
- Pikim kohanimi Suur-Britannias ja Euroopas – küla Walesis: *Llanfairpwllgwyngyllgogerychwyrndrobwll llantysiliogogogoch* on 58 märki pikk
- http://maaleht.delfi.ee/news/maaleht/elu/mis-on-eesti-koige-pikem-kohanimi?id=64691226

Milline n valida VARCHAR puhul? (2)

Milline n valida VARCHAR puhul? (3)

- Kultuurilised eripärad.
  - Isiku perenimi (36 märki): Janice "Lokelani" Keihanaikukauakahihuliheekahaunaele
    - „Long-named US woman celebrates government climb-down," BBC News, September 14, 2013.
- Kunagi ei saa välistada veidrusi.
  - Isiku nimi (115 märki): Baron Venom Balrog Sabretooth Vader Megatron Vegeta Robotnik Magneto Bison Sephiroth Lex Luthor Skeletor Joker Grind
    - Sperling, D., 2011. „Nottingham men win world's longest names with superhero tribute," August 8, 2011.

Võimalikult täpse n väärtuse määramine

- Eelised
  - n väärtus annab skeemi uurivatele inimkasutajatele ja andmebaasi metaandmeid kasutavatele programmidele infot, milliseid andmeid selles veerus hoitakse,
  - aitab tagasi lükata ebakorrektseid andmeid.
- Probleemid
  - n väärtuse muutmine võib olla aeganõudev (sõltub andmete hulgast) ja loodud vaadetest.

Alternatiivne lahendus PostgreSQLis

- VARCHAR ja TEXT tüüpi väärtuseid salvestatakse sisemiselt ühte moodi.
- Kasuta veeru tüübina TEXT ja lisa CHECKid.

Probleem: metaandmeid lugevad programmid võivad jätta CHECKid tähelepanuta.

```
CREATE TABLE Isik (isik_id SERIAL,
eesnimi TEXT NOT NULL,
perenimi TEXT,
CONSTRAINT pk_isik PRIMARY KEY (isik_id),
CONSTRAINT chk_isik_eesnimi_pikkus CHECK(length(eesnimi)<=255),
CONSTRAINT  chk_isik_perenimi_pikkus CHECK(length(perenimi)<=255));
```

Millist andmetüüpi kasutada? (2)

veerg	baastüüp (lisaks veel vaja kitsendusi)
hind	number(19,4) – Oracle decimal(19,4) – PostgreSQL Lisaks kitsendus, mis keelab negatiivseid väärtusi. **Miks skaala 4?** Üldlahendus. Näiteks väikesed tariifid, ühikuhinnad kaupadel, mida saab osta ainult suurtes kogustes, mõned finantstehingud nõuavad suuremat skaalat kui 2 (3 või 4). Kui hinnad on väga väikesed ja müüdavad kogused väga suured, siis lubada ümardamisprobleemide vältimiseks kuni viis kohta peale koma! Võib ka registreerida hinna täisarvuna sentides ja teisendada jooksvalt eurodeks.

Millist andmetüüpi kasutada? (3)

veerg	baastüüp (lisaks veel vaja kitsendusi)
pikk kommentaar, kirjeldus (nt kauba kasutusjuhend)	clob – Oracle text – PostgreSQL
e-meil	varchar2(254 char)– Oracle varchar(254) – PostgreSQL http://www.rfc-editor.org/errata_search.php?rfc=3696&eid=1690

Millist andmetüüpi kasutada? (4)

veerg	baastüüp (lisaks veel vaja kitsendusi)
telefoni number	varchar2(n char)– Oracle varchar(n) – PostgreSQL ITU-T soovituse E.164 (*The international public telecommunication numbering plan*) kohaselt võib telefoninumbri pikkus olla maksimaalselt 15 numbrit. https://en.wikipedia.org/wiki/E.164 Kui salvestada ka ees- ja järelliited (nt suunakood), siis kuni 31 märki.

CHECK kitsenduses või loendus-tüübis!

SQL antimuster – väärtuste nimekiri veeru definitsioonis

- Karwin, B., 2010. *SQL antipatterns : avoiding the pitfalls of database programming*, Pragmatic Bookshelf. 333 p.
  - Peatükk *31 Flavors*
  - Keerulisem kui vaja:
    - leida päringuga tüüpi kuuluvaid väärtuseid, mida kasutajaliideses kuvada,
    - lisada tüüpi uusi väärtuseid.
  - Võimatu väärtust eemaldada.

Eelista klassifikaatorite tabeleid!

Arvutüübi valikust PostgreSQL näitel

- "arithmetic on numeric values is very slow compared to the integer types, or to the floating-point types described in the next section." (PostgreSQL dokumentatsioon)

Arvutüübi valikust PostgreSQL näitel (2)

- CREATE TABLE Isik_int(isik_id SERIAL, synni_aasta **SMALLINT** NOT NULL, CONSTRAINT chk_Isik_int CHECK (synni_aasta BETWEEN -5000 AND 2200));
- CREATE TABLE Isik_num(isik_id SERIAL, synni_aasta **NUMERIC(4)** NOT NULL, CONSTRAINT chk_Isik_num CHECK (synni_aasta BETWEEN -5000 AND 2200));

Arvutüübi valikust PostgreSQL näitel (3)

- Genereerin tabelisse 1 miljon rida, kus sünni aasta on vahemikus -5000 ja 2200 (otspunktid kaasa arvatud)
  - INSERT INTO Isik_int (synni_aasta) SELECT (RANDOM()*(2200-(-5000)))::INT + (-5000) FROM (SELECT generate_series(1,1000000)) AS foo;
  - INSERT INTO Isik_num (synni_aasta) SELECT synni_aasta FROM Isik_int;
- Värskendan statistikat
  - ANALYZE;

Arvutüübi valikust PostgreSQL näitel (4)

- NUMERIC tüübi kasutamise korral on andmemahud suuremad
  - SELECT pg_total_relation_size('Isik_num');
    - 44318720 (baiti)
  - SELECT pg_total_relation_size('Isik_int');
    - 36282368 (baiti)
  - *pg_total_relation_size* – tabeli suurus koos seotud indeksite ja TOAST andmete suurusega

Arvutüübi valikust PostgreSQL näitel (5)

- Aritmeetikaoperatsioonide töökiirus on NUMERIC tüübi puhul madalam, kuid PostgreSQL 10s on kiirus paranenud.
  - \timing
- SELECT Avg(synni_aasta) AS keskm FROM Isik_num;
  - Time: 113.518 ms; Time: 86.469 ms; Time: 93.754 ms
- SELECT Avg(synni_aasta) AS keskm FROM Isik_int;
  - Time: 71.636 ms; Time: 87.580 ms Time: 71.989 ms

Arvutüübi valikust PostgreSQL näitel (6)

- Kui mul on PostgreSQLi andmebaasis vaja säilitada täisarvu tüüpi väärtuseid, võiks kasutada täisarvutüüpe, mitte NUMERIC
- Hindu võib hoida täisarvu tüüpi veerus sentides. Ei tohi unustada vajadusel teisendada (nt eurodeks)
- Kui olete analoogilise küsimuse ees, siis **eksperimenteerige**!!!

Võrdluseks

- Laadisin tabelis *Isik_int* olevad andmed COPY … TO lauset kasutades tekstifaili.
- Laadisin tekstifailis olevad andmed **välist tabelit** kasutades Oracle andmebaasi tabelisse *Isik,* kus veerg *synni_aasta* on tüüpi NUMBER(4).
- SELECT Avg(synni_aasta) AS keskm FROM Isik;
  - Time: 140 ms; Time: 100 ms; Time: 120 ms

Võrdluseks (2)

Võimsam riistvara ja uuem tarkvara versioon ei pruugi tähendada paremat töökiirust

	SMALLINT (PostgreSQL)	NUMERIC (PostgreSQL)	NUMBER (Oracle)
hektor8.ttu.ee (1 CPU, 6 GB mälu) PostgreSQL 9.0 Oracle 11g Rel. 1	Time: 225.329 ms; Time: 209.160 ms; Time: 204.939 ms	Time: 947.025 ms; Time: 946.736 ms; Time: 936.219 ms	Time: 110 ms; Time: 120 ms; Time: 110 ms
hektor3.ttu.ee (3CPU, 12 GB mälu) PostgreSQL 9.2 Oracle 11g Rel. 2	Time: 114.668 ms; Time: 103.941 ms; Time: 104.794 ms	Time: 555.870 ms; Time: 557.944 ms; Time: 545.818 ms	**Time: 70 ms;** **Time: 70 ms;** **Time: 70 ms**
apex.ttu.ee (15 CPU, 40 GB mälu) PostgreSQL 10 Oracle 12c Rel. 1	**Time: 71.636 ms** **Time: 87.580 ms** **Time: 71.989 ms**	**Time: 113.518 ms** **Time: 86.469 ms** **Time: 93.754 ms**	Time: 140 ms Time: 100 ms Time: 120 ms

Link

Vahekokkuvõte

- SQL standard kirjeldab *süsteemi-defineeritud tüüpe*, *süsteemi-defineeritud funktsioone* ja *süsteemi-defineeritud operaatoreid*, kuid konkreetne andmebaasisüsteem võib standardit järgida täielikult, osaliselt või üldse mitte.
- Erinevates andmebaasisüsteemides võib erinevatel tüüpidel (mis tähistavad erinevat väärtuste hulka) olla sama nimi.

Vahekokkuvõte (2)

- Erinevates andmebaasisüsteemides võib samadel tüüpidel (mis tähistavad ühte ja sama väärtuste hulka) olla erinev *nimi*.
- Erinevates andmebaasisüsteemides võib samadel tüüpidel (mis tähistavad ühte ja sama väärtuste hulka) olla erinev sisemine esitusviis ning sellest tulenevalt ka erinevused seda tüüpi väärtustega tehtavate operatsioonide *töökiiruses*.

Vahekokkuvõte (3)

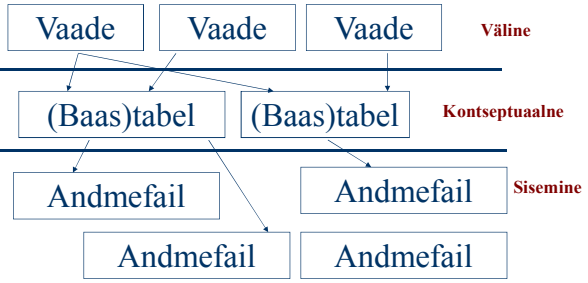
- Erinevates andmebaasisüsteemides võib samade tüüpide jaoks (mis tähistavad ühte ja sama väärtuste hulka) olla erinev hulk süsteemi-defineeritud funktsioone ja operaatoreid, mis võimaldavad seda tüüpi väärtuseid *töödelda*.
- Erinevates andmebaasisüsteemides võib samasuguse nimega operatsioon sama tüüpi andmetega anda *erineva tulemuse*.

Andmebaasi tasemed (kihid)

- **Väline** – andmebaasi kasutajate jaoks mõeldud vaated andmetele (virtuaalsed tabelid).
- **Kontseptuaalne (loogiline)** – andmebaasi struktuuri (baastabelid) ja käitumise (kitsendused) kirjeldus.
- **Sisemine** – andmete salvestamise küsimused (failid, plokid, indeksid jne).
- Tänapäeva SQL-andmebaasisüsteemides pole need tasemed selgelt eristatud!

Andmebaasi tasemed (2)

Huvide lahususe (*separation of concerns*) disainiprintsiibi ilming – igal tasemel oma ülesanded!



Baastabel (tabel)

Oracle	PostgreSQL
◆Ajutised tabelid	◆Ajutised tabelid
◆Tabelitüüpi veerg (*nested table*)	
◆Tabeli loomine tüübi põhjal	◆Tabeli loomine tüübi põhjal
◆Tabeli loomine tüübi põhjal, mis on loodud pärimist kasutades	◆Tabeli loomine pärimise kaudu
◆XML tüüpi tabelid ja veerud	◆XML tüüpi veerud
◆Regulaaravaldised tabeli kitsendustes	◆Regulaaravaldised tabeli kitsendustes
◆Baastabeli loomine päringu tulemusest (CREATE TABLE AS ... SELECT ...)	◆Baastabeli loomine päringu tulemusest (CREATE TABLE AS ... SELECT ...)
◆Nähtamatu veerg	◆Nähtamatud süsteemsed (väärtuseid lisab süsteem) veerud (nt oid, xmin)
◆Virtuaalne veerg	

Baastabeli andmete *sisemisel* tasemel salvestamine

Oracle	PostgreSQL
◆Indeks-organiseeritud tabel (IOT) ◆Klastrisse koondatud tabelid ◆Välised tabelid ◆Tabelite sektsioonideks jagamine ◆Tabelis olevate andmete pakkimine ◆LOB tüüpi andmed salvestatakse üldiselt ülejäänud reast eraldi. Saab määrata, et väärtus <=4000 baiti salvestatakse reaga koos ◆Tabelis olevate andmete krüpteerimine ◆Tabeliruumid	◆Tabeli ridu saab sisemisel tasemel sorteerida ◆Välised tabelid, välistest failidest saab andmeid laadida ◆Sektsioone saab programmeerida. Alates ver 10 saab ka luua deklaratiivselt ◆TOAST (*The Oversized-Attribute Storage Technique*) ◆Tabelis olevate andmete krüpteerimine (moodul *pgcrypto*) ◆Tabeliruumid

Andmed välistest allikatest

Oracle	PostgreSQL
Andmebaasi lingid (*database links*), välised tabelid (*external table*).	Välised serverid (*foreign servers*), väliste andmete pakendajad (*foreign data wrappers*), välised tabelid (*foreign tables*), kasutajate ja väliste serverite vastavused (*user mappings*).

