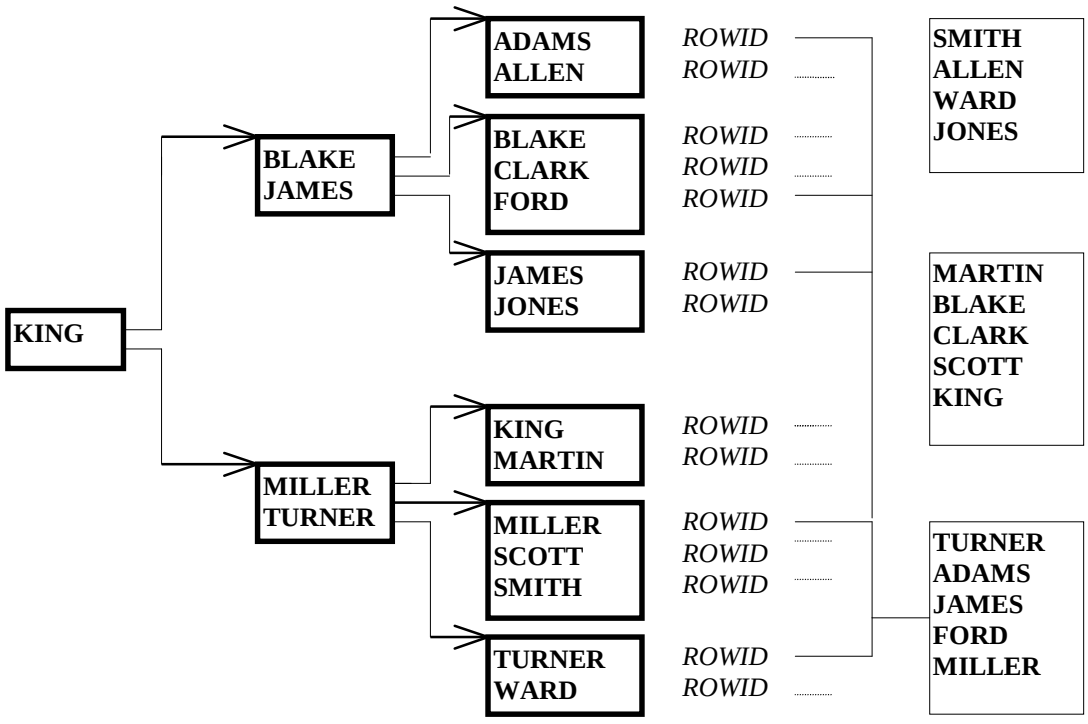
Teema9. Teised andmebaasi objektid: indeksid, jadad (sequences), kasutajad,kasutajate õigused,andmekaitse.

- **Indeksid** - ei ole standardi osa, kuid praktiliselt igas andmebaasis on olemas. Indeksite eesmärgiks on : 1. Kiirendada andmete otsimist ja sorteerimist (juhul kui otsimine toimub mingi võtme e. kriteeriumi järgi.)
  2. Kindlustada vajalike andmeväljade unikaalsus (kõik primaarvõtmed ja UNIQUE-piiranguga väljad indekseeritakse automaatselt andmebaasisüsteemis)

INDEKS -on andmebaasi objekt (nagu tabel või vaade). Indeks on andmebaasi **objekt**, mis sisaldab kõikide indekseeritava(te) veeru (veergude) väärtust ning **viidet**(e.aadressi) nende kirjete **asukohale** (ROWID) andmebaasis, mille koosseisu antud veergudes asuvad **väljad** kuuluvad. Indeksid mingite veergude jaoks luuakse kas automaatselt andmebaasisüsteemi poolt või kasutaja poolt CREATE INDEX lausega. Tavaliselt on indeksi **struktuurina** kasutusel B-tree struktuur.

