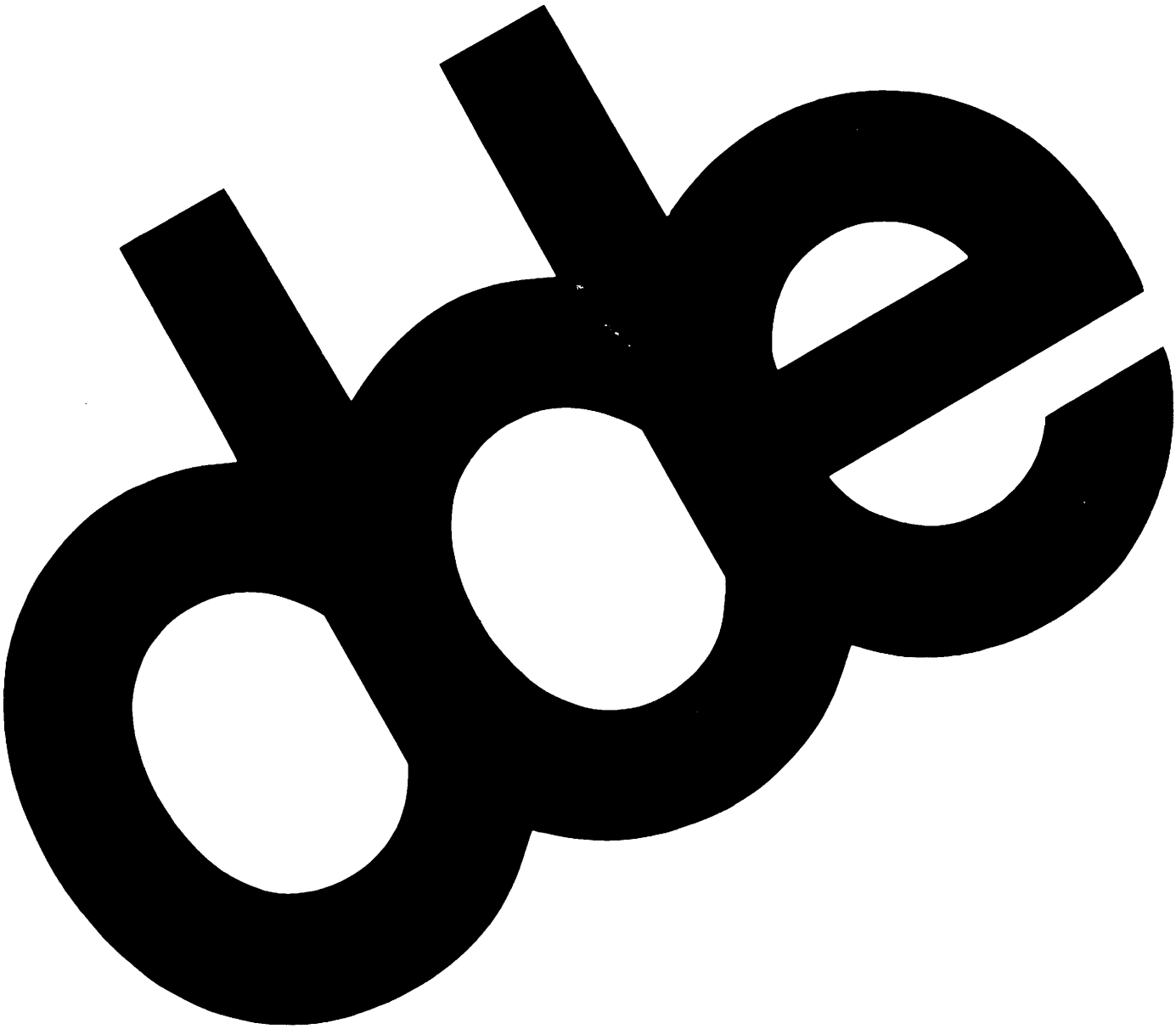
Oracle

Databaseadministration I



29. august 1996

JESPER FROST LUND
email: jesperf@dde.dk

30 sep → 2 okt 1997

SYBASE
CDC
Oracle
DBL

Anbefaler PowerBuilder specielt den nye ver. 5.2 har klasser til
håndtering af datavinduer

Ver 5.0 er meget fejlbehæftiget

---

navn : srv1 , bruger pt2 OPS$pt2
ORACLE_SID = MTS
sqlplus : [illegible]
ora 1-5 / ora 1-5

segment : noget (objekt der [illegible]) der [illegible] i-lads

Indholdsfortegnelse

1. Introduktion til DBA arbejdet ..... Side 3
   - Ansvarsområder
   - Databasen
   - Hjælpeværktøjer
   - Særlige DBA-brugere i Oracle

2. ORACLE7 arkitekturoversigt: ..... Side 9
   - Identifikation og miljø
   - Databasens logiske struktur
   - Fysisk struktur og filer
   - Memory arktektur
   - Process struktur
   - init.ora filen
   - Opgave

3. Data dictionary med mange gode eksempler, der understøtter det gennemgåede i afsnit 2 ..... Side 23

4. Startup & shutdown ..... Side 29
   - Opstarts parametre
   - Særlige hensyn

5. Database objekter/segmenter og pladsallokeringer ..... Side 35
   - Tabeller, extents og storage parametre
   - Index, B+tree princippet, extents og storage parametre
   - Rollback segmenter og storage parametre
   - Temporære segmenter
   - Datablok arkitekturen - PCTFREE, PCTUSED & PCTINCREASE

6. Space management og mere om datadictionary ..... Side 47
   - Default storage i tablespaces
   - Diverse eksempler

7. Brugere, privilegier, sikkerhed og administration ..... Side 53
   - Brugeroprettelse (inkl. nye defaults)
   - System rettigheder
   - Objekt privilegier
   - Roles
   - mere data dictionary

8. Databaseoprettelse ..... Side 57
- Eksempel på databaseoprettelses script
- Eksempel på init.ora filen

9. Backup, restore og recovery ..... Side 61
- Logisk backup med exp/imp (også incremental/cummulative)
- Fysisk offline backup (dskback)
- Intro til archiving og online backup / recovery

10. Overvågning ..... Side 67
- SQLDBA Monitor
- flere SQL-scripts og data dictionary eksempler

11. Introduktion til særlige ORACLE7 specialiteteter: ..... Side 69
- Constraints, Stored procedures & triggers
- Applikationsbalancering i Client/Server miljøer
- Memory/process udnyttelse (MTS)
- Distribuerede databaser

12. Appendix ..... Side 71
- Ofte stillede spørgsmål og svar
- Hvad får man på næste kursus (DBA II v7)

13. Opgaver og praktiske øvelser undervejs ..... Side 75

1. Introduktion til DBA arbejdet

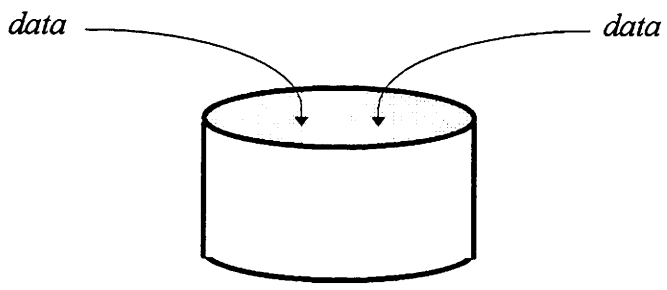
Ansvarsområde

En databaseadministrator (DBA) har typisk ansvaret for den daglige drift af en eller flere databaser i en organisation. Normalt vil der altid være tale om *flere* databaser, da man typisk har minimum ét produktionsmiljø og ét testmiljø, hvor begge miljøer skal køre så optimalt som muligt for at understøtte de brugere og udviklere, der er afhængige af de data ligger heri. DBA'ens opgaver indbefatter eksempelvis.:

