Loeng 12. Suurte andmebaasisüsteemide arhitektuur : Oracle näide

Oracle 7 andmebaasisüsteemi arhitektuur, kirkjeldatud on:
* Süsteemi füüsiline failistruktuur
* Süsteemi mälueralduse struktuurid
* Süsteemi protsessid

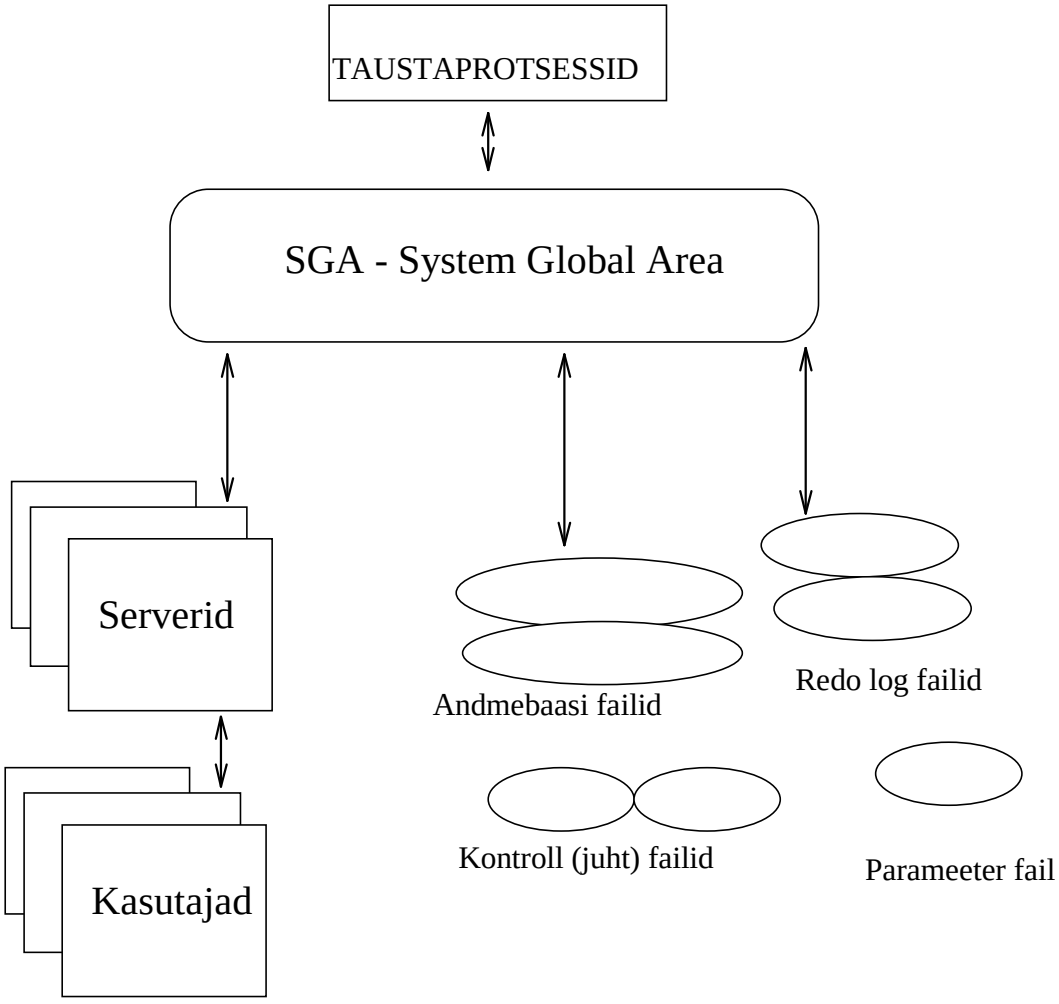
Oracle 7 andmebaasisüsteemi arhitektuur sisaldab:

* **SGA puhvrite ressurssid** (SGA - *System Global Area* - süsteemi tööks eraldatud mälu) andmebaasi blokkide ja *redo log* informatsiooni salvestamiseks (Redo - log-i kirjutatakse kõik andmebaasi tehtud muudatused (nii andmete kui andmebaasi struktuuri muutmised), ja sealt on võimalik süsteemi riknemise korral taastada endist seisu, alguses kirjutatakse muutused mällu (*redo log entries*) ja hiljem püsivalt vastavatesse failidesse, nn. *redo*-failidesse)