Baastabel (PostgreSQL)

```
CREATE TABLE Tudeng(
    yliopilaskood CHAR(10) ,
    eesnimi VARCHAR(1000) NOT NULL,
    perenimi VARCHAR(1000),
    synni_kuupaev DATE NOT NULL,
    sugu CHAR(1) NOT NULL,
    aadress VARCHAR(1000),
    telefon VARCHAR(31),
    e_mail VARCHAR(254),
CONSTRAINT pk_c_Tudeng PRIMARY KEY (yliopilaskood),
CONSTRAINT chk_Tudeng_yliopilaskood CHECK (yliopilaskood ~ '^[0-9]{6}[A-Z]{4}$'),
CONSTRAINT chk_c_Tudeng_synni_kuupaev CHECK (synni_kuupaev<CURRENT_DATE),
CONSTRAINT chk_c_Tudeng_sugu CHECK (sugu IN ('M','N')),
CONSTRAINT chk_c_Tudeng_e_mail CHECK (e_mail~ '^.+@.+[.].{2,3}$'),
CONSTRAINT chk_c_Tudeng_eesnimi CHECK (trim(eesnimi)<> ''),
CONSTRAINT chk_c_Tudeng_perenimi CHECK (trim(perenimi)<> ''));
```

vastab mustrile – tõstutundlik

Regulaaravaldise näide

- **'^[0-9]{6}[A-Z]{4}$'**
  - 6 numbrimärki (0–9), millele järgneb 4 suurtähte (A–Z)
  - Väärtus, mis *vastab* mustrile:
    - 990999LAPB
  - Väärtus, mis *ei vasta* mustrile:
    - 99LAP
    - 990999123
    - LAP990999
    - 040999iapb

Regulaaravaldise näide (2)

- **'^.+@.+[.].{2,3}$'**
  - 1 või rohkem suvalist märki, millele järgneb @, millele järgneb 1 või rohkem suvalist märki, millele järgneb punkt, millele järgneb 2 kuni 3 suvalist märki:
  - Väärtus, mis vastab mustrile:
    - test@ttu.ee
    - @@@.ee
  - Väärtus, mis ei vasta mustrile:
    - test.ttu.ee
    - test_ät_ttu_ee
    - test@ttu.eeee

Samad probleemid loendus-tüübiga!

SQL antimuster – väärtuste nimekiri veeru definitsioonis

```
CREATE TABLE
    Tudeng(…
    sugu CHAR(1) NOT
    NULL,
    …
CONSTRAINT
    chk_c_Tudeng_sugu
    CHECK (sugu IN
    ('M','N')), …);
```

- Kuidas saada väärtused kasutajaliideses valiku tegemiseks?
- Kuidas lisada uusi väärtuseid?
- Kuidas saaks mõne väärtuse mitteaktiivseks muuta?

Ebasobilik just siis, kui väärtuste hulk ajas muutub!

SQL antimuster – kuidas lahendada

- Parem lahendus klassifikaatori tabel.
  - Sugu(soo_kood, nimetus, on_aktiivne) Primaarvõti (soo_kood) Alternatiivvõti (nimetus)
- Sex and Gender Diverse Passport Applicants. Revised Policy. https://www.passports.gov.au/passportsexplained/theapplicationprocess/eligibilityoverview/Pages/changeofsexdoborpob.aspx
- Tsitaat: "We can issue a passport to sex and gender diverse applicants, identifying them as M (male), F (female) or X (indeterminate/intersex/unspecified)." (28.09.2016)

Tabeli sisemise taseme omadused (Oracle)

- Parameetrid, mille väärtused määravad, kuidas tabeli andmeid salvestada – PCTFREE, PCTUSED, INITRANS, MAXTRANS, FREELIST jne.
- Saab määrata, et andmeid tuleks *pakkida*.
- Saab määrata, et mingis veerus peaksid olema andmed *krüpteeritud*.

Mäluseadmes andmed pakituna/krüpteerituna. Lahtipakkimine ja dekrüpteerimine toimub andmebaasi kasutajale nähtamatult.

Tabeli sisemise taseme omadused (PostgreSQL)

- Tabeli andmete salvestamist kirjeldab atribuut FILLFACTOR (vaikimisi väärtus 100).
  - Määrab iga tabeli andmete salvestamiseks mõeldud ploki korral, kui mitme protsendi ulatuses peab olema plokk täitunud (*klaas täis*), et sinna ei saaks enam lisada uusi ridu (*kallata vett*).
  - CREATE TABLE Tellimus(tellimus_id SERIAL PRIMARY KEY, loomise_aeg DATE NOT NULL, kinnitamise_aeg DATE NOT NULL, kommentaar VARCHAR(1000)) WITH (fillfactor=90);

TOAST (The Oversized-Attribute Storage Technique) (PostgreSQL)

- PostgreSQLi andmeplokkidel on fikseeritud suurus (tavaliselt 8KB) ja süsteem ei luba salvestada ühe rea andmeid mitmesse andmeplokki.
- Kuidas toimida, kui rida on nii suur, et ei mahu plokki ära?
  - TOASTi toetavate andmetüüpide puhul saab sellesse tüüpi kuuluvat väärtust pakkida ja/või jagada mitmeks osaks, mis salvestatakse eraldi füüsiliste ridadena eraldiseisvas tabelis.
  - Kõik need tegevused toimuvad andmebaasi kasutaja eest varjatult ja ei mõjuta oluliselt andmete kasutamise kiirust.

Vaikimisi meetod tabelis olevate andmete salvestamiseks

- Andmete salvestamine *kuhja* põhimõttest lähtuvalt (*heap-organized table*)
- Read salvestatakse esimesse leitud plokki, kus on vaba ruumi:
  - ei otsita mingit kindlat plokki, kuhu read salvestada,
  - olemasolevaid ridu ei paigutata teise plokki, et uutele ridadele ruumi teha,
  - andmefailis ei ole read sorteeritud.

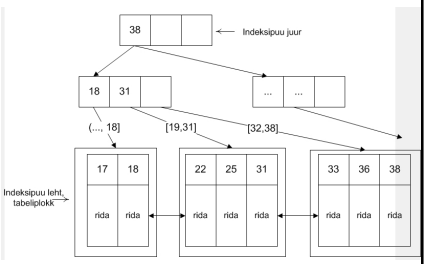
Tabeli andmete sisemisel tasemel sorteerimine

- *Klasterdamine* – koos kasutatavad andmed võiksid olla lähestikku.
- Erinevad lahendused.
  - Index Organized Tables (IOT) (Oracle)
    - Andmebaasisüsteem hoiab järjekorda
  - Indeksi järgi ühekordselt sorteeritud tabel (PostgreSQL)
    - Andmebaasisüsteem ei hoia järjekorda

Indeks-organiseeritud tabel (Oracle)

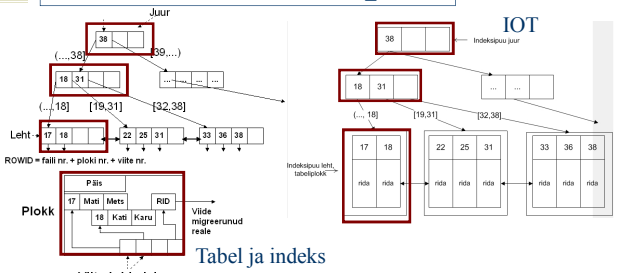
CREATE TABLE Isik (isik_id INTEGER PRIMARY KEY, ...) ORGANIZATION INDEX;

(isik_id) – klastri võti, selle alusel on read sorteeritud



Päringu täitmine

SELECT * FROM Isik WHERE isik_id=17;



Tabel ja indeks eraldi

Indeks-organiseeritud tabeli eelised

- Summaarne andmemaht võib väheneda – pole vaja **eraldi** hoida tabelit ja primaarvõtmele loodud indeksit.
- Read on primaarvõtme väärtuste järgi **sorteeritud** – võimaldab kokku kuuluvad andmed paigutada füüsiliselt **lähestikku**.

Indeks-organiseeritud tabeli eelised (2)

- 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.

Indeks-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**.

Indeks-organiseeritud tabel – millal kasutada?

- Tabelites, kus primaarvõti hõlmab kõiki või enamikke veerge.
  - Tabelid, mis on loodud M:N seosetüübile vastavate andmete hoidmiseks.
  - Kiirendab ühendamisoperatsioone.
    - Kauba_kat_omamine (kauba_kood, kauba_kat_kood) Primaarvõti (kauba_kood, kauba_kat_kood)
    - Sama kauba kõigi kategooria omamiste andmed on füüsiliselt lähestikku. Kiirendab tabelite *Kaup* ja *Kauba_kat* ühendamist.

Indeks-organiseeritud tabel – millal kasutada? (2)

- Kui võtmeveerge kasutatakse vahemikul põhinevaks otsinguks.
  - Tellimus(kliendi_nr, loomise_aeg, ...) Primaarvõti(kliendi_nr, loomise_aeg)
  - SELECT * FROM Tellimus WHERE kliendi_nr=1 AND loomise_aeg BETWEEN '2016-01-01' AND '2016-08-01';

Klasterdamine (PostgreSQL)

- Tabeli read sorteeritakse *ühekordselt* indekseeritud veergude järgi. Sorteerimise ajal ei saa tabelis toimuda lugemise ja muutmise operatsioone.
- Korraga saab tabel olla klasterdatud ühe indeksi järgi.
- CREATE INDEX idx_oppimine_eelreg_kuup ON Oppimine (eelregistreerimise_kuupaev);
- CLUSTER Oppimine USING idx_oppimine_eelreg_kuup;

Klasterdamine – millal kasutada?

- Veeru *v* puhul, kui *v* põhjal tehakse *vahemikul põhinevaid* otsinguid
  - SELECT ... WHERE v BETWEEN <väärtus> AND <väärtus>;
  - SELECT Count(*) AS arv FROM Oppimine WHERE eelregistreerimise_kuupaev BETWEEN '2014-01-01' AND '2014-01-05';
- Veeru *v* puhul, kui (*v*) on välisvõti, et kiirendada ühendamisoperatsioone

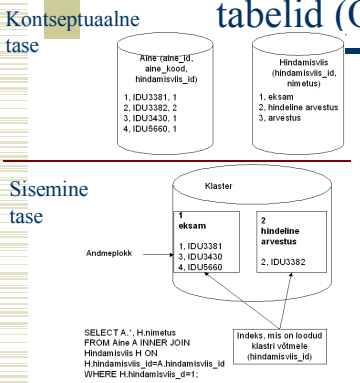
Klastrisse (kobarasse) koondatud tabelid (Oracle)

- Ühes plokis on mitme tabeli andmed.
- Tabelite ühendamine SISEMISEL tasemel.
- Võrdle **denormaliseerimisega**, mis on tabelite ühendamine KONTSEPTUAALSEL tasemel.
- Klastri võti – selle alusel toimub erinevates tabelites olevate andmete ühendamine
  - Näide: (hindamisviis_id) on tabelis *Hindamisviis* primaarvõti ja tabelis *Aine* välisvõti.

Klastrisse (kobarasse) koondatud tabelite probleemid (Oracle)

- Tuleb määrata *fikseeritud suurus*, mis näitab ühe klastri võtme väärtusega seotud andmehulka.
  - *Liiga suur* – salvestusruumi raiskamine.
  - *Liiga väike* – andmed ei mahu määratud plokkidesse ja tekib ülevool.
- Võimalik salvestusruumi raiskamine.
- Andmemuudatused klasterdatud tabelites muutuvad aeglasemaks.

Indeksi alusel klasterdatud tabelid (Oracle)



- Plokk, kuhu andmed paigutada ja kust andmeid otsida, leitakse *indeksi* abil.