- at installere og vedligeholde software
- at planlægge et optimalt databasemiljø til både produktion og test/udvikling
- at overvåge og regulere performance
- at overvåge pladsforbrug
- at oprette, fjerne og vedligeholde brugere, herunder tildeling af rettigheder og passwords
- at garantere sikkerheden på databasen og det omgivende miljø
- at tage sikkerhedskopi regelmæssigt
- at vejlede udviklere og brugere omkring brugen af database miljøet
- at retablere databasen efter software- eller mediefejl

Databasen

For en god ordens skyld skal vi lige ganske kort definere, hvorledes data lagres i en Oracle database. Groft sagt kan man betragte en database som en "**spand**", hvori man hælder data og informationer ned til opbevaring for senere at kunne fremfinde og/eller manipulere disse fra brugernes værktøjer og applikationer.



En database består af en programkerne, der omslutter et fysisk miljø (dvs. filer, devices eller diske), således at man kan betragte dette lagringsområde som ét logisk objekt. Med andre ord vil man som bruger og udvikler altid se databasen og dennes data på et logisk niveau uden at skulle bekymre sig om den fysiske placering eller udformning. Det er derfor ikke nødvendigt at vide på hvilket operativssystem databasen kører for at benytte dens data.

I databasen vil data altid forefindes i navngivne "tabeller", der er opdelt i kolonner og rækker. En tabel kan have optil 255 kolonner, hvor de forskellige kolonner kan indeholde forskellige datatyper (f.eks. numeriske data, karakterdata eller kolonner med datoer). Hver kolonne identificeres med et logisk navn. Kolonnerne identificeres ved et kolonnennavn, og de enkelte rækker kan kun identificeres gennem et søge-kriterie mod selve dataindholdet af en eller flere kolonner. En tabel i en database kan eksempelvis se således ud:

Tabellen firmaer:

Firma	Adr	Postnr	Telefon
Pointercast A/S	Nyhavn 67 C	1051	0311 3666
A.T. Mearney	Midtermolen 3	2100	1543 0600
TT business service	Kompagnistræde 45	1208	2391 0090
ActivCare	Nyvej 71 st.	1851	1131 3174
Allergro Service	Hovedvejen 106	2600	2343 1599
Alternativ Group ApS	Ørnevej 12	2100	2834 8500
Personaleservice A/S	Falkoner Allé 45	2000	1888 9400
B&K Recruting	St. Kongensgade 22	1264	1311 2100
PMK Danmark	Grønnegade 3 A	1051	1312 4711
CAM Kompagniet	Dampfærgevej 7	2100	2526 7100
Repass Rekruttering	Bydammen 10	2800	2368 6670
Consensur	Lyngby Hovedgade 14 B	2800	1593 4144
CP Personaleservice	Fasanhaven 23	2800	2965 6173
Mano Engineering	Hansdalsbrovej 12	2800	1156 5255

Læg mærke til at bynavne mangler i tabellen firmaer. Tabellen er blevet normaliseret for at undgå data-redundans, og der kan således trækkes "relationer" til en anden tabel, hvori man kan finde bynavnene ved hjælp af postnummeret:

Tabellen post:

Postnr	Bynavn
1051	København K
1208	København K
1264	København K
1851	Frederiksberg C
2000	Frederiksberg
2100	København Ø
2600	Glostrup
2800	Lyngby

- Det er her begrebet "relationsdatabase" kommer ind i billedet. Oracle har leveret realations-databaser i over 15 år - alle baseret på SQL som forespørgselssprog. Det er i SQL sætningerne man fortæller hvad man vil se og hvor meget. Her ønsker vi kun at se 4 bestemte kolonner fra de to tabeller - og kun de rækker der opfylder et bestemt søgekriterie:

```
SQL>  select f.firma, f.adr, p.postnr, p.bynavn
      from firma f, post p
      where p.postnr = f.postnr
      and p.bynavn like 'København%';
```

Hvis du har problemer med at forstå hvad der sker i ovenstående SQL sætning, så er et besøg på et SQL-kursus nok en god ide!

Generelt kan man sige at nutidens databaser udmærker sig ved;

- at kunne håndtere store datamængder.
- at kunne håndtere trivielle "space management" opgaver, som f.eks at genbruge og udnytte pladsen fra slettede rækker, at allokere ny plads til nye rækker osv.
- at kunne håndtere mange samtidige brugere.
- at levere et højtransaktionsmiljø i forbindelse med manipulation af on-line data.
- at tilbyde høj tilgængelighed.
- at sikre datakonsistens og integritet.
- at vedligeholde tabellernes indexstrukturer automatisk.
- at omgive data med et sikkerhedssystem der følger nutidens standarder.
- at overholde defacto- og indutristandarder.
- at danne basis for client/server miljøer.
- at tilbyde mulighed for at distribuere data over flere lokationer.
- at tilbyde lokationstransarens i distribuerede miljøer.
- at være ens på det logiske niveau uanset operativsystem eller maskintype.

Hjælpeværktøjer

Databaseadministratoren har et antal værktøjer til sin rådighed fra Oracle's side. De traditionelle værktøjer er hhv.:

- **SQL*Plus**
- **SQL*DBA**
- **PC-Windows baserede DBA værktøjer** (primært i clent/server miljøer)

Fra disse programmer kan DBA'en udføre de SQL sætninger, der skal til for at administrere en Oracle database. SQL*Plus er dog det meste anvendte når der udelukkende afsendes SQL sætninger afsted mod databasen, da man her har mulighed for at redigere de enkelte sætninger linie for linie en en tilnyttet editor (typisk vi-editoren i et Unix-miljø). SQL*DBA-programmet er et specialværktøj med et karakterorienteret menusystem til afvikling af forskellige databaseadministrationsopgaver. Hertil findes der en række grafiske DBA værtøjer til PC-Windows og andre GUI platforme, der kan klare databaseadministrtionsopgaverne blot ved at klikke på knapper, ikoner eller hypperlinks.

SQL*Plus-programmet bruges bl.a. til følgende administrative opgaver udført med SQL-sætninger:

- at vedligeholde brugere (`create/alter/drop user.., create/alter profile...`)
- at administrere sikkerhed (`grant/revoke.., create/drop role.., set role...`)
- at oprette tabeller og indeks (`create/alter/drop table/index/view...`)
- at udvide og ændre databasen (`create/alter/drop database/tablespace...`)
- at søge i Oracles interne tabeller (data dictionary) efter oplysninger om fx konfiguration, definitioner og pladsforhold (`select * from dba_xxxx/all_xxxx/user_xxxx...`)
- og meget andet...