B-TREE INDEKSEERIMINE

<……INDEKS- blokid ……..> TABELI-blokid



Andmebaasis on andmed salvestatud enamasti nn. tabeliblokkidesse, need blokid kirjutatakse täis andmete laekumiser järjekorras, mingit sisulist järjestust tabeliblokkidesse kirjutattud andmed ei oma.
Indeksid on vajalikud selleks, et andmete leidmiseks kuluks ühesugune aeg nii sel juhul, kui andmed on esimestes tabeliblokkides, kui ka siis, kui andmed asuvad viimastes tabeliblokkides.

Iga indeks koosneb reast nn. indeksblokkidest, mis on organiseeritud tasakaalustatud puu (balanced tree) põhimõttel.

Iga indeksiblokk sisaldab endas rea võtmeväärtusi (indekseeritava veeru väärtusi) ja rea viiteid endast madalal asuvatele indeksiblokkidele. Kõige madalama taseme indeksblokid sisaldavad veeru väärtusi ja viita andmete asukohale andmebaasis (tabeli blokkides)

Kõige kõrgema taseme indeksblokis asub sellise veeru väärtus, millest suremaid ja väiksemaid veeru väärtusi on mõlemaid 50% (st antud veeru väärtus asub kõigi väärtuste skaala keskel.) Kõige esimene blokk jagab veeru väärtuste hulga kaheks alamhulgaks.Esimene blokk sisaldab viiteid kahele jargmisele blokile, mis omakorda sisaldavad järgmisi viiteid, jgades selliste viidetega tekkivaid veeru väärtuste alamhulki järjest väiksemaks, kuni viimasel tasemel jõutakse kirje aadressideni.

Veeru väärtus , mis salvestatakse indeksiblokki, kujutab endast põhimõtteliselt arvväärtust, mis arvutatakse lähtuvalt vastava veeru väärtusest - n. kui tegemist on tekstiväljaga, siis saavad alfabeedi eesotsas olevad veeru väljad väiksemad väärtused ja et kõik saadud väärtused oleksid unikaalsed.

Näiteks indeksite väärtused perekonnanimede korral võiksid olla
ADAMS - 100567
TURNER - 170001

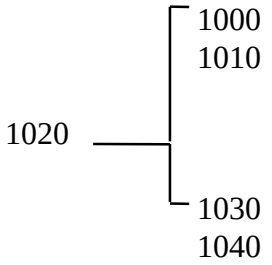
Veergude väärtused indeksblokkidesse arvutatakse nii, et et seda veeru väärtust oleks võimalik paigutada indeksblokkidesse selliselt, et säiliks indeksblokkide puu (e. B-TREE) tasakaal - st. tuleb säilitada, et kui indeksblook sisaldab endas viiteid alam-indeksiblokkidele, siis nendes alamblokkides oleks võrdne arv kirjeid (veeru väärtusi.). See tähendab, et kirjete lisamisel ja eemaldamisel tuleb indeksid andmebaasisüsteemi poolt uuesti arvutada ja indeksipuu struktuuri vastvalt saadud indeksi väärtustele muuta, et vältida indeksipuu (B-TREE) tasakaalust välja viimist. Kui indeksipuu struktuuri n. kirjete lisamisel mitte muuta, läheks indeksipuu tasakaalust välja näiteks siis, kui alafabeedi lõputähtedega nimesid lisataks rohkem kui algustähtedega algavaid nimesid.

Indeksi arvutamise järjekord kirje lisamisel või eemaldamisel -
1. Vastavalt veeru väljadele leitakse veeru väärtused