Edasi

Indeksi alusel klasterdatud tabelid (Oracle) (2)

- CREATE CLUSTER hindamisviis_id_ind_cluster (hindamisviis_id NUMBER(2)) SIZE 1024;
  - SIZE – kui mitu baiti andmeid on seotud ühe klastri võtme väärtusega?
- CREATE INDEX idx_hindamisviis_id ON CLUSTER hindamisviis_id_ind_cluster;

Indeksi alusel klasterdatud tabelid (Oracle) (3)

- CREATE TABLE Aine (ainekood VARCHAR2(7) PRIMARY KEY, hindamisviis_id NUMBER(2) NOT NULL) CLUSTER **hindamisviis_id_ind_cluster(hindamisviis_id)**;
- CREATE TABLE Hindamisviis(hindamisviis_id NUMBER(2) PRIMARY KEY) CLUSTER **hindamisviis_id_ind_cluster(hindamisviis_id)**;
- INSERT INTO Hindamisviis(hindamisviis_id) VALUES (1);
- INSERT INTO Aine(ainekood, hindamisviis_id) VALUES ('IDU0120', 1);

Indeksi alusel klasterdatud tabelid (Oracle) (4)

```
TUD10@SQL> SELECT
dbms_rowid.rowid_object (rowid) objekti_identifikaator,
dbms_rowid.rowid_relative_fno (rowid) faili_identifikaator,
dbms_rowid.rowid_block_number(rowid) ploki_identifikaator,
dbms_rowid.rowid_row_number(rowid) positsiooni_identifikaator
FROM Hindamisviis;
  2    3    4    5    6
OBJEKTI_IDENTIFIKAATOR FAILI_IDENTIFIKAATOR PLOKI_IDENTIFIKAATOR POSITSIOONI_IDENTIFIKAATOR
---------------------- -------------------- -------------------- --------------------------
                114520                    4                  512                          0

TUD10@SQL> SELECT
dbms_rowid.rowid_object (rowid) objekti_identifikaator,
dbms_rowid.rowid_relative_fno (rowid) faili_identifikaator,
dbms_rowid.rowid_block_number(rowid) ploki_identifikaator,
dbms_rowid.rowid_row_number(rowid) positsiooni_identifikaator
FROM Aine;
  2    3    4    5    6
OBJEKTI_IDENTIFIKAATOR FAILI_IDENTIFIKAATOR PLOKI_IDENTIFIKAATOR POSITSIOONI_IDENTIFIKAATOR
---------------------- -------------------- -------------------- --------------------------
                114520                    4                  512                          0
```

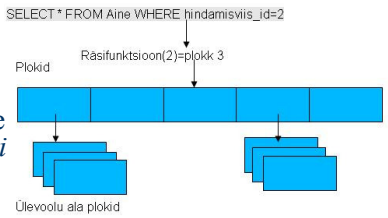
Erinevates tabelites olevad read on sama füüsilise aadressiga!

Räsifunktsioon

- Räsifunktsiooniks nimetatakse funktsiooni $f:\{0,1\}^* \Rightarrow \{0,1\}^k$, mis seab suvalise pikkusega bitijadale vastavusse fikseeritud pikkusega k bitijada.
- Probleem – mitme erineva sisendi korral võib tulemuseks olla sama väärtus, st leidub f(x)=f(y) ja x!=y – tekib põrge e kollisioon.

Räsifunktsiooni alusel klasterdatud tabelid (Oracle)

- Plokk, kuhu andmed paigutada ja kust andmeid otsida, leitakse *räsifunktsiooni* abil.



Edasi

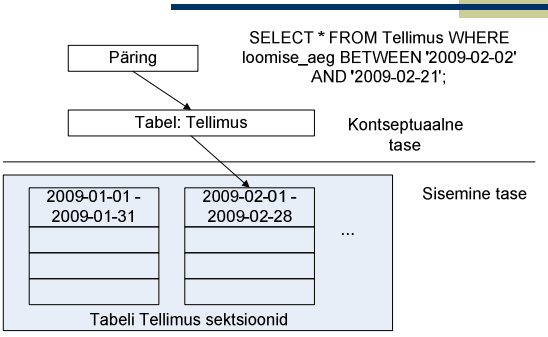
Räsifunktsiooni alusel klasterdatud tabelid (Oracle) (2)

- CREATE CLUSTER hindamisviis_id_hash_cluster (hindamisviis_id NUMBER(2)) HASHKEYS 3 SIZE 8192;
  - **HASHKEYS** – oodatav erinevate klastri võtmete väärtuste arv
  - **SIZE** – kui mitu baiti andmeid on seotud ühe klastri võtme väärtusega?
- CREATE TABLE Aine (...) CLUSTER hindamisviis_id_hash_cluster(hindamisviis_id);
- CREATE TABLE Hindamisviis(...) CLUSTER hindamisviis_id_hash_cluster(hindamisviis_id);

SQL antimuster – tabeli kloonimine

- Jaga suur tabel *horisontaalselt* väikesteks tabeliteks – antimuster (kirjeldab **halba probleemi lahendust ning ühtlasi kuidas paremini lahendada**)
  - CREATE TABLE Tellimus_2009_01 (…);
  - CREATE TABLE Tellimus_2009_02 (…);
  - …
- Probleemid järgmistes valdkondades:
  - Kitsenduste jõustamine (sh viidete terviklikkus), tabelite sünkroniseerimine, unikaalsuse tagamine üle tabelite, päringud ja andmemuudatused üle tabelite jne.

Tabeli sektsioonideks jagamine



Millal kasutada tabelite sektsioonideks jagamist?

- Tabelis on **palju andmeid** (suurus üle 2GB)
- Tabelis on **ajaloolisi andmeid**, mida ainult loetakse, kuid ei muudeta
- Tabelis olevad andmed soovitatakse salvestada **erinevaid mäluseadmeid** kasutades
  - Aktuaalsed andmed peavad olema kiiresti leitavad, kuid ajalooliste andmete puhul pole see nii oluline

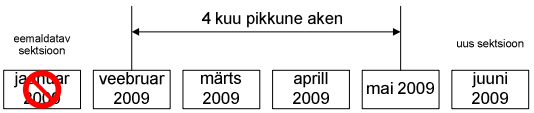
Tabelite sektsioonideks jagamise eelised (Oracle)

- Andmete **otsimine** kindlast sektsioonist
  - SELECT * FROM Tellimus WHERE loomise_aeg BETWEEN '2009-02-02' AND '2009-02-21';
  - SELECT * FROM Tellimus PARTITION (veebruar_2009);
    - Halvem, sest päringu tegija peab teadma tabeli sisemist ülesehitust – järelikult muutub sisemise ülesehituse muutmine keerukamaks.
- Sektsioonide **paigutamine** erinevatesse failidesse, erinevates mäluseadmetes (erinevad sektsioonid paiknevad erinevates tabeliruumides)
- **Indeksi** loomine eraldi igale sektsioonile

Tabelite sektsioonideks jagamise eelised (Oracle) (2)

- Sektsiooni kuuluvate andmete **pakkimine**
- Andmete **varundamine/taastamine** sektsiooni tasemel
- Andmete **import/eksport** sektsiooni tasemel
- Kaudne **klasterdamine** – sarnaste sektsiooni võtmetega read paiknevad ühes sektsioonis e füüsiliselt lähestikku (vahemikul või väärtuste nimekirjal põhineva sektsioonideks jagamise korral)
- Sektsiooni kuuluvate andmete kiire **kustutamine**

Sektsioonide kasutamine ajalooliste andmete hoidmisel



- ALTER TABLE Tellimus DROP PARTITION jaanuar_2009;
- DROP INDEX idx_jaanuar_2009 ON Tellimus;

Kuidas määrata, millisesse sektsiooni rida kuulub? (Oracle)

- Väärtuste **vahemikel** põhinev sektsioonideks jagamine
- Väärtuste **nimekirjal** põhinev sektsioonideks jagamine
- **Räsifunktsioonil** põhinev sektsioonideks jagamine
  - CREATE TABLE Invoice (...) PARTITION BY HASH (**invoice_no**) PARTITIONS 4 STORE IN (users, users, users, users);
  - Optimaalseks andmete salvestamiseks peab sektsioonide arv olema kahe aste.
  - Jaotab read ühtlaselt erinevate sektsioonide vahel.

Tabelite sektsioonideks jagamine – näide (Oracle)

CREATE TABLE dept
(deptno NUMBER(2),
dept_name VARCHAR2(30))
PARTITION BY RANGE (deptno)
(PARTITION d1 VALUES
LESS THAN (10) TABLESPACE dept1,
PARTITION d2 VALUES
LESS THAN (20) TABLESPACE dept2,
PARTITION d3 VALUES
LESS THAN (maxvalue) TABLESPACE dept3);

Väärtuste vahemikel põhinev sektsioonideks jagamine – kolm sektsiooni

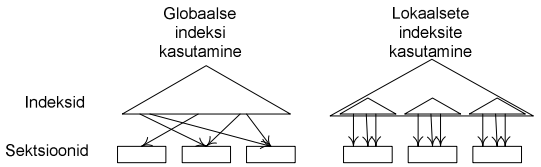
Sisuliselt toimus andmete osakonna numbrite alusel klasterdamine.

Edasi

Kuidas määrata, millisesse sektsiooni rida kuulub? (Oracle) (2)

- **Välisvõtme** alusel sektsioonideks jagamine
  - Klient (klient_id, seisund) Primaarvõti (klient_id)
    - Sektsioonideks jaotatud veerus *seisund* olevate võimalike väärtuste alusel.
  - Arve (arve_nr, klient_id, loomise_kuup)
    Primaarvõti (arve_nr)
    Välisvõti (klient_id) Viitab Klient (klient_id)
    - Sama palju sektsioone kui tabelis *Klient*. Rida paigutatakse sektsiooni vastavalt sellele, millise kliendiga on arve seotud.

Globaalne indeks *vs*. lokaalsed indeksid



- Lokaalsed indeksid on väiksema mahuga.
- Lokaalse indeksi kasutamise korral ei mõjuta ühes sektsioonis toimuv andmemuudatus/sektsiooni kustutamine teiste sektsioonide kasutamist.

Tabelite sektsioonideks jagamine (PostgreSQL)

- Kuni ver 9.6 tuli ise programmeerida.
- Alates ver 10 saab luua **deklaratiivselt**.
  - Toetatud **väärtuste nimekirjal** ja **väärtuste vahemikel** põhinev sektsioonideks jagamine.
  - Tuleb deklareerida põhitabel ning vahemikel põhineva sektsioonideks jagamise korral kindlasti ka vahemikele vastavad sektsioonid.
  - Versioonis 10 palju piiranguid.
    - https://wiki.postgresql.org/wiki/Table_partitioning

Tabelite sektsioonideks jagamine (PostgreSQL kuni 9.6)

- Tuleb ise programmeerida!
- CREATE TABLE Dept (deptno SMALLINT PRIMARY KEY, dept_name VARCHAR(30));
- Loon sektsioonid kasutades tabelite pärimise võimalust:
  - CREATE TABLE dept_10 (CHECK (deptno<10)) INHERITS (Dept);
  - CREATE TABLE dept_10_20 (CHECK (deptno BETWEEN 10 AND 20)) INHERITS (Dept);
  - CREATE TABLE dept_max (CHECK (deptno>20)) INHERITS (Dept);

Näide demonstreerimaks PostgreSQL programmeerimise võimalusi.

Tabelite sektsioonideks jagamine (PostgreSQL kuni 9.6) (2)

- Lokaalsed indeksid sektsioonidele:
- CREATE INDEX idx_dept_10 ON dept_10 (deptno);
- CREATE INDEX idx_dept_10_20 ON dept_10_20 (deptno);
- CREATE INDEX idx_dept_max ON dept_max (deptno);

PostgreSQL – reeglid

- Reegel võimaldab andmekäitluskeele (SELECT/INSERT/UPDATE/DELETE) lause käivitamisel täita täiendavaid/asendavaid tegevusi.

Tabelite sektsioonideks jagamine (PostgreSQL kuni 9.6) (3)

- Reeglid: kui lisan põhitabelisse rea, siis lisada see hoopis mõnda alamtabelisse (sektsiooni)
- CREATE RULE r_dept_10 AS ON INSERT TO Dept WHERE NEW.deptno<10 **DO INSTEAD** INSERT INTO dept_10(deptno, dept_name) VALUES (NEW.deptno, NEW.dept_name);

DO INSTEAD reegli asemel võib kasutada ka INSTEAD OF trigerit.