SQL*DBA-programmet bruges bl.a. til følgende administrative opgaver udført fra menupunkterne, som særlige DBA-kommadoer eller som SQL sætninger:

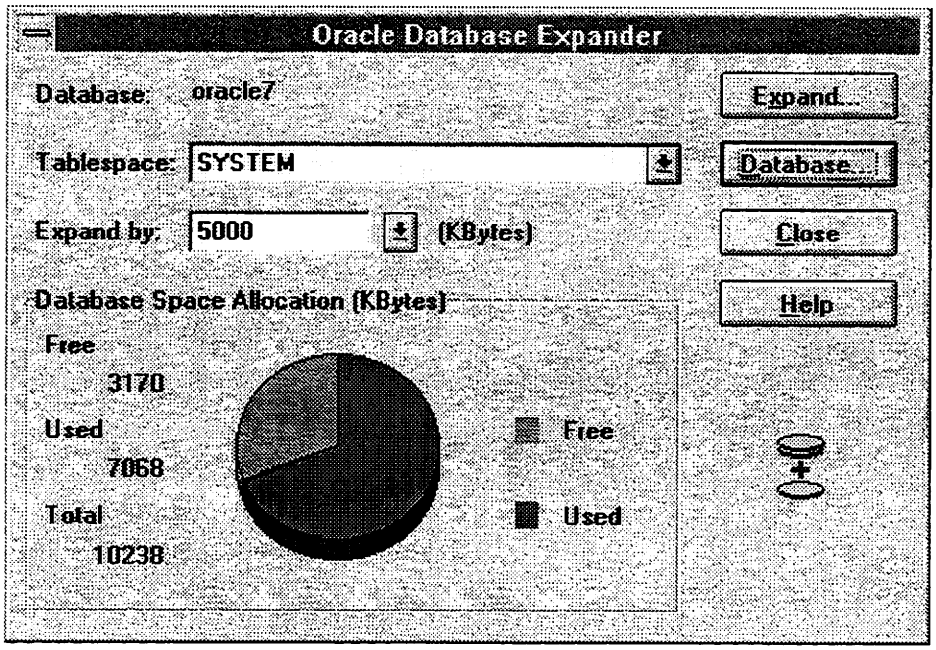
- at logge på systemet selv om databasen ikke kører (`connect internal...`)
- at starte og stoppe eksisterende databaser (`startup/shutdown...`)
- at overvåge aktiviteten på basen (`monitor...`)
- at udføre recovery efter software- eller hardwarefejl (`archive log/recover...`)
- at udføre samme SQL sætninger som i SQL-Plus (se forrige side)

SQL*DBA-programmet kaldes med kommandoen **sqldba.** De fleste opgaver, der kan udføres herfra er priviligerede kommadoer og kræver, at der logges på basen som dba-bruger. Meget priviligerede kommandoer som `startup/shutdown` kræver endvidere at SQL*DBA-programmet kaldes fra en bruger, der tilhører unix-gruppen **dba**.

DBA-værktøjer på PC-Windows og andre GUI platforme kan stort set det samme som SQL*Plus og SQL*DBA, men udmærker sig ved at visualisere status og informationer grafisk over for DBA'en. Disse værktører forudsætter dog en netforbindelse til databaseserveren i gennem SQL*Net eller lign. Af værktøjsleverandørere kan nævnes:

- Oracle Corp. / DDE A/S
- Platinum Technologies Inc. (Desktop-DBA)
- Embaracado Technologies Inc. (DB Artisan) ✓
- Bradmark Technologies Inc. (DB General for Oracle)

Blot et eksempel på et tilfældigt DBA-skærmbillede fra et PC-Windows miljø:



Særlige DBA brugere i Oracle

Der findes to standardbrugere fra Oracle-installationen, som har DBA-privilegier (databaseadministrator privilegier) i databasen. Disse brugere er:

- **sys**
- **system**
- internal (har default ikke password, kan det samme som 'sys',

Almindelige brugere, må derfor aldrig logge på basen som **sys** eller **system**. hvis man er privilegeret os-bruger –

Databaseadministratoren bør normalt operere som brugeren **system**, når der skal udføres administrative opgaver på databasen. Brugeren sys ejer alle de tabeller og views databasen selv bruger internt (også kendt som "data dictionary"). Som hovedregel må ingen logge på som brugeren **sys**, da denne bruger direkte kan ændre indholdet af data dictionary tabellerne.

De to brugere **sys** og **system** er fra Oracle's side installeret med standard passwords, hhv. ***system/manager*** og ***sys/change_on_install***. Det er meget vigtigt, at disse passwords bliver ændret efter installationen. Ellers er der i praksis fri adgang til databasen, da alle som har arbejdet med Oracle kender disse passwords.

(Se afsnit. 7, ændring af passwords m.v.).

Kursusejet Unix-maskine: ejer af oracle DB er 'ora7' unix group 'dba'

Denne side er blank med vilje !

2. Oracle7 arkitekturoversigt

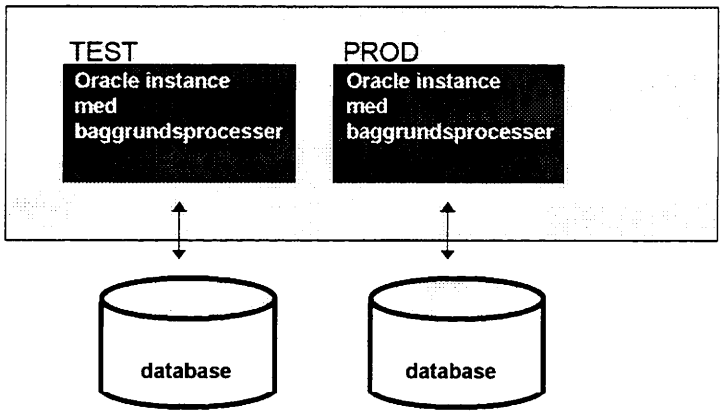
Identifikation og miljø

En kørende Oracle database kan nemt genkendes i operativsystemet, og består reelt af flere delkomponenter:

- et antal processer i operativsystemet som udgør databasens kerneprocesser (også kaldet "**bagrundsprocesser**")
- de fysiske filer eller rå-diske som udgør selve databasen (også kaldet "**datafiles**")
- et reserveret arbejdsareal i memory (også kaldet "**SGA**", som står for Sytem Global Area)
- nogle småfiler, der benyttes internt af Oracle til logning og kontrol information (også kendt som "**redo logs**" og "**kontrol filer**")

Når en Oracle database startes, får den et navn, som angiver dens unikke identitet på maskinen. Det er primært for at brugere kan finde frem til den rigtige Oracle database, hvis der køre flere på samme tid. En Oracle database kerne kaldes et "**instance**" og består af "SGA" plus "baggrundsprocesserne".

En maksine (f.eks. UNIX), hvor der kører to Oracle databaser



En bruger angiver ved hjælp af såkaldte "environment variabler", hvilket Oracle-instance han/hun ønsker at blive koblet op på i forbinelse med logon/connect proceduren på Oracle.

Environment variablerne kan defineres i filen "`/etc/profile`" som er **fælles** for alle unix-brugere eller "`.profile`" filen, som skal sættes **individuelt** hos de enkelte brugere, der vil i kontakt med Oracle miljøet. De vigtigste environment variabler er:

- **ORACLE_SID**
- **ORACLE_HOME**

Tilsammen identificerer disse entydigt et Oracle instance med tilhørende database. ORACLE_HOME identificerer basens hjemmekatalog, hvor bl.a. Oracles kerneprogrammer ligger. ORACLE_SID er en tekststreng på højst 4 tegn, f.eks. TEST eller PROD, der fungerer som et database-ID.