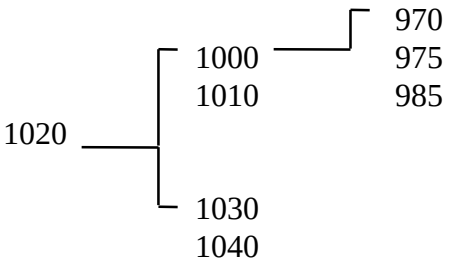
BLAKE - 1000
JAMES - 1010
KING - 1020
MILLER - 1030
TURNER - 1040

2. Organiseeritakse indeksipuu arvestusega, et indeksipuu esimesest blokist otsima hakates oleks ja sealt üles- või allapoole liikudes oleks leidmise tõnäosus 50%

indeksblokid koos
veeru väärtustega



tasakaalustamata puu (kui lisatud palju A-tähega isikuid



Otsimine indeksi järgi - n. kui on vaja leida kirjet, milles perekonnanimi on TURNER ja vajalik (perekonnanime veerg) on indekseeritud vastavlt ülalnimetatud näitele, siis

a. sisenetakse otsinguga esimest blokist ja kontrollitakse, kas otsitava veeru väärtus on suurem või väiksem esimesest indeksobloki väärtusest (korralikult organiseeritud indeksipuu korral peavad olema need tõenäosused võrdsed)

b. kui indeksi väärtus osutus suuremaks (TURNER = 1040), liigutakse kahendvaliku põhimõttel alumisse blokki ja kontrollitakse, ka seal indeksit, vastavalt saadud tulemusele liigutakse järgmistesse blokkidesse jne., kuni viimase tasemeni, pidevalt võrreles otsitava veeru väärtust indeksiblokkides olevate veergude väärtustega.

- Indeksite loomine tabeli veerule või veergudele:

CREATE [UNIQUE] INDEX index_name
ON table (column [, <column2>]…);

näiteks:

CREATE INDEX index_pnimi
ON tootaja (pnimi);

CREATE UNIQUE INDEX arve_nr
ON arve (arve_nr,kliendi_nr);

- **Indeksite tüübid -**

UNIQUE - kindlustab, et selles veerus oleksid unikaalsed väärtused, samuti kiirendab otsimist kirjetest indekseeritud veeru (veergude) järgi.

NON UNIQUE - kindlustab kiiresti otsimise ja sorteerimise indekseeritud veeru (veergude)

SINGLE COLUMN - ühele veerule loodud indeks

CONCATENATED - kuni 16 veergu võib olla ühe indeksiga indekseeritud kas siis otsimise kiiruse või kirjete unikaalsuse kindlustamise eesmärgil

- **Indeksite kasutamine** - enamus andmebaasisüsteeme otsustab ise, millal ta kasutab otsimiseks indeksit, millal mitte. Indeksite kasutamise reeglid on järgmised:

- mingi veeru indeksit kasutatakse, kui päringus on WHERE-klausel ja selles klauslis on märgitud see indekseeritud veerg.
- indeksit ei kasutata, kui WHERE-klauslis toodud veerg on osa mingist funktsioonist või tehtest (expression). Näiteks:
- indeks võib kiirendada SELECT -lauses veel DISTINCT klausli täitmist ning ORDER BY sorteerimist ning GROUP BY grupeerimist

SELECT * FROM tootaja
WHERE UPPER(pnimi) = ‘ KASK’;

- tabeli veergudele loodud indekseid ei kasutata, kui otsitakse vaatest, mis sellel tabelil põhineb. Vaate veergudele ei saa indekseid luua.Seega ei ole vaadete (VIEW) puhul indeksitest eriti kasu.

Soovitused indeksite kasutamiseks:

veergu indekseerida on eriti mõtet siis, kui on ette näha, et sellest tabelist hakatakse tegema “kitsaid” WHERE-klausliga päringuid, kus kriteeriumiks on antud veerg . (“kitsas” päring - päring, mille tulemuseks on vähem kui 10% tabeli kirjete koguarvust.)

indekseerida tuleks kõik veerud, mille unikaalsus on nõutud

indekseerida need veerud, mida kasutatakse tihti WHERE- klauslis

sellisestele veergudele, mida kasutatakse koos ühes ja samas WHERE- klauslis, on mõtet luua ühine, nn. CONCATENATED indeks.