Tabelite sektsioonideks jagamine (PostgreSQL kuni 9.6) (4)

- CREATE RULE r_dept_10_20 AS ON INSERT TO Dept WHERE NEW.deptno BETWEEN 10 AND 20 **DO INSTEAD** INSERT INTO dept_10_20(deptno, dept_name) VALUES (NEW.deptno, NEW.dept_name);
- CREATE RULE r_dept_max AS ON INSERT TO Dept WHERE NEW.deptno>20 **DO INSTEAD** INSERT INTO dept_max (deptno, dept_name) VALUES (NEW.deptno, NEW.dept_name);

Tabelite sektsioonideks jagamine (PostgreSQL kuni 9.6) (5)

- INSERT INTO Dept(deptno, dept_name) VALUES (1, 'rahandus'), (2, 'personal'), (21, 'tööohutus');
- ANALYZE;--Statistika kogumine
- /*Päringu täitmisel kasutada vaid vajalikke sektsioone – leitakse kitsenduste alusel*/
- SET constraint_exclusion = on;
- SELECT Count(*) AS arv FROM Dept **WHERE deptno>15**;

Tabelite sektsioonideks jagamine (PostgreSQL kuni 9.6) (6)

```
erki=# SET constraint_exclusion = on;
SET
erki=# EXPLAIN SELECT Count(*) AS arv FROM Dept WHERE deptno>15;
                                    QUERY PLAN
-----------------------------------------------------------------------
 Aggregate  (cost=40.75..40.76 rows=1 width=0)
   ->  Append  (cost=0.00..39.51 rows=495 width=0)
         ->  Seq Scan on dept  (cost=0.00..19.25 rows=247 width=0)
               Filter: (deptno > 15)
         ->  Seq Scan on dept_10_20 dept  (cost=0.00..19.25 rows=247 width=0)
               Filter: (deptno > 15)
         ->  Seq Scan on dept_max dept  (cost=0.00..1.01 rows=1 width=0)
               Filter: (deptno > 15)
(8 rows)
```

```
erki=# SET constraint_exclusion = off;
SET
erki=# EXPLAIN SELECT Count(*) AS arv FROM Dept WHERE deptno>15;
                                    QUERY PLAN
-----------------------------------------------------------------------
 Aggregate  (cost=41.78..41.79 rows=1 width=0)
   ->  Append  (cost=0.00..40.54 rows=496 width=0)
         ->  Seq Scan on dept  (cost=0.00..19.25 rows=247 width=0)
               Filter: (deptno > 15)
         ->  Seq Scan on dept_10 dept  (cost=0.00..1.02 rows=1 width=0)
               Filter: (deptno > 15)
         ->  Seq Scan on dept_10_20 dept  (cost=0.00..19.25 rows=247 width=0)
               Filter: (deptno > 15)
         ->  Seq Scan on dept_max dept  (cost=0.00..1.01 rows=1 width=0)
               Filter: (deptno > 15)
(10 rows)
```

Tabelite sektsioonideks jagamine (PostgreSQL kuni 9.6) (7)

- Üksiku sektsiooni kustutamine väga kiire:
  - DROP TABLE dept_10;
  - DROP RULE r_dept_10 ON Dept;
- Sektsiooni saab tabelist eemaldada ja säilitada:
  - ALTER TABLE dept_10 NO INHERIT Dept;
  - DROP RULE r_dept_10 ON Dept;

Tabelite sektsioonideks jagamine PostgreSQLis (kuni 9.6) – kriitika

- Andmebaasi *sisemise taseme* konstruktsiooni (sektsioonid) üritatakse realiseerida andmebaasi *kontseptuaalsel tasemel* (kasutades tabeleid), muutes sellega kasutajatele nähtavaks asjad, millest ei peaks tal tegelikult aimugi olema.
- Alternatiiv, millel sama probleem – laiendus *tab_tier*.(http://bonesmoses.org/2016/05/20/pg-phriday-trusty-table-tiers/)

Tabelite sektsioonideks jagamine PostgreSQLis (alates 10.0)

- CREATE TABLE Dept (deptno SMALLINT, dept_name VARCHAR(30)) **PARTITION BY RANGE (deptno)**;
- CREATE TABLE dept_10 *PARTITION OF Dept* FOR VALUES FROM (**MINVALUE**) TO (9);
- CREATE TABLE dept_10_20 *PARTITION OF Dept* FOR VALUES FROM (10) TO (20);
- CREATE TABLE dept_max *PARTITION OF Dept* FOR VALUES FROM (21) TO (**MAXVALUE**);

Tabelite sektsioonideks jagamine PostgreSQLis (alates 10.0) (2)

```
erki=# EXPLAIN SELECT Count(*) AS arv FROM Dept WHERE deptno>15;
                                QUERY PLAN
-------------------------------------------------------------------------
 Aggregate  (cost=21.02..21.03 rows=1 width=8)
   ->  Append  (cost=0.00..20.39 rows=251 width=0)
         ->  Seq Scan on dept_10_20  (cost=0.00..19.38 rows=250 width=0)
               Filter: (deptno > 15)
         ->  Seq Scan on dept_max  (cost=0.00..1.01 rows=1 width=0)
               Filter: (deptno > 15)
(6 rows)
```

Loetakse vaid sektsioone, mida on vaja päringu täitmiseks (sektsiooni *dept_10* ei loeta).

Virtuaalne veerg (Oracle)

- Tabeli T veerg, milles olevad väärtused ei ole salvestatud kettale, vaid on leitud T teiste mittevirtuaalsete veergude põhjal.
- Neid veerge saab indekseerida, kuid nendes ei saa otse andmeid muuta.

Virtuaalne veerg (Oracle) (2)

- CREATE TABLE Aken(aken_id NUMBER(3) PRIMARY KEY, laius NUMBER(3) NOT NULL, korgus NUMBER(3) NOT NULL, pindala NUMBER(6) GENERATED ALWAYS AS (laius*korgus));
- INSERT INTO Aken(aken_id, korgus, laius) VALUES (1, 200, 100);

```
C##TUD1@ORCL > SELECT * FROM Aken;

   AKEN_ID      LAIUS     KORGUS    PINDALA
---------- ---------- ---------- ----------
         1        100        200      20000
```

Nähtamatu veerg (Oracle)

- Kasutaja näeb selles veerus olevaid andmeid vaid siis, kui taolisele veerule selle nime kasutades viitab.
  - SELECT * FROM … päringu tulemuses selliseid veerge ei ole.
- Saab kasutada andmebaasi ja rakenduse evolutsioneerimiseks.
  - Lisa nähtamatu veerg, muuda tabelit kasutavat koodi, muuda veerg nähtavaks,

Nähtamatu veerg (Oracle) (2)

- CREATE TABLE Aken(aken_id NUMBER(3) PRIMARY KEY, laius NUMBER(3) NOT NULL, korgus NUMBER(3) NOT NULL, pindala NUMBER(6) INVISIBLE GENERATED ALWAYS AS (laius*korgus));
- INSERT INTO Aken(aken_id, korgus, laius) VALUES (1, 200, 100);

```
C##TUD1@ORCL > SELECT * FROM Aken;

   AKEN_ID      LAIUS     KORGUS
---------- ---------- ----------
         1        100        200

C##TUD1@ORCL > SELECT aken_id, korgus, laius, pindala FROM Aken;

   AKEN_ID     KORGUS      LAIUS    PINDALA
---------- ---------- ---------- ----------
         1        200        100      20000
```


Ajutised tabelid

- **PostreSQL**
  - Tabel luuakse sessiooni käigus, spetsiaalses ajutiste andmete skeemis
  - Tabel kustutatakse transaktsiooni/sessiooni lõpus
  - Tabel on "nähtav" vaid sessioonis, kus loodi
- **Oracle**
  - Tabel luuakse enne sessiooni algust, kasutaja skeemis
  - Tabelit "näevad" kõik sessioonid
  - Sessiooni *s* käigus saab tabelisse panna *andmeid*, mida näeb vaid sessioonis *s* ja mis kustutatakse sessiooni *s* lõpus
  - Tabel säilib ka peale sessiooni *s* lõppu

XML tüüpi tabel (Oracle)

- CREATE TABLE xoppeaine OF XMLTYPE;
- INSERT INTO xoppeaine VALUES

```
(xmltype( '<?xml version="1.0"?> <Oppeained>
 <Oppeaine>
   <oppeaine_id>1</oppeaine_id>
   <oppeaine_kood>IDU3381</oppeaine_kood>
   <nimetus>Andmebaaside projekteerimine</nimetus>
 </Oppeaine>
 <Oppeaine>
   <oppeaine_id>2</oppeaine_id>
   <oppeaine_kood>IDU3382</oppeaine_kood>
   <nimetus>Andmebaaside projekteerimine-projekt</nimetus>
 </Oppeaine> </Oppeained>'));
```

- SELECT a.getClobVal() FROM xoppeaine a;

XML tüüpi tabel (Oracle) (2)

```
SELECT xt.nimetus
FROM xoppeaine yt,
XMLTable('Oppeained/Oppeaine'
PASSING yt.SYS_NC_ROWINFO$
COLUMNS
nimetus  VARCHAR2(100) PATH 'nimetus') xt;
```

Nii PostgreSQLis kui Oracles saab luua tabelites veerge, mis on XML tüüpi.

```
NIMETUS
----------------------------------------
Andmebaaside projekteerimine
Andmebaaside projekteerimine-projekt
```


Väline tabel (Oracle)

- Andmebaasisüsteemi poolt mittehallatavas failis olevaid andmeid näeb andmebaasis tabeli kujul.
- Sellest tabelist saab andmeid *lugeda*, kuid olemasolevas välises tabelis ei saa andmeid *muuta*.
- Päringu tulemusest on võimalik luua väline tabel ning ühtlasi *laadida* päringu tulemused *faili*.
- Saab kasutada andmete laadimiseks andmebaasi ja sealt välja.
  - CREATE OR REPLACE DIRECTORY *data_dir* as '/tmp/';
  - GRANT READ, WRITE ON DIRECTORY data_dir TO c##Yliopilane;

Väline tabel (Oracle) (2)

```
CREATE TABLE External_Aine
(aine_kood CHAR(7),
nimetus VARCHAR2(50))
ORGANIZATION EXTERNAL
( TYPE oracle_loader
DEFAULT DIRECTORY data_dir
ACCESS PARAMETERS
(RECORDS DELIMITED BY
    NEWLINE
FIELDS TERMINATED BY '/' )
LOCATION ('Aine_t990999.dat')
) REJECT LIMIT UNLIMITED;
```

1.Aine_t990999.dat

```
IDU0220/Andmebaasid I
IDU0230/Andmebaasid II
```

```
TUD1@ORCL > SELECT * FROM External_aine;

AINE_KO NIMETUS
------- --------------------------------
IDU0220 Andmebaasid I
IDU0230 Andmebaasid II
```


Väline tabel tekstifaili põhjal (PostgreSQL)

- Andmebaasisüsteemi poolt mittehallatavas tekstifailis olevaid andmeid näeb andmebaasis tabeli kujul.
- Sellest tabelist saab andmeid *lugeda*, kuid olemasolevas failil põhinevas välises tabelis ei saa andmeid *muuta*.
- Saab kasutada andmete laadimiseks andmebaasi.
  - CREATE EXTENSION file_fdw;
  - CREATE SERVER file_fdw_server FOREIGN DATA WRAPPER file_fdw;

Väline tabel tekstifaili põhjal (PostgreSQL) (2)

- CREATE FOREIGN TABLE External_Aine (aine_kood CHAR(7), nimetus VARCHAR(50)) SERVER file_fdw_server OPTIONS (format 'csv',header 'false', filename '/tmp/Aine_t990999.dat', delimiter '/', null '');

1.Aine_t990999.dat

```
IDU0220/Andmebaasid I
IDU0230/Andmebaasid II
```

```
erki=# SELECT * FROM External_aine;
 aine_kood |    nimetus
-----------+---------------
 IDU0220   | Andmebaasid I
 IDU0230   | Andmebaasid II
(2 rows)
```


Väline tabel (PostgreSQL) (3)

- Oracle välised tabelid saavad vahendada ainult tekstifailides või Oracle enda firmaomastes failides olevaid andmeid
- PostgreSQL väliste tabelite mehhanism võimaldab **juurdepääsu erinevat tüüpi failide sisule,** andmetele, mis on **teistes andmebaasides** (võimalik, et teine andmebaasisüsteem/andmemudel), **veebis**, **infokataloogides** jne.

Osadel juhtudel saab andmeid ka muuta.

Tabeli loomine teise tabeli põhjal (PostgreSQL)

- CREATE TABLE Tootaja (LIKE Isik INCLUDING DEFAULTS INCLUDING CONSTRAINTS INCLUDING INDEXES);
- Uude tabelisse kantakse ka üle lähtetabeli:
  - vaikimisi väärtused,
  - kitsendused,
  - indeksid.
- Andmeid üle ei kanta.

Saab ka määrata, et uude tabelisse tuleb üle kanda lähtetabeli identiteedi omadus, veergude salvestusparameetrite väärtused ja lähtetabeliga seotud kommentaarid.