Eksempel på relevante linier i enten "`/etc/profile`" som er fælles for alle unix-brugere eller "`.profile`" filen:

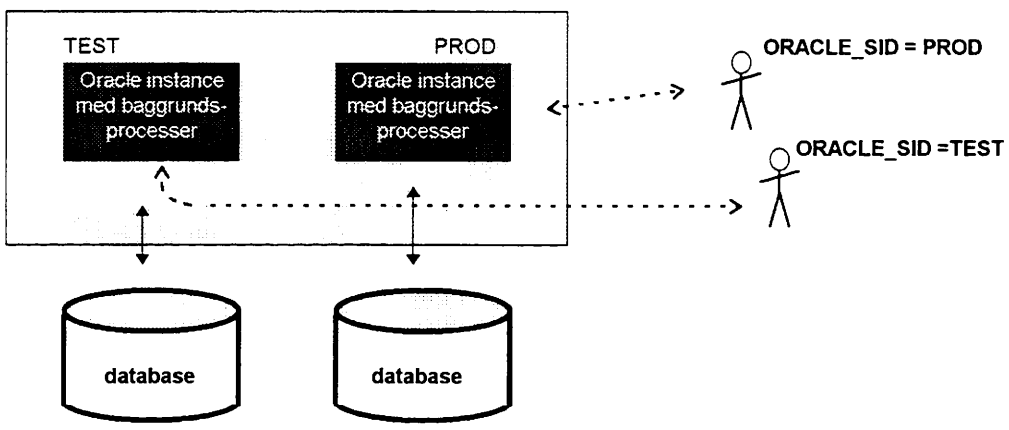
```
ORACLE_HOME=/usr/oracle; export ORACLE_HOME
ORACLE_SID=TEST;         export ORACLE_SID
ORAENV_ASK=NO;           export ORAENV_ASK
. /usr/bin/oraenv
```

Tilføjes variablen ORAENV_ASK=NO som ovenfor, vil brugeren ikke blive promptet for værdien af ORACLE_SID og ORACLE_HOME ved logon/connect.

Alternativt, kan variablene sættes umiddelbart før kaldet af hver enkelt applikation. En brugers Oracle-opsætning, kan læses med UNIX kommandoen:

```
$ env ^ grep ORA
```

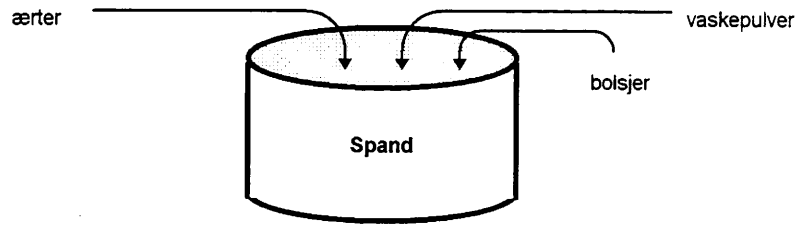
To brugere, der forbinder sig til hver sin database:



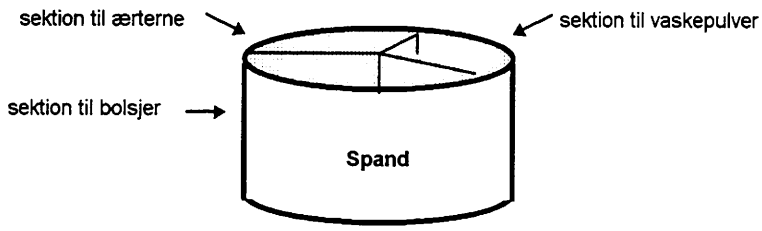
Databasens logiske struktur

Som nævnt tidligere kan man groft sagt betragte en database som en "spand" eller en beholder hvori man opbevarer data og informationer.

I det virkelige liv hvor man reelt benytter spande til at opbevare ting og sager, er det kun meget få menesker der opbevarer forskelligartede ting i én og samme spand. Tænke hvis nu man forsøgte at opbevare sager som "ærter", "bolsjer" og "vaskepulver" i samme spand. Det ville være lidt af en opgave at hente "ærterne" op fra denne spand:

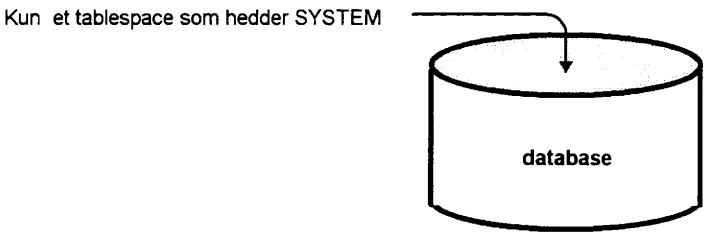


Det ville være noget nemmere hvis man nu forestillede sig at denne spand var indelt i indiviuelle rum eller sektioner, således at tingene ikke bliver sammenblandede. Således vil man lettere kunne hente "ærterne"op fra spandens sektion:



Sådan er det også for Oracle databasen...

På en helt almindelig standard installation af en Oracle database oprettes selve databasen også kun med en enkelt sektion til opbevaring af data og information i form af tabeller og index. En sådan opbevaringssektion kaldes et "**tablespace**". Et tablespace har et logiskt navn, således at man kan referere hertil i forbindelse med oprettelse af f.eks. tabeller eller index. Oracles eneste obigatoriske tablespace fra en standardinstallation hedder SYSTEM. Således ser en Oracle database ud, hvis man overhoved intet gør ved den efter installationsprocedurens database-oprettelse:



Der er en del forskellige informationstyper og objekter, der opbevares i en database;

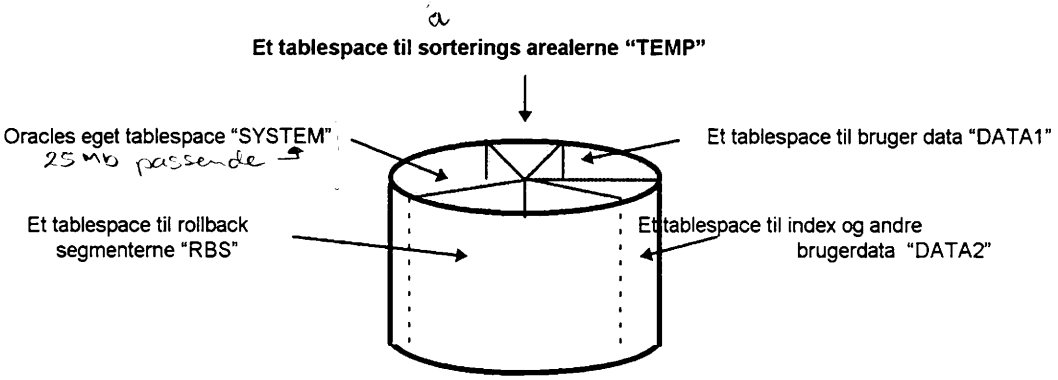
- **Data dictionary**. Oracles eget systemkatalog bestående af interne tabeller til styring af selve databasen og miljøet heromkring.