Kõigest ülaltoodudst hoolimata tuleks hoiduda rohkem kui kolmest indeksist tabeli peale sest:
- indekseid ja B-TREE-d tuleb uuendada, kui lisatakse uus kirje, kustutatakse kirje või muudetekse indekseeritud veeru väärtust.
Pragmaatiline käitumine - kui kohe pärast tabeli loomist on vaja sisestada suur hulk kirjeid ja ei ole vaja sellest tabelist midagi otsida, siis on kaval alguses luua tabel, siis sisestada kirjed ja lôpuks luua indeks.
Peale selle vôtavad indeksid kettaruumi - mida pikem on indekseeritav väli ja mida erinevamad on selle välja väärtused, seda rohkem ruumi vôtab indeks. Seega tuleks näiteks primaarvôtit moodustava veeru (veergude) puhul hoolikalt kaaluda väljapikkusi.

Mida suurem on tabel, seda rohkem on indeksist kasu, kuid seda suuremad on ka indeksi ülalpidamise kulud - ajakulu uuendamisel ja indeksi all olev kettaruum.
Suhteliselt kôige rohkem kasu on indeksitest selliste tabelite puhul, mis on suured, mida uuendatakse harva ja millest otsitakse sageli .

Indeksit ei saa kahjuks luua vaatele, näiteks kui on vaja välja trükkida tudengite eksamitulemused perekonnanimede järjestuses, siis ei ole kasu tudengi pnimi-indeksist, sest indekseeritud on tudengi tabel, sorteerida tahetakse aga hoopis oppim-tabelit
Nii ei saa:

```
CREATE VIEW hinded AS
   SELECT pnimi, enimi, hinne FROM tudeng,oppim
   WHERE tudeng.tkood=oppim.tudkood;
CREATE INDEX pn_ind ON hinded (pnimi);
SELECT * FROM hinded ORDER BY pnimi;
```

Olenevalt andmebaasisüsteemist käsitletakse NULL-e erinevalt. Tavaliselt NULL-e ei indekseerita, ja seega päring
SELECT * FROM tudeng WHERE pnimi IS NULL
ei kasuta indeksit.

- **Indeksi kustutamiseks**

```
DROP INDEX index;
```

Primaarvôtme indeksi kustutamisel ...
Unikaalse välja indeksi kustutamisel ...

• jadad - sequences

Andmebaaside tabelitesse kirjete lisamisel osutuvad tihti vajalikuks unikaalsed järjenumbrid, mida saaks lisada uutesse kirjetesse, näiteks arve numbriteks, kliendi koodideks , primaarvõtmeteks jne. Andebaasisüsteemides kasutatakse nn. generaatoreid selleks, et genereerida unikaalseid, järjest suurenevaid numbreid tabeli kirjete jaoks.
Mingi tabeli jaoks numbrite genereerimiseks tuleb:

1. luua numbrite generaator (mingi CREATE-lause, erinevates süsteemides erinev)
2. iga kord, kui tekib vajadus uue järjenumbri järele, tuleb, kasutades loodud generaatorit , genereerida uus järjenumber.

Generaator Oracle näitel

```
CREATE SEQUENCE  arve_nr
INCREMENT BY 10
START  WITH 10
MAXVALUE 10000;
```

kasutamine:
pseudo-veerule NEXTVAL viidates on võimalik generaatorist saada kätte uut järjenumbrit.

```
SELECT arve_nr.NEXTVAL
FROM SYS.DUAL;
```

```
NEXTVAL
-------------------
     10
```

järgmine viide NEXTVAL-ile annab juba järgmise numbri

```
NEXTVAL
-------------------
     20
```

Generaator Borlandi Local Interbase Serveri näitel

```
CREATE GENERATOR EMPNO_GEN;
EMPNO = GEN_ID(EMPNO_GEN, 1);
```

Generaator ei ole seotud andmebaasi tabeliga, milles tema poolt tekitatud numbreid kasutatakse. Generaatorist saadud järjenumbreid ei ole võimalik tagasi võtta Kui on saadud (vastavalt eelnevale näitele) generaatorist number 10, pandud see kirjesse arve numbriks, see kirje on ära kustutatud, siis uue kirja lisamisel saadakse arve numbriks sellest generaatorist 20, 10 -t ei ole võimalik saada.