Deklaratiivsed kitsendused

Oracle	PostgreSQL
•PRIMARY KEY, UNIQUE, NOT NULL, FOREIGN KEY, CHECK •CHECK kitsenduses ei saa kasutada alampäringuid	•PRIMARY KEY, UNIQUE (alates PostgreSQL 9.0 lisaks selle üldistus *exclusion constraint*), NOT NULL, FOREIGN KEY, CHECK •CHECK kitsenduses ei saa kasutada alampäringuid •Võrreldes Oraclega rohkem võimalikke tegevusi viidete terviklikkuse vea korral

Välisvõtmete kompenseerivad tegevused

Oracle	PostgreSQL
ON DELETE CASCADE ON DELETE SET NULL Kasutamaks NO ACTION, tuleb seda fraasi mitte kasutada. CREATE TABLE Emp_tab (... FOREIGN KEY (deptno) REFERENCES Dept_tab);	ON DELETE NO ACTION ON DELETE RESTRICT ON DELETE CASCADE ON DELETE SET NULL ON DELETE SET DEFAULT ON **UPDATE** NO ACTION ON **UPDATE** RESTRICT ON **UPDATE** CASCADE ON **UPDATE** SET NULL ON **UPDATE** SET DEFAULT

Deklaratiivsed kitsendused (2)

Oracle	PostgreSQL
•Võimalik on kitsenduse kontrolli lükkamine transaktsiooni lõppu	•Võimalik on kitsenduse kontrolli lükkamine transaktsiooni lõppu (FOREIGN KEY kitsenduse korral ning alastes PostgreSQL 9.0 ka UNIQUE, EXCLUDE ja PRIMARY KEY kitsenduse korral)

Vaade (virtuaalne tabel)

Oracle	PostgreSQL
Saab luua. Läbi teatud tingimustele vastavate vaadete on võimalik muuta andmeid baastabelites. Vaadetega saab siduda INSTEAD OF *trigereid*, et realiseerida andmemuudatusi vaadete kaudu, kus see vaikimisi pole võimalik.	Saab luua. Läbi „lihtsate“ ühel baastabelil põhinevate vaadete on võimalik baastabelites andmeid muuta. Vaadetega saab siduda INSTEAD OF *trigereid* või DO INSTEAD *reegleid*, et realiseerida andmemuudatusi vaadete kaudu, kus see vaikimisi pole võimalik.

Vaade Oracle näitel

- CREATE OR REPLACE VIEW Aken_pindala AS SELECT aken_id, korgus, laius, korgus*laius AS pindala FROM Aken WITH CHECK OPTION;
  - Alternatiiv virtuaalse veeru kasutamisele
- INSERT INTO Aken_pindala(aken_id, korgus, laius) VALUES (2, 400, 300);

WITH CHECK OPTION kitsendus

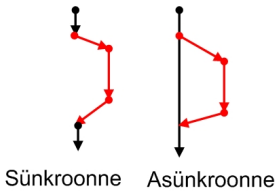
- Tagab, et vaade pole "must auk", mis "neelab" ridu.
- Tagab, et ei saa juhtuda et,
  - rea lisamisel vaatesse ei ilmu see vaates nähtavale,
  - rea muutmisel vaates kaob see vaatest ära.
- Tungiv soovitus: kasutage WITH CHECK OPTION kõigi vaadete korral, mille kaudu põhimõtteliselt saab andmeid muuta.

Hetktõmmis (materialiseeritud vaade)

Oracle	PostgreSQL
jah Võimalik nii täielik kui kiire värskendamine. Värskendamise sagedus määratakse hetktõmmise spetsifikatsioonis. Võimalik päringute automaatne ümberkirjutamine.	jah Võimalik vaid täielik värskendamine – selleks eraldi lause (REFRESH MATERIALIZED VIEW). Automaatseks värskendamiseks võib kasutada Unixi tööplaanurit *cron*. Päringuid ümber ei kirjutata.

Hetktõmmis (Oracle) – sünkroniseerimise aeg

- Kohe, baastabelis andmeid muutes (sünkroonne)
- Perioodiliselt (asünkroonne)
  - Protsessid sõltuvad üksteisest vähem, saavad toimuda iseseisvalt, seega sageli eelistatud



Hetktõmmis (Oracle) – sünkroniseerimise ulatus

- **Täielik** värskendamine – kogu hetktõmmis tehakse uuesti
- **Kiire** värskendamine – värskendatakse vaid muutunud osa
  - Piirangud hetktõmmise alampäringule – peab olema "lihtne"
  - Muudatusi tuleb logida

Hetktõmmise logi (Oracle)

- Mõeldud hetktõmmise aluseks olevates tabelites toimunud muudatuste logimiseks.
- Aitavad läbi viia kiiret värskendamist.
- CREATE MATERIALIZED VIEW LOG ON Emp WITH (deptno) INCLUDING NEW VALUES;
- CREATE MATERIALIZED VIEW LOG ON Dept WITH PRIMARY KEY INCLUDING NEW VALUES;

Hetktõmmis (Oracle) (2)

```
CREATE MATERIALIZED  VIEW Dept_with_emps
/*Hetktõmmis tuleb kohe luua*/
BUILD IMMEDIATE
/*Hetktõmmist tuleb värskendada kord päevas – kell 11 õhtul, kasutades
  kiiret meetodit*/
REFRESH FAST START WITH sysdate NEXT trunc(sysdate) + 1 +
  23/24
/*Andmebaasisüsteem võib kirjutada SELECT lauseid ümber nii, et
  nende täitmiseks kasutatakse seda hetktõmmist*/
ENABLE QUERY REWRITE AS
SELECT D.deptno, D.dname FROM Dept D WHERE EXISTS
  (SELECT E.empno FROM Emp E WHERE D.deptno=E.deptno);
```


Päringu ümberkirjutamine hetktõmmise põhjal

```
TUD1@SQL> set autotrace on
TUD1@SQL> SELECT D.deptno, D.dname FROM Dept D WHERE EXISTS (SELECT E.empno FROM Emp E WHERE D.dept
no=E.deptno);

    DEPTNO DNAME
---------- --------------
        10 ACCOUNTING
        20 RESEARCH
        30 SALES


Execution Plan
----------------------------------------------------------
Plan hash value: 1860959842

-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                 |     3 |    66 |     3   (0)| 00:00:01 |
|   1 |  MAT_VIEW REWRITE ACCESS FULL| DEPT_WITH_EMPS  |     3 |    66 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------
```


Skeem, tabeliruum, domeen, sünonüüm

	Oracle	PostgreSQL
Tabeliruum	jah	jah
Skeem	jah	jah
Domeen	ei	jah
Assertion (Üldine kitsendus)	ei	ei
Sünonüüm	jah	ei

Edasi

Skeem (Oracle)

- CREATE SCHEMA AUTHORIZATION **c##Ylikool** CREATE TABLE Aine (...) CREATE TABLE Yliopilane (...);
  - Kasutaja *c##Ylikool* käivitab selle lause, et lisada enda skeemi tabelid. Kasutaja *c## Ylikool* ja skeem *c##Ylikool* olid olemas enne lause käivitamist.
  - Kui kasvõi ühe lauses sisalduva CREATE või GRANT lause täitmine ebaõnnestub, siis jäetakse terve CREATE SCHEMA lause täitmata. Võimaldab koondada skeemiobjektide loomise ja õiguste jagamise laused üheks loogiliseks tervikuks.
  - Oracles luuakse skeem kasutaja loomisel.

Skeem (PostgreSQL)

- CREATE SCHEMA **Ylikool** AUTHORIZATION **erki** CREATE TABLE Aine (...) CREATE TABLE Yliopilane (...);
  - Luuakse skeem *Ylikool* ja selles vastavad tabelid. Skeemi omanikuks saab kasutaja *erki*.
  - Kui kasvõi ühe lauses sisalduva CREATE või GRANT lause täitmine ebaõnnestub, siis jäetakse terve CREATE SCHEMA lause täitmata. Võimaldab koondada skeemiobjektide loomise ja õiguste jagamise laused üheks loogiliseks tervikuks.
- PostgreSQL andmebaasi loomisel luuakse automaatselt skeem nimega *public*.

Tabeliruum

- Andmefailide hulk (kus üks või rohkem faili), milles salvestatakse andmebaasis hoitavate andmete alamhulk.
- Lihtsustab andmebaasi administreerimist.

Milleks tabeliruume vaja läheb (Oracle)

- Võimaldab määrata, millisesse **faili** ja **mäluseadme** abil tabelite ja indeksite andmed salvestada.
- Andmebaasis erineva **suurusega plokkide** kasutamine.
- Andmebaasi kasutajatele maksimaalse lubatud **salvestusruumi** määramine (määrates kasutajale vaikimisi tabeliruumi).
- Tabeliruumi saab viia **offlain** režiimi (selles olevad andmed pole kättesaadavad) ilma tervet andmebaasi eksemplari sulgemata.

Milleks tabeliruume vaja läheb (Oracle) (2)

- Andmebaasi **füüsiline varundamine/ taastamine** tabeliruumide kaupa.
- Andmebaasi **eksport/import** (loogiline varundamine) tabeliruumide kaupa.
- Tabeliruumi **krüpteerimine**.
- Tabeliruumile **kirjutuskaitse** määramine.
- Tabeliruumi **transportimine** ühest andmebaasist teise.

Tabeliruum (PostgreSQL)

- CREATE TABLESPACE ruum1 LOCATION '/data/dbs';
  - LOCATION '/data/dbs' – tühi kataloog, kuhu paigutatakse tabeliruumi kuuluvad failid.
  - Tabeliruumid on ühised üle kõigi serveril olevate PostgreSQL andmebaaside (nimi peab olema serveril unikaalne).
- CREATE TABLE Isik (isikukood CHAR(11) PRIMARY KEY, perenimi VARCHAR(30) NOT NULL) TABLESPACE ruum1;

Tabeliruum (Oracle)

- CREATE TABLESPACE ruum1 DATAFILE 'diskb:tbs_f5.dat' SIZE 500K REUSE AUTOEXTEND ON NEXT 500K MAXSIZE 100M;
  - Üks andmefail (diskb:tbs_f5.dat) esialgse suurusega 500KB.
  - Kui andmefail saab täis, siis seda suurendatakse 500KB ekstentide kaupa kuni suuruseni 100MB.

Domeen (PostgreSQL)

- CREATE DOMAIN d_nimetus AS VARCHAR(30) NOT NULL DEFAULT 'teadmata' CONSTRAINT chk_nimetus_ei_tohi_olla_tyhi_koosneda_tyhikutest CHECK (trim(**VALUE**)<>'');
  - Nimetus ei tohi olla tühi string ega koosneda ainult tühikutest.
  - Kasutan kõigis klassifikaatorite tabelites, et nimetused oleksid samade omadustega veergudes ja omaduste muutmiseks oleks üks koht.

Domeen (PostgreSQL) (2)

- CREATE TABLE Hindamisviis(
  ... nimetus d_nimetus,
  ...);
- CREATE TABLE Aine_seisundi_liik(
  ... nimetus d_nimetus,
  ...);

Mida rohkem kasutan, seda rohkem domeeni loomiseks mõeldud pingutus ennast ära tasub!

Domeeni probleem

- CREATE DOMAIN d_telefon AS VARCHAR (20) NOT NULL;
- CREATE TABLE Isik (... telefon d_telefon ...);
- Järgnev päring SQLis töötab, sest mõlemad domeenid põhinevad VARCHAR tüübil
  - SELECT * FROM Isik I, Hindamisviis H WHERE **H.nimetus=I.telefon**;
- SQL-domeen ei ole tüüp!
- Domeenide kasutamine ei võimalda tugevat tüüpimist!!

Tugev tüüpimine

- Igal väärtusel, avaldisel ja muutujal on tüüp.
- Iga kord, kui proovime teha väärtuse, avaldise või tüübiga mõne operatsiooni, kontrollib süsteem, kas osalised on operatsiooni läbiviimiseks sobivat tüüpi.
  - Operatsiooni läbiviimiseks peab leiduma sobivat tüüpi parameetritega operaator.

Tugev tüüpimine – näide

- CREATE TYPE vanus AS ENUM ('noor', 'vana');
- CREATE TYPE emakeel AS ENUM ('eesti','inglise');
- CREATE TABLE Isik (isik_id INTEGER PRIMARY KEY, vanus **vanus** NOT NULL, emakeel **emakeel** NOT NULL);
- SELECT * FROM Isik WHERE vanus=emakeel;
  - ERROR: operator does not exist: vanus = emakeel

Tugev tüüpimine – näide (2)

CREATE OR REPLACE FUNCTION f_leia_isik() RETURNS TABLE (isik_id INTEGER, vanus vanus, emakeel emakeel) AS
$$ SELECT isik_id, vanus, emakeel
FROM Isik WHERE vanus=emakeel;
$$ LANGUAGE sql STABLE LEAKPROOF;

- ERROR: operator does not exist: vanus = emakeel
- LINE 3: FROM Isik WHERE vanus=emakeel;
- ^
- HINT: No operator matches the given name and argument type(s). You might need to add explicit type casts.

Mida teha?

- Luua uus operaator võrdluse läbiviimiseks.
  - CREATE OPERATOR ...
- Teostada ilmutatud tüübiteisendus, et kasutada olemasolevat operaatorit.
  - SELECT * FROM Isik WHERE CAST(vanus AS VARCHAR)=CAST(emakeel AS VARCHAR);
- Loobuda operatsiooni läbiviimisest.