- **Bruger segmenter**. *Tabeller*, *index* og *clusters* oprettet af udviklerne som dataregistre til de forskellige brugere, værktøjer og applikationer der benytter databasen. I nogle miljøer kan det være en fordel at holde tabeller og index adskildt.

- **Temporære segmenter**. Midlertidige arbejdsarealer der anvendes af systemet i forbindelse med store data sorteringer og index oprettelser. Systemet opretter og fjerner selv temporære segmenter efter behov.

- **Rollback segmenter**. En slags "konsistensbeholdere" som benyttes af systemet til at opbevare øjebliksbilleder af data under opdaterende transaktioner, således andre brugere ikke ser dataændringerne før en transaktion helt afsluttes med COMMIT. Rollback segmenterne bliver også brugt hvis en transaktion fortrydes med en ROLLBACK kommando. Det vil her fra de "gamle" data læses fra.

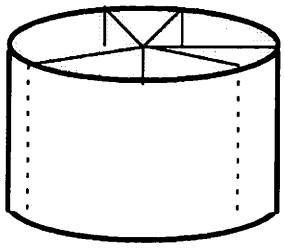
Således kan man med fordel opdele en database i flere tablespaces for at kunne adskille de forskellige segment- og informationstyper fra hinanden. Altså på samme måde som man kunne inddele den før omtalte spand i opbevringssektioner:



Databasens fysiske struktur og filer

Et tablespace er fysisk lagret i minimum en fysisk **datafil** eller **rå-/subdisk**, og det er størrelsen på denne fysiske allokering der udgør størrelsen på et tablespace. En database med f.eks. fem tablespaces vil altså have minimum fem filer eller rå-/subdiske i det fysiske miljø:

Logisk ser databasen således ud med de 5 tablespaces:



Fysisk består databasen af min. 5 datafiler eller rå-/subdiske:

/u1/oracle7/dbs/system_ts.dbf	(25 Mb)
/u1/oracle7/dbs/data1_ts.dbf	(200 Mb)
/u2/andendisk/dbs/data1_ts.dbf	(300 Mb)
/u1/oracle7/dbs/temp_ts.dbf	(50 Mb)
/u1/oracle7/dbs/rbs_ts.dbf	(50 Mb)

Databasens totale allokering i dette eksempel er ca. 625 Mb

Et tablespace oprettes med kommandoen:

```
create tablespace navn
datafile 'filnavn el. subdisk' size n
default storage ( initial n
                  next n
                  pctincrease n
                  minextents n
                  maxextents n);
```

- normalt benyttes kun:

```
create tablespace navn
datafile 'filnavn el. subdisk' size n ;
```

Datafiler: Angives et filnavn vil Oracle selv allokere og danne denne fil i operativsystemet.

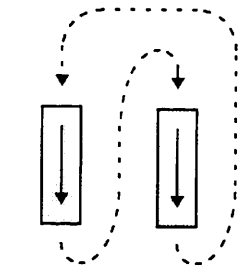
Subdiske: Angives en reference til en rå-/subdisk skal denne eksistere i forvejen.
*Ved brug af rå-/subdiske: Husk at størrelsesangivelsen skal være **8K mindre** en det disken fysisk er allokeret med af hensyn til headerinfo,*

Andre filer i Oracle Miljøet

Ud over datafilerne der tilsammen udgør databasen, benytter Oracle også nogle andre filer, der har helt specifikke formål:

- **Redologs:** Heri noteres samtlige transaktioner på databasen.

  En Oracle7 database har mindst to filer eller rå-/subdiske afsat til redolog-filer. Redolog-filernes primære formål er at beskytte databasen i tilfælde af nedbrud, idet redolog-filerne gemmer information og data fra de transaktioner, der løbende bliver udført på basen. Når en af redolog filerne er fyldte fortsætter Oracle på den anden, og når den er fyldt fortsætter Oracle på ny i den første redolog fil.



  Da Oracle skriver til de underliggende datafiler asynkront (periodisk) kan der være data i memory, som er ændret af diverse transaktioner, som endnu ikke fysisk er blevet nedskrevet heri. Ved et eventuelt strømsvigt eller lign. vil man altså miste disse opdateringer. - Og det er her redolog filerne kommer ind i billedet.

  Når Oracle skriver til og synkroniserer datafilerne kaldes dette et "**checkpoint**". Der er minste et checkpoint for hver gang en redolog fyldes. Med udgangspunkt i seneste checkpoint (synkronisering) kan databasen startes igen, hvorefter den vil læse i redolog filerne for at genskab de tabte opdateringer. Dette kaldes "**roll-forward**" eller "**recovery**" og gennemgåes senere i materialet.

- **Kontrol filer**: Kontrolfilerne er små binære UNIX filer, der læses hver gang databasen startes og som løbende opdateres ved checkpoints. Filernes indhold er identiske og indeholder bl.a. oplysninger om placeringen af de fysiske datafiler og redologs samt information om seneste checkpiont. De er meget vigtige for databasen. Ødelægges kontrolfilerne er databasen principielt tabt ! (kan muligvis re-etables med `create controlfile` komandoen)

Memory arkitektur

Oracle's reserverede memoryareal Sytem Global Area der også kaldes "**SGA**" arealet, er det areal hvor de data der bearbejdes af systemet og brugeren befinder sig på behandlings-tidspunktet . SGA arealet er primært delt op i fire dele:

- **Data Buffer Cache:** Her ligger alle aktuelle datablokke indeholdende brugerdata. Når en bruger udsteder en SQL sætning hvor han/hun ønske at læse dele af indholdet i nogle tabeller, læses der altid her fra.

  Findes de datablokke brugeren skal bruge er det jo bare alletiders. Brugeren behøver ikke at vente på en fysisk læsning fra en eller anden fil eller disk. Brugeren har i dette tilfælde oplevet en **logisk** læsning.

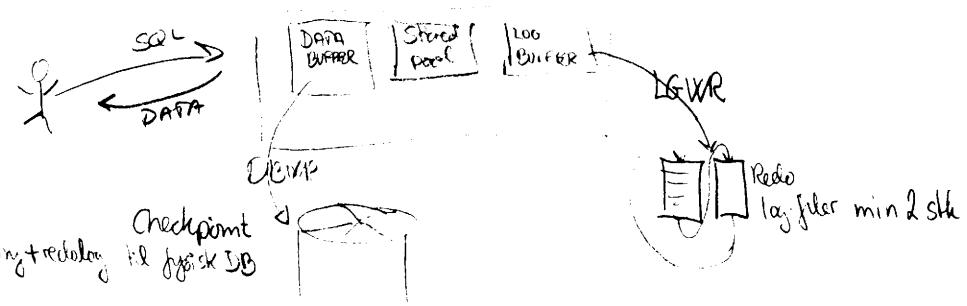
  Hvis de derimod ikke findes i data buffer cachen, må systemet så fremskaffe de nødvendige datablokke fra datafilen på disken således at brugeren kan få sine data og informationer. Dette kaldes en **fysisk** læsning og tager længere tid at udføre.

  SGA arealet fyldes op med datablokke efter behov og efter en algorime, der kaldes for "Least Resently Used" (**LRU**) algoritmen. Det betyder at hyppigst anvendte datablokke overlever længst tid, hvis der er kamp om pladsen heri, - og det er der for det meste. Nye datablokke vil altid overskrive de datablokke som har ligget i længst tid uden at blive læst. Ændrede datablokke er mere sejlivet, da disse opdateringer jo først skal synkroniseres/skrives ned i datafilerne ved et kommende checkpoint.