* SGA jagatud ressurss (*SGA shared pool*) SQL lausete salvestamiseks ja andmesõnastiku cache mälu (andmesõnastiku muutuste salvestamiseks)

* Tausta ja kasutajaprotsessid (*backround and user processes*)

* Failid: andmebaasi failid, kontrollfailid ja *redo log* failid



Joonis 1.Oracle 7 arhitektuur

1) Oracle andmefailid. Oracle andmebaas koosneb ühset või mitmest andmebfailist mis sisaldavad:
- andmesõnastikku, mis hoiab informatsiooni andmebaasit endast (andmestruktuurid)
- kasutajate andmeid (nn. *user data*)

Oracle andmefailid luuakse vastavte SQL käskude poolt (Antakse läbi andmebaasi administraatori vahendite - SQL Plusi või SQL*DBA .

Andmefailid on loogiliselt jagatud nn. Oracle blokkideks.

2) Redo log failid salvestavad
- kõik andmebaasi tehtud muudatused

Redo log faile kasutatakse süsteemi automaatseks taastamiseks (*recovery*) peale kokkukukkumisi.

Redo log failidesse ei kirjutata kunagi otse kasutajate poolt

Redo log failid luuakse vastavate SQL lausete puhul (mis käivituvad andmebaasi loomisel)

3)Kontroll fail on suhteliselt väike binaarfail, mis kirjeldab andmebaasi struktuuri
ja mida kasutatakse kasutajatele andmebaasi juurdepääsu avamiseks kasutajatele. See fail peab olema Oracle andmebaasisüsteemi jaoks kättesaadav alati kirjutamiseks, kui andmebaas on avatud.

Kontroll fail identifitseerib
- kõik andmefailid ja log failid
- andmebaasi nime
- on seotud andmebaasi taastamisetega

Kontroll fail on luukse andmebaasi loomise ajal.

Kindluse mõttes peab ühel andmebaasil olema vähemalt kaks kontroll faili, mis sisaldavad identset infot.

4) Parameeter fail - määrab nn. *instance* (vt. järgmine punkt) karakteristikud - eraldatud mälupuhvrite arv ja suurus ning taustaprotssside arv. Parameeterfail on väike tekstifail, mida andmebaasiadn´ministraator võib vabalt regigeerida. Seda faili loeb süsteem ainult Oracle startimisel. Selle faili nimeks on tavaliselt init.ora.

*** Oracle andmebaasi startimine e. mällu laadimine (nn. üles tõstmine)**

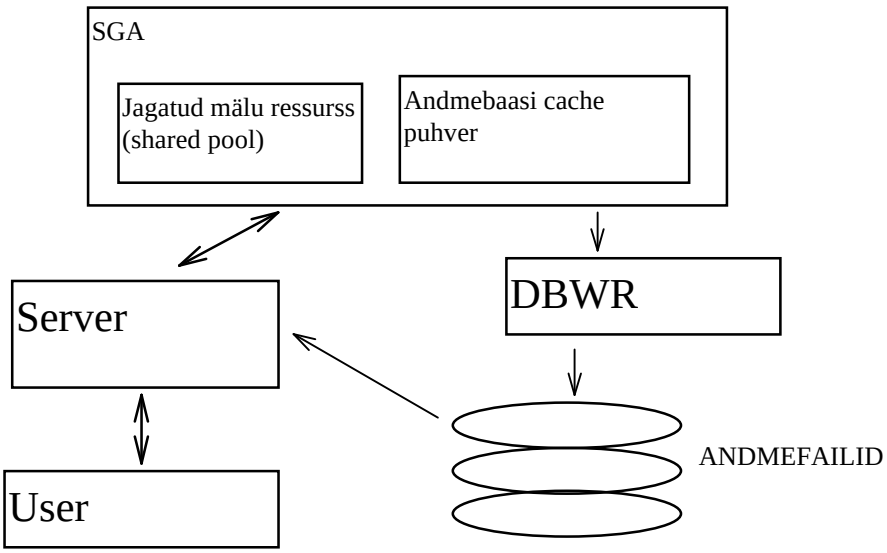
Iga kord, kui Oracle starditakse:
- Eraldatakse mälus süsteemi tööks vajalik piirkond - SGA
- starditakse Oracle taustaprotsessid

Kombinatsiooni taustaprotsessidest ja eraldatud mälupuhvritest nimetatakse inglise keeles *instance*
Instance võib eksisteerida ka ilma andmebaasita (n. kui andmebaasi pole veel loodud)

Alati töötavad vähemalt neli taustaprotsessi
DBWR - andmebaasi, kirjutamine
LGWR - redo log failidesse kirjutamine
PMON - Process Monitor - sulgem ebanormaalselt katkenud ühendused andmebaasiga, "rullib tagasi" mittekinnitatud transaktsioonid, vabastab SGA ressursid, mis on hõivatud "kokkukukkunud" protsesside poolt.
SMON - System Monitor - automaatsed *instance* taastamised, hooldab kontroll faile.