Generaatoreid võib kasutada SELECT-lauses, INSERT-lauses VALUES-listis, UPDATE -lauses SET -klauslis.

- **Kasutajad, õiguste jagamine, privileegid**

Enamus suuremaid andmebaasisüsteeme omavad detsentraliseeritud andmekaitsesüsteemi. Kasutajate õiguste jagamist on käsitletud Oracle näitel. Detsentraliseeritud õiguste jagamine andmebaasisüsteemis tähendab, et kasutajad (andmebaasis defineeritud kasutajad) on ise vastutavad nende poolt loodud objektide kohta õiguste andmises teistele kasutajatele. Kasutaja loob andmebaasis tabeleid, indekseid, protseduure ja muid objekte. Esialgu saab neid objekte kasutada ainult neid loonud kasutaja (ja süsteemiadministraatori õigustega isik).Kasutaja võib anda tesitele kasutajatele õiguse nende objektidega tegutsemiseks.

- **Kasutajate loomine**

Andmebaasi kasutaja loomiseks (kellel on õigus oma kasutajanime ja parooliga andmebaasi sisse logida) :

CREATE USER isik **IDENTIFIED BY** salasona;

Oracles on andmebaasi kasutaja loomisel on võimalik määrata veel:
- määrata seda, kas andmebaasi kasutaja nime küsimiseks kasutatakse andmebaasisüsteemi või operatsioonisüsteemi vahendeid (nn. EXTERNALLY option, selliste süsteemide, kus operatsioonisüsteem alati küsib kasutajanime sisselogimisel - mitmesugused UNIX-id, Windows NT jne.)
- määrata kasutajale andmebaasiruum, millese paigutatakse selle kasutaja poolt loodud objektid (tabelid, protseduurid jne.)

- **Kasutajate õigused., liigid.**

Kasutajate õigused jagunevad **süsteemiprivileegideks** ja **objektiprivileegideks.**

Objektiprivileegid käsitlevad andmebaasi objektide (tabelid, protseduurid, indeksid) kasutamist.

Objekti privileegide liigid:

SELECT - lubab lugeda tabelist või vaatest
INSERT - lisada tabelisse või vaatesse
UPDATE - muuta olemasolevaid andmeid
DELETE - kustutada
ALTER - muuta olemasolevate tabelite struktuuri ja veergude definitsioone
INDEX - indekseerida tabeleid
REFERENCES - luua viiteid sellele tabelile
ALL
- kõik ülaltoodud õigused

Kasutajale privileegide andmise süntaks:

GRANT privileges
ON object
TO user;

Objektiks, mille kohta õigusi jagatakse, võib olla tabel , vaade, jada (sequence), sünonüüm.
Näiteks: lubada kasutajal nimega isik lugeda tabelit arve.

GRANT SELECT
ON arve
TO isik;

ALTER, INDEX ja REFERENCES on pole võimalikud VIEW-de korral.

- **Süsteemiprivileegid**

käsitlevad üldisenmaid kasutaja õigusi andmabaasis, peamiselt seda, milliseid üldtegevusi kasutaja võib teha.

Mõned põhilisemad privileegid Oracle-s:

CREATE SESSION - lubab kasutajal ühenduda 8connect) andmebaasi
CREATE TABLE - lubab luua tabeleid
CREATE VIEW - lubab luua vaateid
CREATE USER - lubab luua uusi kasutajaid

- **Rollid**

Rollid on (mingi nimega identifitseeritud) teatud süsteemiprivileege. Kui mitmele kasutajale tahetakse anda samasugune komplekt süsteemiprivileege, siis on mõttekas koguda need privileegid kokku, moodustada nendest roll ja kasutajatele anda see roll. Kasutaja saab rolli saamisel kõik selles rollis määratud süsteemiprivileegid. Kasutajal võib olla mitu rolli ja ühel rollil võib olla mitu kasutajat. Slline süsteem loob kasutajatele õiguste jagamiseks mugavamad ja paindlikumad võimalused.