- **Dictionary Cache** Systemets eget "data buffer cache" til at have de mest aktive dele af data dictionary i SGA arealet. Principperne for dictionary cache er de samme som for data buffer cachen beskrevet ovenfor.

I Oracle er det en del af 'Shared pool'.

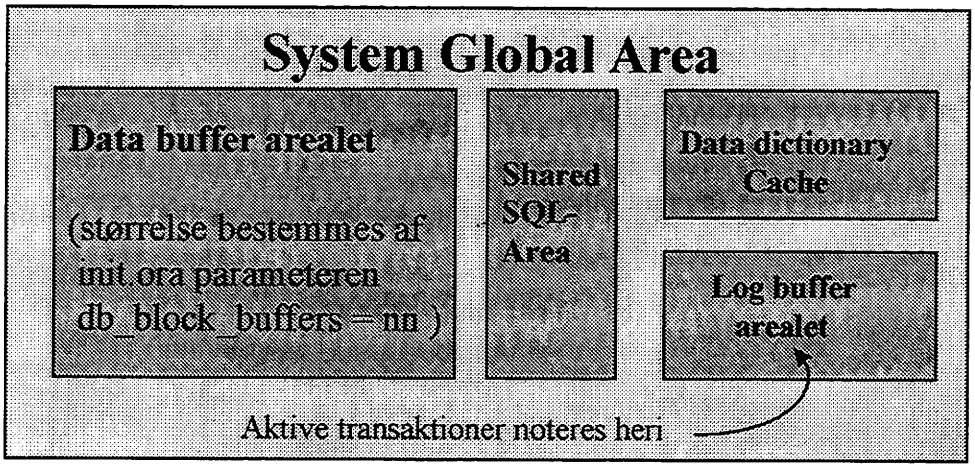
- **Log Buffer Cache** En buffer elller et "stødpudeareal" til at holde information om de aktive transaktioner. Først når en transaktion afsluttes med `commit` vil transaktions informationen blive skrevet ned i redologgen. Oracle prøver således at undgå skrivinger i redologgen hvor trasaktionerne rulles tilbage / fortrydes med `rollback` kommandoen. Der er dog ikke altid det er tilfældet, da Oracle har en rækkes consistens hensyn at tage over for de andre transaktioner der kører samtidigt.



- **Shared pool**: I Oracle7 (modsat Oracle v.6) er der afsat et areal i SGA'en til at opbevare selve SQL sætningerne og procedurekaldene som brugerne sender afsted mod databasen. SQL sætninger bliver fortolket optimeret og oversat (**parsed**) til pseudokode, og dette opbevares ligeledes heri.

  Ideen med dette shared pool areal er at brugere, der abejder med sammen applikation (f.eks et personalesystem) ofte sender samme SQL sætninger afstedt til databasen. Når én af brugerene har fået parsed en SQL sætning, så kan andre udnytte denne pseudokode frem for at de også skal parse samme SQL. Der er performance at hente hvis det udnyttes rigtigt.

Et samlet overblik over Oracle's reserveret memory areal:



En oversigt over SGA'ens allokering vises ved opstart. Man kan også selv undersøge SGA'ens allokering med komandoerne:

```
SQLDBA> show sga
```

eller

```
SQLPLUS> select * from v$sga;
```

Samspillet mellem det fysiske miljø (datafiler, redologs, konrolfil, brugere m.v.) og SGA arealet styres af et antal bagrunds- og brugerprocesser i operativsystemet. Oracle's baggrunds-processer gennemgåes på næste side...

Processarkitektur

I et Oracle miljø skelnes der mellem to forskellige typer processer; brugerprocesser og baggrundsprocesser (også kaldet Oracle processerne). Der vil være et sæt baggrundsprocesser for hvert Oracle instance på maskinen.

- **Brugerprocesser**: Enhver bruger der logger på Oracle database får sin egen bruger-process, hvorigennem han/hun (via sin applikation eller sit værktøj) kommunikerer med Oracle miljøet. Det er denne process der sørger for parsning af SQL sætningerne og henter data op til data buffer cachen i tilfælde af fysiskelæsninger.

  Til enhver brugerprocess hører en "skyggeprocess". Oracle på en Supermax afvikles som en såkaldt two-task arkitektur, hvilket vil sige, at alle brugerprocesser afføder en skyggeproces i Oracle kernen, der har adgang til data i SGA'en for brugerprocessen.

- **Oracle processer**: Oracle har selv et antal baggrundsprocesser til at varetage de forskellige opgaver i database driften.

  Eksempelvis kan nævnes:
  - Synkronisering af datafiler (checkpoints) - Udføres af **DBWR**
  - Skrivning af transaktioner til redologgen. - Udføres af **LGWR**
  - Overvågning af brugerprocesser - Udføres af **PMON**
  - Overvågning af Oracle egne registre - Udføres af **SMON**
  - Håndtering af recovery ved 2-phase commit - Udføres af **RECO**
  - Evt. løbende backup af redologs - Udføres af **ARCH**
  - o.s.v...

Note: Oracle7 kan iøvrigt konfigureres til at køre med en "Multi Threaded Server" (**MTS**) opsætning, hvor flere brugere kan deles om en process. Dette gennemgåes på kurset "*Oracle7 Databaseadministration 2*" hvorfor alle eksempler heri dette materiale tager udgangspunkt i en simplere konfiguration uden MTS - altså en brugerprocess pr. bruger