SGA koosneb andmebaasis cache puhvrist ja jagatud mäluressursist.

Cache puhvris hoitakse koopiaid andmeblokkidest, mida loetakse kettalt.



Andmetele juuredpääs (Data Access) : Enne kui andmetele saab juurde, peab serfveri protsess paigutama nad andmebaasi cache mällu. SQL lausete töötlemiseks serveril kasutatakse SGA jagatud mälu ressurssi. Taustaprotsess , andmebaasi kirjutaja (DBWR) kirjutab muudetud andmete blokid tagasi kettale.

Teised protsessid süsteemis:

Serveri protsess:
- tõlgib kasutaja poolt antud SQL laused ja täidab need.
- loeb andmeid kettalt andmebaasi cache puhvrisse (SGA-s)
- tagastab kasutajale SQL lausete tulemused

DBWR
- kirjutab kõik muutunud andmeblokid andmebaasi cache mälust kettale
- kasutab vastavaid algoritme hoidmaks kõige rohkem kasutatavaid andmeblokke kogu aeg mälus.

Andmebaasi jagatud mälu ressurss (shared pool):
sisaldab
- SQL lausete tekste, mida töödeldakse
- SQL lausete tõlgitud, töödeldud (parsed) versioonid
- SQL lausete täitmise plaanid ja algoritmid
- infot andmesõnastik, mis on vajalik SQL-lausete töötlemiseks.

Teema27. *Upsizing* - andmebaasisüsteemi vahetamine võimsama (SQL) andmebaasisüsteemi vastu. Eesmärgid ja tulemused, *upsizingu* käigus tehtavad tegevused ja esilekerkivad probleemid. Vahendid

Upsizing - andmebaasi "tõstmine" koos andmetega võimsamasse andmebaasisüsteemi , andmebaasisüsteemi vahetus. Andmebaasi tabelid realiseeritakse uues andmebaasisüsteemis ning kantakse andmed sinna üle. Enamasti kaasneb *upsizinguga* ka mingi arendustöö, muutuste tegemine süsteemis.

Näiteks: Access → Oracle
Paradox → Interbase
FoxPro → MS SQL Server

Upsizingu eesmärgid (põhjused, miks asi ette võetakse):

- viia andmebaas võimsama andmebaasi mootori "peale", sest suuremate andmehulkade tõttu jääb andmetöötluse jõudlus vana süsteemis madalaks.

- parandada süsteemi tööd võrgus, teha andmebaas kättesaadavaks korraga suuremale hulgale kasutajatele

- parandada andmebaasi turvalisust ja andmete säilimist . SQL süsteemides on turvalisus (kasutajate õigused, rollid, andmete näitamine VIEW-de kaudu jms.) paremini reguleeritav ja kaitsemehhanismid töökindlamad (sissetungimiskindlamad). Andmete riknemise tõenäosus arvutisüsteemi (või mingi konkreetse faili või kõvaketta) rikete tõttu on samuti väiksem - andmete arhiveerimine (Backup/Restore) süsteemis ja automaatne varukoopiate tegemine mitmele seadmele, süsteemi automaatne taastamine (recovering) peale rikkeid, andmebaasi tehtud muudatuste kirjutamine nn. log failidesse jne. Lihtsamates ABS-ides sellised võimalused puuduvad.

- viia kogus süsteemi arhitektuur vastavusse klient/server süsteemide ülesehituse põhimõtetega (andmete tegemine kättesaadavaks paljudele kasutajatele erinevates kohtades, serveri ja kasutajaliidese funktsionaalsuse lahutamine ja jagamine jne.)

- ehitada hajussüsteemi või parandada juba olemasolevaid hajussüsteemilisi omadusi (SQL baase on kergem ühendada hajussüsteemi.)