Rolli loomine:

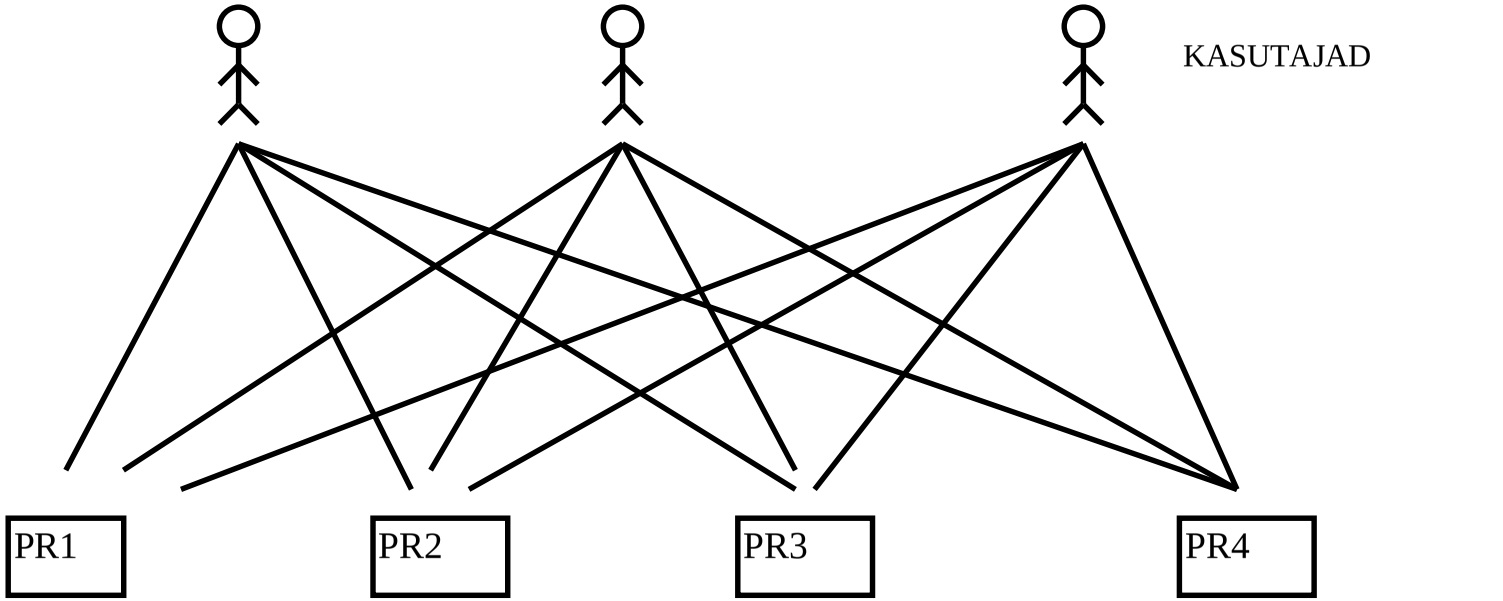
CREATE ROLE teller **IDENTIFIED BY** raha;

Kasutajad, kellele on omistatud roll **teller**, peavad siis andmebaasiga töötamisel kasutama ka rolliga kaasaskäivat password-I (kui see on rolli loomise lauses määratud, nagu praegu **raha**)

Rollile õiguste andmine toimub samuti kui kasutajale õiguste andmine, GRANT lausega.

N: **GRANT CREATE SESSION TO** teller;

Privileegide andmine ilma rollideta



Pivileegide andmine rollide kasutamisega