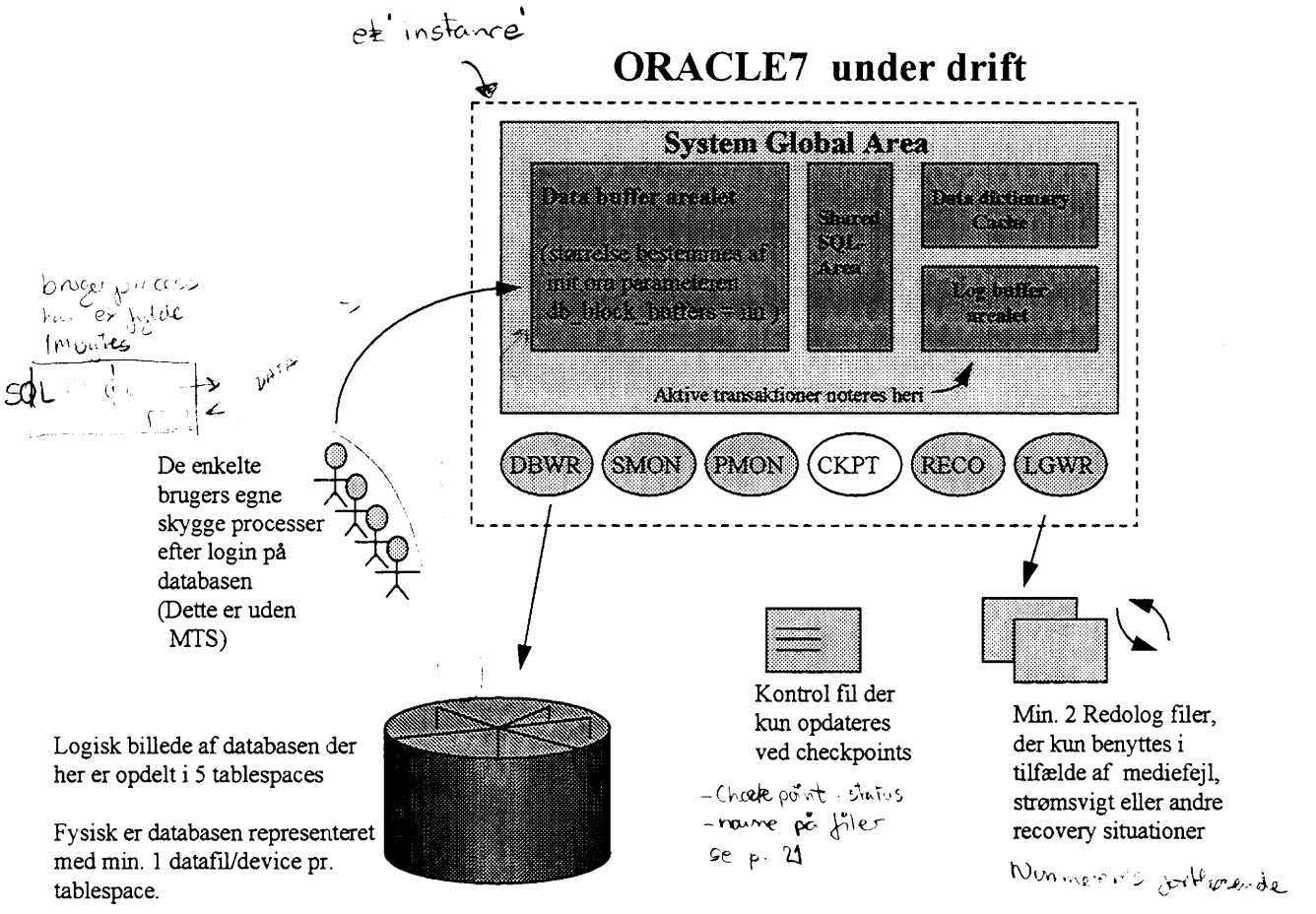
Ønsker man at se baggrundsprocesserne fra det instance der hedder (ORACLE_SID=TEST), udføres følgende UNIX kommado:

```
# ps -ef ^ grep TEST
```

Hvis TEST basen er kørende skal følgende processer eksistere:

```
ora_pmon_TEST
ora_lgwr_TEST
ora_smon_TEST
ora_dbwr_TEST
ora_reco_TEST
o.s.v.
```

En oversigt over sammenhængen:



Baggrundsprocessernes funktioner kan beskrives således:

DBWR (Database Writer)
Skriver modificerede blokke fra data buffer cachen ned på disken ved checkpoints, og sætter checkpoint markeringer i kontrol filen, redolog filerne og datafilerne. DBWR arbejder asynkront i forhold til opdateringer og kommits.

LGWR (Log writer):
Skriver log buffer blokke til redolog filerne når der kommittes, når der ikke er flere frie redolog buffers og når den får besked fra DBWR om at gøre det jf. ovenstående.

PMON (Process monitor)
Rydder op efter aborterede brugerprocesser (process recovery), og ruller ukommittede transaktioner tilbage, frigiver låse og SGA ressourcer hvis der er en bruger der forsvinder ud i det blå.

SMON (System monitor) Control-processen der styrer de øvrige
Denne process udfører instance recovery, opretter og nedlægger temporære segmenter ved store sorteringer. Den adminstrerer også et antal låsningsmekanismer og interprocess kommunikation.

ARCH (Archiver) som er valgfri
Arkiverer online redolog filer. Er kun aktiv når databasen kører i archivelog mode.

CKPT (Checkpoint) som også er valgfri
Kan overtage håndtering af checkpoints fra LGWR og DBWR i højtransaktionsmiljøer. Er kun aktiv hvis man sætter `CHECKPOINT_PROCESS = TRUE` i Oracle's parameter fil (init.ora) som gennemgåes på næste side.

Recovery (RECO)
Udfører bl.a. recovery på distribuerede transaktioner som ikke fuldføres pga. netværksfejl.
distribuerede miljøer

ora_SXXXX_<SID>
ora_dXXXX_<SID>
} multithreaded server

Parameterfilen init.ora

Når et Oracle instance startes henvises der til en bestemt fil indeholdende de nødvendige parametre til databasen og dets miljø. Filen referers til som "**init.ora**", men vil oftest ligge i et helt bestemt direktory og have et navn svarende default navnet på et bestemt instance. Derfor er default formatet for filnavnet:

```
$ORACLE_HOME/dbs/init$ORACLE_SID.ora
```

- hvilket betyder at Oracle altid vil lede i ./dbs directory'et under det angivet i environment varablen `$ORACLE_HOME` for at finde og læse en fil der hedder `init$ORACLE_SID.ora`, hvis intet andet angives under opstart (Se iøvrigt kapitel 4)

Ekempel:

```
$ORACLE_HOME = /u1/oracle7
$ORACLE_SID  = TEST
```

default parameter filen for TEST databasen er altså:

```
/u1/oracle7/dbs/initTEST.ora
```

Der findes over 125 forskellige parametre der kan sættes i denne fil, men det er kun ganske få der er obligatoriske. De forskellige parametre benyttes f.eks til at:

- navngive Oracle instance'et i forbindelse med opstart
- pege på hvor den vigtige kontrolfil befinder sig. (Det er jo i kontrol filen Oracle har gemt information om datafilernes og redolog'enes fysiske placering.)
- allokere memory plads til SGA'en og delkomponeneterne (data buffer cache, log cache mv.)
- angive datablokstørrelse (gælder kun i forbindelse med `create database` første gang)
- process konfigurationer
- diverse optimeringer
- angivelse af dump direktories
- o.s.v...

De vigtigste er :