Sünonüüm e alias (Oracle)

- Sünonüüm on alternatiivne nimi tabelile, vaatele, arvujada generaatorile, operaatorile, protseduurile, funktsioonile, paketile, materialiseeritud vaatele, kasutaja-defineeritud objektitüübile või teisele sünonüümile.

Avalik sünonüüm (Oracle)

- Ligipääsetav kõigile kasutajatele
- Avalik sünonüüm lihtsustab kasutaja *k* pöördumist väljaspool kasutaja *k* skeemi olevate skeemiobjektide poole
- Enne:
  - SELECT * FROM K1.T1;
- CREATE PUBLIC SYNONYM **Tab1** FOR K1.T1;
- Pärast:
  - SELECT * FROM **Tab1**;
  - Kasutaja ei pea teadma skeemiobjekti asukohta (skeemi)
  - Kasutajal peab olema SELECT õigus tabeli K1.T1 suhtes – muidu päring ei õnnestu

Mitteavalik sünonüüm (Oracle)

- Seotud konkreetse skeemiga ja nimi peab olema skeemi piires unikaalne.
- Võib kasutada skeemiobjektide ümbernimetamise lihtsustamiseks.
  - Abistab andmebaasi refaktoreerimisel.
- CREATE SYNONYM Oppeaine FOR Aine;
- DROP SYNONYM Oppeaine;
- ALTER TABLE Aine RENAME TO Oppeaine;

Unikaalsete väärtuste genereerimine

Oracle	PostgreSQL
Rea füüsiline aadress – ROWID	Rea süsteemi poolne identifikaator (täisarv) – OID (kasutamise saab määrata iga tabeli korral eraldi)
Arvujada generaator	Arvujada generaator
GENERATED AS IDENTITY omadusega veerg	SMALLSERIAL, SERIAL ja BIGSERIAL notatsioon Alates PostgreSQL 10 GENERATED AS IDENTITY omadusega veerg

Arvujada generaator (PostgreSQL)

- CREATE SEQUENCE seq_tootaja INCREMENT 5 START 5 OWNED BY Tootaja.tootaja_id;
- ALTER TABLE Tootaja ALTER COLUMN tootaja_id SET **DEFAULT nextval('seq_tootaja')**;
- *nextval* – funktsioon, mille abil saab küsida arvujada generaatorilt uue väärtuse.

SERIAL notatsioon (PostgreSQL)

- CREATE TABLE Tootaja (tootaja_id ***SERIAL*** CONSTRAINT pk_tootaja PRIMARY KEY,...);
- Veerg *tootaja_id* on tüüpi INTEGER.
- Andmebaasisüsteem loob ise arvujada generaatori objekti ja seostab baastabeliga.
- Generaatori omanikuks saab *Tootaja.tootaja_id* – Tabeli *Tootaja* või veeru *tootaja_id* kustutamisel, kustutatakse ka arvujada generaatori objekt.

Arvujada generaator (Oracle)

```
CREATE SEQUENCE seq_tootaja;

CREATE TABLE Tootaja ( tootaja_id NUMBER( 10 ) CONSTRAINT
    pk_tootaja PRIMARY KEY, ...);

CREATE OR REPLACE TRIGGER trig_tootaja_insert_seq
BEFORE INSERT ON Tootaja
FOR EACH ROW
BEGIN
    :new.tootaja_id:=seq_tootaja.nextval;
END;
/
```

nextval – pseudoveerg

Arvujada generaator (Oracle) (alates Oracle 12c)

- CREATE TABLE Tootaja (tootaja_id NUMBER(10) **GENERATED AS IDENTITY** CONSTRAINT pk_tootaja PRIMARY KEY, …);
- ALTER TABLE Tootaja2 MODIFY (tootaja2_id **DEFAULT ON NULL seq_tootaja2.nextval**);

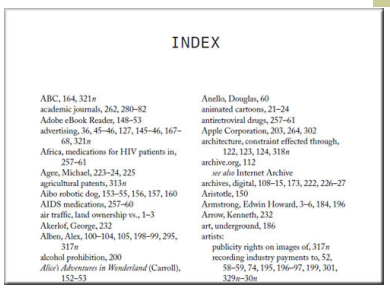
Indeksid

Oracle	PostgreSQL
◆(Unikaalne) B-puu indeks ◆Bitmap-indeks ◆Bitmap-join indeks ◆Võimalus luua programmeerijatel uut tüüpi indekseid (*domain indexes*) ◆Funktsioonil põhinev indeks ◆(Unikaalne) osaline indeks ◆Osade (mitte kõigi) tabeli sektsioonide indekseerimine	◆(Unikaalne) B-puu indeks ◆BRIN (Block Range Index) indeks ◆Hash indeks ◆GiST (Generalized Index Search Trees), SP-GiST, GIN (Generalized Inverted Index) indeksid – taristu, mis võimaldab realiseerida paljusid erinevaid indekseerimise strateegiaid ◆Funktsioonil põhinev indeks ◆(Unikaalne) osaline indeks

Indeksid (2)

Oracle	PostgreSQL
◆Pööratud indeks ◆Tekstimassiivide indekseerimine: *Oracle Text* ◆Lokaalsete indeksite loomine sektsioonidele ja sektsioonideks jaotatud indeks ◆Indeksile tabeliruumi määramine ◆Indeksite pakkimine ◆Nähtamatud indeksid ◆Indeksite kasutamise jälgimine ◆Indeksi uuesti ülesehitamine	◆Tekstimassiivide indekseerimine ◆Lokaalsete indeksite loomine sektsioonidele ◆Indeksile tabeliruumi määramine ◆Indeksite kasutamise jälgimine ◆Indeksi uuesti ülesehitamine

Raamatu indeks


INDEX

ABC, 164, 321n
academic journals, 262, 280–82
Adobe eBook Reader, 148–53
advertising, 36, 45–46, 127, 145–46, 167–68, 321n
Africa, medications for HIV patients in, 257–61
Agee, Michael, 223–24, 225
agricultural patents, 313n
Aibo robotic dog, 153–55, 156, 157, 160
AIDS medications, 257–60
air traffic, land ownership vs., 1–3
Akerlof, George, 232
Alben, Alex, 100–104, 105, 198–99, 295, 317n
alcohol prohibition, 200
Alice's Adventures in Wonderland (Carroll), 152–53
Anello, Douglas, 60
animated cartoons, 21–24
antiretroviral drugs, 257–61
Apple Corporation, 203, 264, 302
architecture, constraint effected through, 122, 123, 124, 318n
archive.org, 112
see also Internet Archive
archives, digital, 108–15, 173, 222, 226–27
Aristotle, 150
Armstrong, Edwin Howard, 3–6, 184, 196
Arrow, Kenneth, 232
art, underground, 186
artists:
publicity rights on images of, 317n
recording industry payments to, 52, 58–59, 74, 195, 196–97, 199, 301, 329n–30n

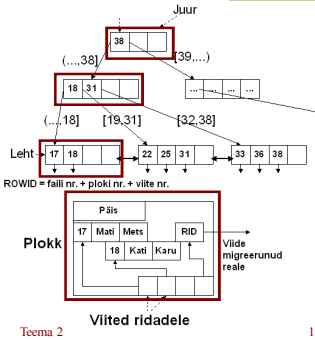
http://social.technet.microsoft.com/wiki/cfs-filesystemfile.ashx/__key/communityserver-wikis-components-files/00-00-00-00-05/7585.index_5F00_sample.jpg

B-puu (tasakaalustatud puu) indeks

Isik(isik_id, perenimi)
Primaarvõti(isik_id)

Nii PostgreSQL kui Oracle loovad automaatselt indeksi veerule *isik_id*.

SELECT * FROM Isik WHERE isik_id=17;



Osaline indeks (PostgreSQL, Oracle)

- Indeks luuakse ainult teatud tingimust rahuldavatele *ridadele*.
- Aktuaalsete õppeainete nimetused peavad olema unikaalsed.
- PostgreSQL
  - CREATE UNIQUE INDEX idx_aine_nimi ON Aine (nimetus) **WHERE on_aktuaalne=TRUE**;
- Oracle
  - CREATE UNIQUE INDEX ak_Aine_nimetus ON Aine (CASE WHEN on_aktuaalne=1 THEN nimetus ELSE NULL END);

Funktsioonil põhinev indeks (PostgreSQL, Oracle)

- CREATE INDEX yliopilane_perenimi_idx ON Yliopilane (**Upper**(perenimi));
- SELECT * FROM Yliopilane WHERE **Upper**(perenimi)='KASK';
- Kui veerul *perenimi* oleks "tavaline" indeks, siis eelnev päring seda indeksit ei kasutaks.

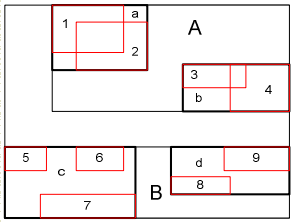
Edasi

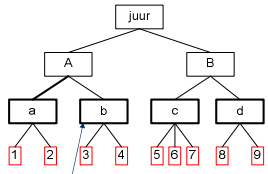
R-puu indeks (PostgreSQL) – näide

- PostgreSQLis on süsteemi-defineeritud tüübid ja funktsioonid geomeetriliste andmete hoidmiseks
  - CREATE TABLE Kast (kast_id serial CONSTRAINT pk_kast PRIMARY KEY, kuju box);
  - INSERT INTO Kast(kuju) VALUES ('(3,3),(4,4)');
  - INSERT INTO Kast(kuju) VALUES ('(2,3),(3,4)');
  - CREATE INDEX kuju_idx ON Kast USING **gist** (kuju);
    - R-puu indeks – kiirendab ruumiliste andmete otsimist.
    - PostgreSQL on R-puu indeks realiseeritud kasutades **GiST (Generalized Search Tree) taristut**.

R-puu indeks (PostgreSQL) – näide (2)

Päringu näide: Leia poed, mis asuvad etteantud poest 5 vähem kui 200 meetri kaugusel.





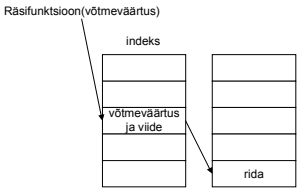
Minimal Bounding Rectangle – indeksipuus selle koordinaadid

Geomeetriliste päringute näited

- Leian kastide laiuse ja kõrguse
  - SELECT width(kuju) AS laius, height(kuju) AS korgus FROM Kast;
- Leian sellised kastid, mis on rangelt vasakul kastist '(4,4),(5,5)'.
  - SELECT * FROM Kast WHERE kuju<< box '(4,4),(5,5)';

Räsifunktsioonil põhinev indeks (PostgreSQL)

- CREATE INDEX idx_isik_isikukood ON Isik USING hash (isikukood);
- Indeksis viide rea asukohale.
- Räsifunktsiooni abil leitakse **VIITE asukoht**.
- Viite alusel leitakse rida.
- Efektiivse kasutuse jaoks peaks indeks (räsitabel) mällu mahtuma.



Räsifunktsiooni näide

for each *märk* in *string*

i:=leia_märgi_numbriline_kood(*märk*);

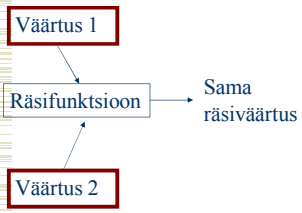
summa:=summa+i;

next;

tulemus:=summa mod b;

--b – plokkide arv, kuhu saab ridu paigutada

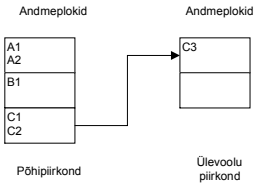
Põhiprobleem



- Räsitabelis *plokid*, milles on *pesad*
- Räsifunktsiooni alusel leitakse plokk, mille vabasse pessa paigutada väärtus
- Põhiprobleem – kuidas toimida, kui mitmetele erinevatele väärtustele vastab sama räsiväärtus (kokkupõrge) ja väärtused ei mahu ploki pesadesse ära?

Kokkupõrgete haldamine

- Staatiline räsitabel:
  - Open addressing
    - Lineaarne otsing
    - Kasvava intervalliga otsing
    - Korduv räsifunktsiooni rakendamine
  - Unchained overflow (overflow – ülevooluala)
  - Chained overflow
  - Multiple hashing with overwlow
- Dünaamiline räsitabel (suureneb dünaamiliselt):
  - Chaining



BRIN (plokkide vahemiku) indeks (PostgreSQL)

- Sisaldavad kokkuvõtteinfot plokkide vahemike kohta.
  - plokkides 1–20 on veeru x väärtused 1 kuni 10
  - plokkides 21–40 on veeru x väärtused 20 kuni 80
- Võimaldab kiiresti leida plokid, mis VÕIVAD sisaldada huvipakkuvaid andmeid ja välistada otsingust plokid, mis KINDLASTI neid andmeid ei sisalda.

BRIN indeks (PostgreSQL) (2)

- Töötab hästi, kui andmed indekseeritud veerus on kettal sorteeritud, mitte ei paikne plokkides läbisegi.
- Leia read, kus x väärtus on vahemikus 2 ja 8.