Tulemused (tulenevad eesmärkidest), millele *upsizi*-tud süsteem vastab:

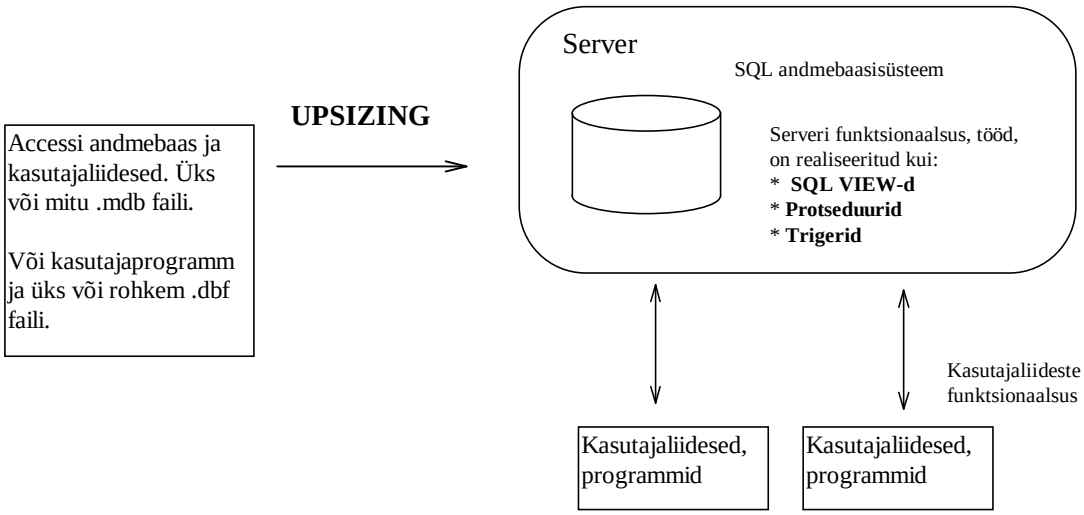
- andmetöötluse jõudlus tõuseb (see ei pruugi olla alati märgatav ja sõltub eritingimustest, vt. altpoolt)

- andme säilimine ja kaitsmine on paremini tagatud (nii mittesanktsioneeritud kasutamise kui arvutisüsteemi rikete eest)

- süsteem on ülesehituse saab nüüd viia vastavaks klient/serveriga

a) eraldada kasutajaliidesed ja serveri funktsionaalsuse (mis enne asusid kõik kliendi arvutis), anda põhilised andmebaasioperatsioonid (päringud, andmeuuendus, andmevahetus teiste allsüsteemidega) **serverile** ja **kliendile** jätta ainult kasutajaliidese loogika ilma suuremate andmetöötlusprotseduurideta. Milleks see hea? Süsteem on optimiseeritud, andmetöötlusega tegeleb selleks tööks paremini sobiv server, mitte klient. Samuti muutub paindlikumaks süsteemi arendamine ja muutmine - kasutajaliideste tegemist ja muutmist saab paremini eraldada serveri (andmebaasi) projekteerimisest

b) liidestada andmebaas paremini teiste allsüsteemidega



- süsteem on nüüd valmis ühendmiseks hajussüsteemi , SQL baasid suudavad omavahel suhelda ka läbi laivõrgu.

Põhiegevused, mis tuleb läbida süsteemi *upsizimise* käigus ja probleemid, mis nende tegevuste käigus ette tulla võivad:

1. andmestruktuuride loomine serveril (CREATE laused). Koos andmetabelitega tuleb üle kanda ka relatsioonid ja andmete õigsuse kontrollid, mis on kirjutatud konkreetse andmebaasi tabelitele.

Mõned probleemid:

* Osa SQL andmebaase (MS SQL Server) ei tunne mõistet CASCADE ON UPDATE ja CASCADE ON DELETE üks-mitu relatsioonide korral. Siis tuleb serverile projekteerida vastavad trigerid, mis selliseid suhteid realiseerivad või teha mitu-poolte kustutamine kliendi programmis (halvem meetod, ka aeglasem)
* Andmebaasi tabelitele kirjutatud kontrollid (Validation Rule, Check vms. , nimetus sõltub andmebaasisüsteemist) vajavad ülevaatamist ja nende süntaks muutmist, juhul kui nad ei sobi konkreetse andmebaasisüsteemiga.

2. kasutajaliideste ja programmide ümbertöötamine nii, et nad kasutaksid uusi serveri andmetabelid. Kliendi ja serveri ühenduste testimine ja silumine.

3. Kasutajaliidestest ja kliendiprogrammidest selliste andmetöötlusprotseduuride eraldamine, mida saaks üla kanda serverile.