Loe plokke 1–20
- plokkides 1–20 on veeru x väärtused 1 kuni 10
- plokkides 21–40 on veeru x väärtused 20 kuni 80

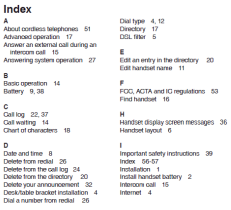
Loe plokke 1–40
- plokkides 1–20 on veeru x väärtused 1 kuni 70
- plokkides 21–40 on veeru x väärtused 6 kuni 80

BRIN indeks (PostgreSQL) (3)

- BRIN on konfigureeritav. Saab muuta detailsemaks, mis kasvatab suurust, aeglustab loomist ja uuendamist, kuid parandab veidi päringute kiirust.
- BRIN *vs*. B-puu
  - BRIN väiksem, kiirem luua, kiirem uuendada
  - B-puu parem päringute töökiirus, sest täpsem

BRIN *vs*. B-puu

BRIN indeks on nagu **dokumendikapp**. E-tähega algavad dokumendid on kõik ühes sahtlis, konkreetse dokumendi leidmiseks tuleb sahtli sisu läbi lehitseda. Dokumendikapist (ja BRIN indeksist) on kasu kui dokumendid (ja indekseeritavad andmed) on sorteeritud.



Index

A
About cordless telephones 51
Advanced operation 17
Answer an external call during an intercom call 15
Answering system operation 27

B
Basic operation 14
Battery 9, 38

C
Call log 22, 37
Call waiting 14
Chart of characters 18

D
Date and time 8
Delete from redial 26
Delete from the call log 24
Delete from the directory 20
Delete your announcement 32
Desk/table bracket installation 4
Dial a number from redial 26
Dial type 4, 12
Directory 17
DSL filter 5

E
Edit an entry in the directory 20
Edit handset name 11

F
FCC, ACTA and IC regulations 53
Find handset 16

H
Handset display screen messages 36
Handset layout 6

I
Important safety instructions 39
Index 56-57
Installation 1
Install handset battery 2
Intercom call 15
Internet 4

B-puu indeks on nagu **raamatu indeks**, mis sisaldab viiteid konkreetsetele lehekülgedele.

Pööratud indeks (Oracle)

- Tellimus (tellimus_id, loomise_aeg) Primaarvõti (tellimus_id)
  - Indeks veerul *tellimus_id*
  - Kui registreeritakse tellimus **20**, siis indeksisse "pööratud" võtmeväärtuse baidid: C1,15 asemel 15,C1
- **Eesmärk**: *hajutada* indeksi värskendamise koormus erinevate indeksi plokkide vahel
- **Probleem**: indeksit ei saa kasutada *vahemikul põhinevaks otsinguks (index range scan)*.
  - CREATE INDEX idx_Tellimus ON Tellimus(tellimus_id) REVERSE;
  - ALTER INDEX idx_Tellimus REBUILD REVERSE;

Edasi

Indeksi pakkimine (Oracle)

- Andmed indeksiplokis enne pakkimist:
  - 0 Mets ROWID
  - 1 Mets ROWID
  - 2 Salu ROWID
  - 3 Salu ROWID
  - 4 Salu ROWID
- Andmed indeksiplokis peale pakkimist:
  - **Prefiksi tabel**
  - 1 Mets
  - 2 Salu
  - **Indeks**
  - 0 1 ROWID
  - 1 1 ROWID
  - 2 2 ROWID
  - 3 2 ROWID
  - 4 2 ROWID

- Loomine:
  - CREATE INDEX idx_perenimi_compressed ON Isik(perenimi) COMPRESS;
- Eelised
  - Salvestusruumi kokkuhoid
  - Väheneb plokkide lugemiste arv
  - Rohkem andmeid mahub mällu
- Puudused
  - Aeglustab andmete muutmist
  - Selle alusel otsingu tegemine nõuab süsteemilt rohkem ressurssi

Tekstimassiivide indekseerimine (PostgreSQL)

- Alates PostgreSQL 8.3 on süsteemi tuumaga integreeritud moodul *tsearch2* – seda ei pea eraldi installeerima
- Funktsioonid tekstivektorite koostamiseks ja töötlemiseks
  - SELECT to_tsvector('simple', 'New systems generate new problems');
  - *Tulemus*: 'new':1,4 'systems':2 'generate':3 'problems':5

Tekstimassiivide indekseerimine (PostgreSQL) (2)

- Tabelis salvestatud tekstivektoritele võib luua indeksi
  - CREATE INDEX idx_tekst ON Murphy_seadus USING gist(vektor);
    - Tabeli *Murphy_seadus* veerg *vektor* on tüüpi *tsearch*
    - gist – Generalized Search Tree
- Indekseeritud tekstivektorite põhjal võib teha päringuid
  - SELECT * FROM Murphy_seadus WHERE vektor @@ 'systems'::tsquery;

Indeksi loomise aktiivselt kasutatavas baasis (PostgreSQL)

- Vaikimisi ei saa tabeli indekseerimise ajal tabelis andmeid muuta.
- PostgreSQL võimaldab indekseeritavas tabelis samaaegset andmete muutmist käsuga:
  - CREATE INDEX CONCURRENTLY idx_tellitud_kogus ON Tellimuse_rida (kogus);
  - Probleem – indeksi loomine võtab rohkem aega. Suurenenud süsteemi töökoormus aeglustab andmemuudatusi.

Indeksi uuesti loomine (PostgreSQL, Oracle)

- PostgreSQL
  - Indeksi, tabeli, andmebaasi kaupa
  - REINDEX INDEX idx_tellitud_kogus;
  - REINDEX TABLE Tellimuse_rida;
  - REINDEX DATABASE minubaas;
  - REINDEX SYSTEM minubaas;
    - Loob uuesti indeksid süsteemikataloogi tabelitele
- Oracle
  - ALTER INDEX idx_tellimuse_rida REBUILD NOREVERSE;

Indeksi uuesti loomine (PostgreSQL) (2)

- Strateegia 1:
  - DROP INDEX idx_tellitud_kogus;
    - Blokeerib andmete lugemise ja muutmise tabelis *Tellimuse_rida*.
  - CREATE INDEX CONCURRENTLY idx_tellitud_kogus ON Tellimuse_rida (kogus);
- Strateegia 2:
  - REINDEX INDEX idx_tellitud_kogus;
  - Ei ole võimalik muuta andmeid tabelis *Tellimuse_rida*.
  - Samuti blokeeritakse transaktsioonid, mis üritavad indeksit lugeda

Operatsioone indeksiga (Oracle)

- Indeksi nähtamatuks/nähtavaks muutmine
  - ALTER INDEX my_index_i INVISIBLE;
  - ALTER INDEX my_index_i VISIBLE;
  - Andmebaasisüsteem sellist indeksit vaikimisi ei kasuta.
  - Võimaldab testida indeksi eemaldamise mõju.
- Indeksi kasutamise jälgimise alustamine/lõpetamine
  - ALTER INDEX my_index_i MONITORING USAGE;
  - ALTER INDEX my_index_i NOMONITORING USAGE;
  - Jälgimise andmeid näeb läbi vaate *v$object_usage*.
  - Võimaldab otsida indekseid, mida andmebaasisüsteem ei kasuta.
  - Sellised indeksid tuleks andmebaasist eemaldada, sest nendest tulenev kulu ületab nendest saadava võimaliku tulu.

Andmebaasiserveris talletatud rutiinid

Oracle	PostgreSQL
◆Saab kasutada anonüümseid plokke	◆Saab kasutada anonüümseid plokke (alates PostgreSQL 9.0)
◆Saab luua protseduure ja funktsioone	◆Saab luua funktsioone
◆Protseduurid ja funktsioonid saab koondada pakettideks	◆Põhiliseks kasutatavaks keeleks on on: PL/pgSQL
◆Põhiliselt kasutatavaks keeleks on: PL/SQL	◆Saab kasutada ka SQL, PL/Tcl, PL/Perl, PL/Python ja teisi keeli
◆Saab kasutada ka Java keelt	◆Arendajal on teiste keelte toetust võimalik juurde lisada

Edasi

Süsteemi-defineeritud funktsioonid (PostgreSQL)

- Leia, kui mitu päeva on möödunud käesoleva aasta algusest.
- SELECT date_part('days', (CURRENT_DATE - date_trunc('years', CURRENT_DATE))) AS paevi_aasta_algusest;

Deterministlik funktsioon – samade argumentidega alati sama tulemus.

Kasutaja-defineeritud funktsioon (PostgreSQL)

- SQL keeles kirjutatud **deterministlik funktsioon**, mis teisendab Fahrenheiti temperatuuriskaalal temperatuuri Celsiuse skaalale
- CREATE FUNCTION **Fahrenheit_Celsius**(numeric) RETURNS numeric AS $$SELECT (($1 - 32.0) * 5.0 / 9.0);$$ LANGUAGE sql **IMMUTABLE** STRICT LEAKPROOF;
- SELECT **Fahrenheit_Celsius**(temp) AS temperatuur FROM Mootmine WHERE **Fahrenheit_Celsius**(temp)>22;

Kasutaja-defineeritud funktsioon (PostgreSQL) (2)

- CREATE OR REPLACE FUNCTION lopeta_oppimine () RETURNS **void**
  AS $$**UPDATE** Oppimine SET lopetamise_kuupaev=CURRENT_DATE
  WHERE lopetamise_kuupaev IS NULL;$$
  LANGUAGE sql SECURITY DEFINER
  SET search_path = public, pg_temp;

void – funktsioon ei tagasta väärtust

Tabelifunktsioon

- Nii PostgreSQLis kui Oracles saab luua *tabelifunktsioone* – funktsiooni väärtus koosneb nullist või rohkemast reast.
- Funktsiooni väärtuses võib olla korduvaid ridu.
- Selliseid funktsioone võib kasutada leidmaks andmeid, mida kasutajatele aruannetes näidata.

Tabelifunktsioon (PostgreSQL)

CREATE OR REPLACE FUNCTION f_Get_Tudeng_by_id (p_matrikli_nr Tudeng.matrikli_nr%TYPE) RETURNS SETOF Tudeng AS $$ SELECT * FROM Tudeng WHERE matrikli_nr= p_matrikli_nr; $$ LANGUAGE sql SECURITY DEFINER STABLE ROWS 1 SET search_path = public, pg_temp;

- SELECT * FROM f_Get_Tudeng_by_id(p_matrikli_nr:='990999');
- Tagastab sellise tudengi andmed, kelle matrikli number on '990999'.

Aktiivne andmebaas

- *Aktiivne andmebaas* reageerib kasutajate poolt tehtud tegevustele.
- SQL-andmebaasides on selle saavutamiseks:
  - tabelitega seotud deklaratiivsed kitsendused,
  - vaikimisi väärtused,
  - trigerid.
- PostgreSQLis on lisaks võimalik luua ka reegleid.

Aktiivne andmebaas (2)

- Selleks, et kirjutada **tarkvara** on vaja nõudeid.
  - Nõuded, mille järgi *realiseerida aktiivne andmebaas*, tulevad *kontseptuaalsest andmemudelist*.
- Selleks, et esitada **nõudeid** on vaja tunda **valdkonda**, mille jaoks **tarkvara** luuakse.
  - Nõuded aktiivse andmebaasi kohta tulevad kontseptuaalsesse andmemudelisse valdkonnas kehtivatest *ärireeglitest*.

Triger (millised sündmused võivad trigeri protseduuri käivitada?)

Oracle	PostgreSQL
◆Tabelites andmete muutmine (INSERT, UPDATE, DELETE ja TRUNCATE sündmused) ◆Veel sündmuseid, mille toimumine võib käivitada trigeri: ALTER, ANALYZE, ASSOCIATE STATISTICS, AUDIT, COMMENT, CREATE, DISASSOCIATE STATISTICS, DROP, GRANT, NOAUDIT, RENAME, REVOKE, DDL, AFTER STARTUP, BEFORE SHUTDOWN, AFTER DB_ROLE_CHANGE, AFTER SERVERERROR, AFTER LOGON, BEFORE LOGOFF, AFTER SUSPEND, AFTER CLONE, BEFORE UNPLUG	◆Tabelites andmete muutmine (INSERT, UPDATE, DELETE ja TRUNCATE sündmused) ◆TRUNCATE – lause tabelis olevate andmete kiireks kustutamiseks ◆Sündmustrigerid (alates PostgreSQL 9.3) – CREATE, ALTER, DROP, SECURITY LABEL, COMMENT, GRANT, REVOKE

Funktsioon ja triger (PostgreSQL)

```
CREATE TABLE Tabel_log(
tabel_log_id SERIAL CONSTRAINT pk_Tabel_log
   PRIMARY KEY,
rida_id VARCHAR(255) NOT NULL,
muutja VARCHAR(128) NOT NULL,
muutmise_aeg TIMESTAMP NOT NULL
   DEFAULT LOCALTIMESTAMP(0),
tegevus VARCHAR(128) NOT NULL,
tabel VARCHAR(128) NOT NULL);
```


Funktsioon ja triger (PostgreSQL) (2)

```
CREATE OR REPLACE FUNCTION
   f_tudeng_log_update_delete() RETURNS TRIGGER AS $$
DECLARE
 v_rida_id Tudeng.matrikli_nr%TYPE;
BEGIN
v_rida_id:=upper(OLD.matrikli_nr);
INSERT INTO Tabel_log(rida_id, muutja, tegevus, tabel)
VALUES (v_rida_id, SESSION_USER, TG_OP,
   TG_TABLE_NAME);
RETURN OLD;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER
SET search_path = public, pg_temp;
```

Oracle võimaldab soovi korral trigeri protseduuri kirjutada CREATE TRIGGER lausesse

Funktsioon ja triger (PostgreSQL) (3)

- Spetsiaalmuutujad trigeri funktsioonis.
  - **OLD** – *reatüüpi* muutuja, mille väärtuseks on rea muutmiseelne versioon (UPDATE operatsiooni korral) või kustutatav rida (DELETE operatsiooni korral).
  - **TG_OP** – *text* tüüpi muutuja, mille väärtus on trigeri funktsiooni käivitanud operatsiooni tüübi nimi (INSERT, UPDATE, DELETE või TRUNCATE).
  - **TG_TABLE_NAME** – *name* tüüpi muutuja, mille väärtus on tabeli nimi, millega seotud sündmus põhjustas trigeri funktsiooni käivitumise.

Funktsioon ja triger (PostgreSQL) (4)

- CREATE TRIGGER t_tudeng_log_update_delete AFTER UPDATE OR DELETE ON Tudeng FOR EACH ROW EXECUTE PROCEDURE f_tudeng_log_update_delete();
- Reataseme triger, mis on seotud tabeliga *Tudeng* ja mille protseduur käivitub peale rea kustutamist või muutmist.
- Logib tabelis *Tudeng* toimunud muutmise ja kustutamise sündmused.

Alternatiivne lahendus alates PostgreSQL 10

```
CREATE OR REPLACE FUNCTION f_tudeng_log_update_delete() RETURNS
TRIGGER AS $$
BEGIN
IF TG_OP IN ('UPDATE','DELETE') THEN
        INSERT INTO Tabel_log(rida_id, muutja, tegevus, tabel)
        SELECT matrikli_nr, SESSION_USER, TG_OP, TG_TABLE_NAME
        FROM old_table;
END IF;
RETURN NULL;
END; $$ LANGUAGE plpgsql SECURITY DEFINER
SET search_path = public, pg_temp;
```

old_table – *ülemineku tabel*, kus on trigeri käivitanud lause poolt kustutatud read / muudetud ridade muutmiseelne versioon

Alternatiivne lahendus alates PostgreSQL 10 (2)

```
CREATE TRIGGER
t_tudeng_log_update
AFTER UPDATE ON Tudeng
REFERENCING OLD TABLE
AS old_table
FOR EACH STATEMENT
EXECUTE PROCEDURE
f_tudeng_log_update_delete();
```

```
CREATE TRIGGER
t_tudeng_log_delete
AFTER DELETE ON Tudeng
REFERENCING OLD TABLE
AS old_table
FOR EACH STATEMENT
EXECUTE PROCEDURE
f_tudeng_log_update_delete();
```

Lausetaseme trigerid, mis käivituvad iga tabelis *Tudeng* andmeid muutva/kustutava lause kohta üks kord – peale lause täitmist.

Milleks trigerit kasutada?

- **Väärtuse leidmine**, mis uue rea lisamisel tabelisse, tuleb paigutada mõnda selle rea välja.
  - Surrogaatvõtme väärtuse leidmine (tabeli ja arvujada generaatori sidumine).
  - Tuletatud väärtuste leidmine.
  - Vaikimisi väärtuse leidmine, kui see sõltub andmebaasis juba olemasolevatest andmetest.

Milleks trigerit kasutada? (2)

- Andmemuudatuse **reeglitele vastavuse** kontroll.
  - Trigeri protseduuri täitmise katkestamine võimaldab tühistada protseduuri käivitumise põhjustanud andmemuudatuse.
  - Võimalusel eelistage deklaratiivset kitsendust – häda on vaid selles, et tänapäeva SQL-andmebaasisüsteemide maailmas pole see sageli võimalik.

Milleks trigerit kasutada? (3)

- Andmemuudatustega seotud **kompenseerivate tegevuste** tegemine tagamaks, et andmete liiasus on *kontrollitud*
- Andmebaasis tehtud tegevuste **logimine**
- Keerukamad **turvakontrolli** reeglid
- **Vaadete** „programmeerimine“, et keerukamate alampäringutega vaadete kaudu saaks andmebaasis andmeid muuta (*instead of* trigerid)

Triger (2)

Oracle	PostgreSQL
Instead of triger	*Instead of* triger ja *do instead* reegel
Ei	Kitsenduste trigerid, mille protseduuri käivitamise saab lükata transaktsiooni lõppu. Võimaldavad jõustada keerukamaid kitsendusi.
Skeemitaseme ja andmebaasi taseme trigerid	Alates PostgreSQL 9.3 sündmustrigerid – võimalik reageerida nt skeemimuudatustele (CREATE, ALTER, DROP laused)

Operaator

Oracle	PostgreSQL
Kasutajad saavad luua uusi operaatoreid	Kasutajad saavad luua uusi operaatoreid

Kasutaja-defieeritud operaator (PostgreSQL)

- CREATE FUNCTION textcat_coalesce(text, text) RETURNS text AS $$ SELECT coalesce($1,'') || coalesce($2,''); $$ LANGUAGE sql IMMUTABLE LEAKPROOF;
- COMMENT ON FUNCTION textcat_coalesce(text, text) IS 'Stringide konkatenatsiooni operaatorit & realiseeriv funktsioon. Tagab, et operatsiooni tulemus pole NULL.';
- CREATE OPERATOR & (LEFTARG = text, RIGHTARG = text, PROCEDURE = textcat_coalesce);
- SELECT NULL || 'katsetus' AS tulemus;
  - Tulemus: NULL
- SELECT NULL & 'katsetus' AS tulemus;
  - Tulemus: katsetus

Laiendus

Oracle	PostgreSQL
Ei Oracle pakub hulka süsteemi-defineeritud pakette, millest igaühes on mingi konkreetse valdkonna kohta käivad protseduurid ja funktsioonid. Enamasti installeeritakse need paketid andmebaasis automaatselt.	Jah Laienduse (*extension*) installeerimise tulemusena lisatakse andmebaasi hulk funktsioone/tüüpe/operaatoreid, mis on mingi konkreetse valdkonna kohta. Arendajatel võimalik luua uusi laiendusi.

Reegel

Oracle	PostgreSQL
Ei	Jah Reegel (*rule*) (täpsemalt *andmekäitluskeele lausete ümberkirjutamise reegel*) on andmebaasiobjekt, mis määrab tegevused, mis täidetakse andmebaasis juhul, kui toimub sündmus ning on täidetud tingimused. Võimaldab näiteks realiseerida vaadete kaudu andmebaasis andmete muutmist.

Reegel *vs*. triger PostgreSQLis

- **Reegel**
- Rakendatakse lause *l* töötlemise käigus (muudab lauset või lisab täiendavaid lauseid).
- Saab luua SELECT, INSERT, UPDATE, DELETE reegleid.
- Saab luua DO INSTEAD reegleid.

- **Triger**
- Käivitub lause *l* täitmise tulemusena.
- Saab luua INSERT, UPDATE, DELETE, TRUNCATE trigereid ning sündmustrigereid.
- Saab luua INSTEAD OF trigereid.

Reegel (PostgreSQL)

- CREATE RULE keela_uute_tudengite_lisamine AS ON INSERT TO Tudeng **DO INSTEAD NOTHING**;
  - INSERT lauset ei hakatagi täitma. Andmete lisaja ei saa veateadet.
  - Ei rakendu COPY...FROM lause kasutamise korral!
    - Muudab problemaatiliseks *kitsenduste jõustamise* reeglite abil, sest sellisest kitsendusest saab mööda hiilida.
- DO INSTEAD reegleid kasutatakse, et võimaldada vaadete kaudu andmebaasis andmeid muuta.

Märkus Oracle kohta

- Oracles ei saa luua andmekäitluskeele lausete ümberkirjutamise reegleid.
- Küll aga saab Oracles keelata tabelis andmete muutmise.
  - ALTER TABLE Tudeng READ ONLY;
  - DELETE FROM Tudeng;
    - ERROR at line 1: ORA-12081: update operation not allowed on table "C##TUD1"."TUDENG"
  - ALTER TABLE Tudeng READ WRITE;

Kommentaarid

- Nii Oracle kui ka PostgreSQL võivad kommenteerida andmebaasiobjekte (nagu näiteks tabelid või vaated), et selgitada nende semantikat.
- Kommentaarid salvestatakse süsteemikataloogis.
- Igal objektil saab olla maksimaalselt üks kommentaar. Uus kommentaar kirjutab vana üle.
- Näide nii PostgreSQLis kui Oracles.
  - COMMENT ON TABLE Koristaja IS 'Koristaja on isik, kelle tööülesannete hulka kuulub ruumide igapäevane korrashoid.';

Andmebaasi skeemi evolutsioonist – näide Oracles

- CREATE TABLE t_test (test VARCHAR2(10));
- CREATE VIEW v_test AS SELECT test FROM t_test;
- ALTER TABLE t_test MODIFY (test VARCHAR2(20));
  - Tabeli veeru väljapikkuse muutmine õnnestub.

Andmebaasi skeemi evolutsioonist – näide PostgreSQLis

- CREATE TABLE t_test (test VARCHAR(10));
- CREATE VIEW v_test AS SELECT test FROM t_test;
- ALTER TABLE t_test ALTER COLUMN test TYPE VARCHAR(20);
  - ERROR: cannot alter type of a column used by a view or rule
  - DETAIL: rule _RETURN on view v_test depends on column "test"

PostgreSQL (2) – kuidas leida veerust sõltuvad vaated?

- SELECT view_schema, view_name FROM Information_schema.view_column_usage WHERE table_schema='public' AND table_name='t_test' AND column_name='test';
- Päring süsteemikataloogi vaate põhjal.

PostgreSQL (3) – vajalikud muudatused

- Pean kustutama vaate, muutma veeru omadusi, looma vaate uuesti
- Kui mõne funktsiooni parameetri tüüp on deklareeritud **t_test.test%TYPE**, siis on vaja ka see funktsioon kustutada ja uuesti luua
- Kui veerg on defineeritud domeeni põhjal, siis veel keerulisem, sest ALTER DOMAIN ei luba muuta tüüpi/väljapikkust

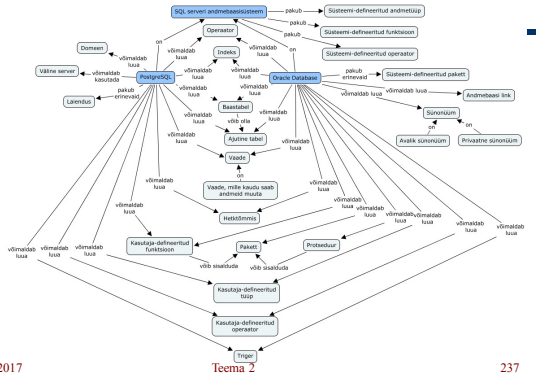
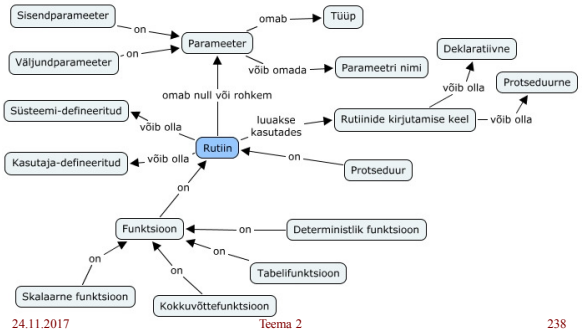
PostgreSQL (4) – järeldused

- PostgreSQLis tüübi/väljapikkuse valimisel tehke valik varuga
- Kui pole maksimaalses suuruses väga kindlad, siis eelistada tüüpe nagu INTEGER, BIGINT, TEXT ning lisada piirangud väljapikkusele CHECK kitsenduste abil

Järeldused (5)

- Väljapikkuse varuga valik on näide *kaitsvast andmebaasi disainist.*
  - *Kaitsva programmeerimise* alamliik: https://en.wikipedia.org/wiki/Defensive_programming
- Kaitsev andmebaasi disain tähendab, et andmebaas kavandatakse nii, et seda oleks vaja võimalikult vähe muuta või oleks muutmine lihtne.

Mõningad teema põhimõisted



Mõningaid teema põhimõisteid (2)