Üheks põhiliseks on siin kõikvõimalike kasutajaliideste tasemel konstrueeritud päringutest SQL VIEW-de konstrueerimine ja nende serverile paigutamine. Kasutajaliidesed kasutavad siis neid VIEW-sid, mitte aga nendes endis salvestatud päringuid . (Näiteks Accessi *Query*-id). Andmete kokkuotsimise töö päringusse teeb SQL VIEW korral ära nüüd server.

Samuti tuleks kliendiprogrammidest välja võtta suuremad andmeuuendusprotseduurid - sellised protseduurid , mis töötlevad suuri kirjete hulki andmebaasis. Nendest tuleks teha serveri **trigerid ja protseduurid**.

Siin võivad tekkida ka mõningad **probleemid**:

* Probleem andmevahetusega kasutajaprogrammi ja serveri vahel.Tihti on need operatsioonid, mida server peab andmetega baasis tegema, seotud tihedalt kasutajaliidese loogikaga - näiteks tuleb kustutada see kirje, mida näitab parajasti mingi kasutajaliidese ekraanipilt. Nüüd tuleb peale vastava SQL protseduuri käivitamise kõrval serverile saata ka info (mingi kirje identifikaator, võti), mille alusel protseduur saaks kustutamise ära teha. St. tuleb anda serveri protseduurile **argument**. Argument tuleb samuti saata serverile, sest serverile kirjutatud SQL protseduurist ei saa viidata mingi kasutajaliidesele, ekraanivormile või kasutajaprogrammi muutujale. Protseduuri argumentide saatmiseks serverile saab

mõnel juhul kasutada serveri muutujaid (variables) , juhul, kui konkreetne andmebaasisüsteem seda võimaldab.
Teine võimalus on vastavate abitabelite tegemine serverile argumentide salvestamiseks. Kasutajaprogramm salvestab siis vastavasse tabelisse kõigepealt argumendi ja siis käivitab protseduuri, mis seda argumenti kasutab. Siin võib projekteerimisel tekkida probleeme sellega, et ühte protsedduri (seega ka sama abitabelit) soovivad korraga kasutada mitu kasutajaprogrammi.

* Probleem serverile projekteeritavate protseduuride stabiilsusest, lokaalsusest ja sõltumatusest.
Võib ette tulla olukord, kus mingi andetöötlusprotseduuri vajab (käivitab) ainult üks või väga vähe kliente (kui vaadata suure süsteemi konteksris, kus on palju erinevaid kasutjaid ja programme, mis serveriga ühenduses on). Sanuti võib juhtuda, et selle protseduuri sisu võib aja jooksul kiiresti **muutuda** ja sõltuda rohkem kasutajaliidese loogikast (kui selle protsedduri muutumisega ei kaasne muudatusi serveri andmebaasis). St. protseduur on lokaalne ja tema realisatsioon kiiresti muutuv ning kergemini kasutajaprogrammis realiseeritav. Kuna serveri muutmine suures süsteemis on enamasti kaalukam ja pikemat realiseerimisaega nõudev ettevõtmine, on võimalik, et need lokaalsed protseduurid realiseerida (esialgu) kasutajaliideste vahenditega. Arvestada tuleb siin sellega, et server on alati mingis mõttes tsentraalne ja ühine mitmete kasutajate peale. Seetõttu tuleb ette näha, milline mõju on mingil andmete uuendamise protseduuril kogu süsteemile (teistele serveri protseduuridele, andmetabelitele ja nende trigeritele jne. **Selles mõttes on võimaluse korral siiski parem anda protseduur serverile ja protseduuri realiseerimine serveri projekteerijale**.Serveri protseddur on loomulikult ka kiirem .

* Probleem kasutajaliideste(arendamise) vahendite piiratud võimalustest.
Kasutajavahend peaks võimaldama serveri ja kliendi tihedat integratsiooni, mis tähendab eelkõige:
- kliendi vahendiga saab käivitada serveri protseduure (otse, ilma SQL trigeri vahenduseta.
- andmevahetus on kliendiprogrammi ja serveri vahel hea (vt. eespoolt probleemi andmevahetusega).
Peaks saam vahetada peale tabelisse kirjutatavate andmete ka muutujaid. Siin seab omad tõkked vahel ette ODBC (võib olla konkreetne ODBC draiver, mis kõike vajalikku ei võimalda.) Teine lugu on OLE-ga.

Abivahendid - Microsoft Access Upsizing Tool