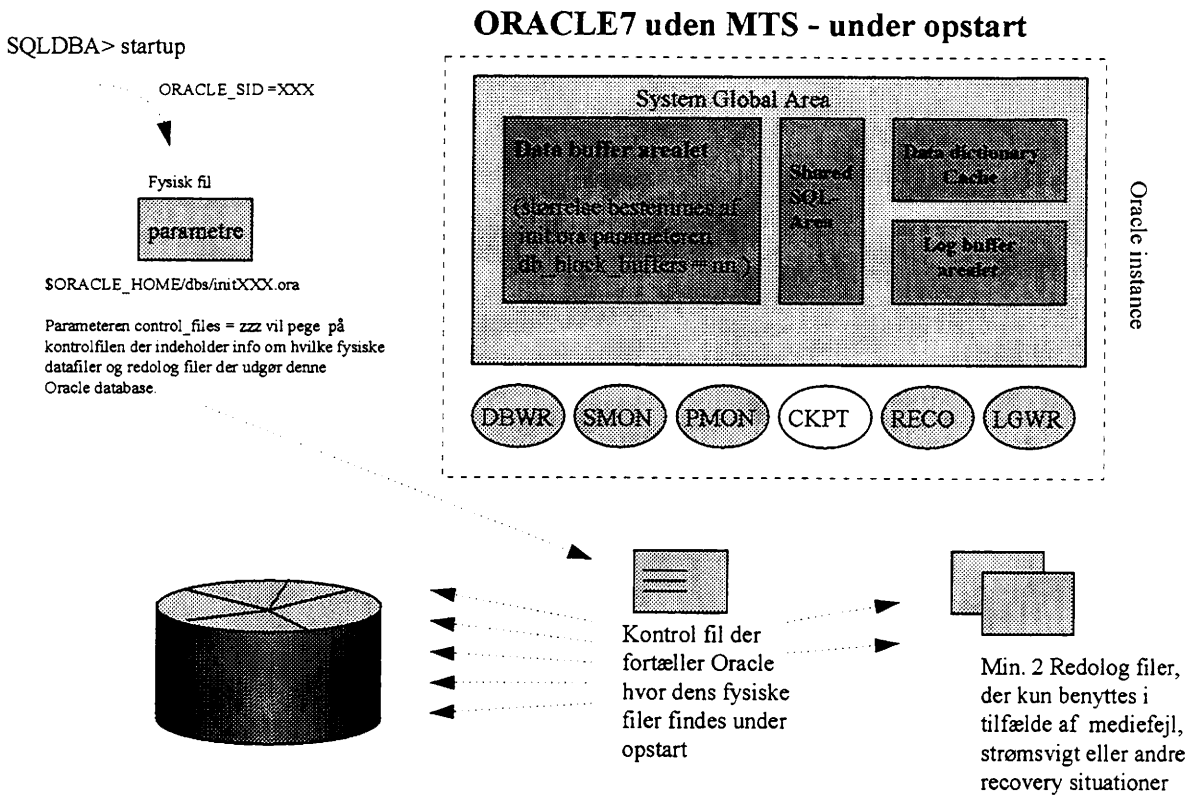
```
control_files = ?/dbs/ctrl_TEST.dbf,/u2/andendisk/ctrl_TEST.dbf
db_name = TEST
db_block_buffers = 200
db_block_size = 2048
log_buffers = 32768
log_checkpoint_interval = 10000
nls_territory=Danish_Denmark.WE8ISO8859P1
nls_sort=True
shared_pool_size = 3000000
```

- men flere parametre vil dukke op undervejs i dette materiale.

Ref.: Oracle7 Administrators Guide app. A

Sådan læses init.ora filen under opstart:

(opstart og nedlukning gennegåes i kapitel 4)



I forbindelse med et kørende insatnce, vil der altid være en fil indeholdende memory adresserne på Oracle's SGA. Denne fil hedder: `$ORACLE_HOME/dbs/sgadef$ORACLE_SID.dbf` og læses af bruger når de logger på en kørende Oracle. På denne måde kan en bruger finde den rigtige Oracle i memory. Se næste side...

Normalt/default checkpoint når redolog-filen er fuld ellers kan det konfigureres i

ORACLE.INI:

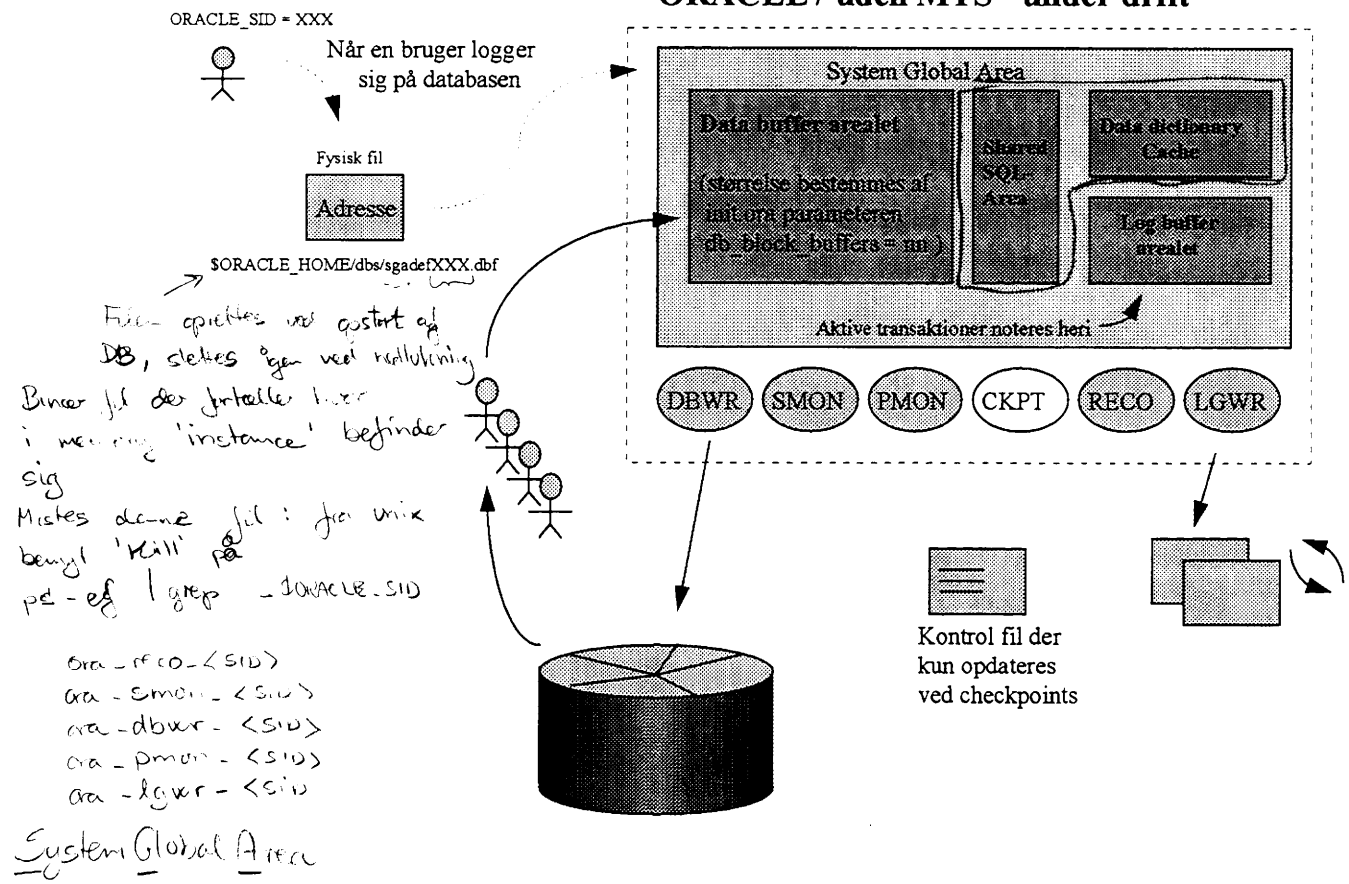
LOG_CHECKPOINT_INTERVAL = 1000
LOG_CHECKPOINT_TIMEOUT = 360

alle 'ALTER Database' kommandoer genererer et checkpoint

Sørg for at have en kopi af 'kontrol fil'

Sådan finder en bruger et bestemt Oracle instance

(Husk brugernes $ORACLE_SID)



Der findes et antal system tabeller i data dictionary som indeholder information om de gennemgåede emner i dette afsnit. Selve data dictionary gennemgåes i næste afsnit, men de vigtigste i relation til dette afsnit er:

Tabelnavn	Beskrivelse	Tabelnavn	Beskrivelse
V$DATABASE	Info om DB navn & status	DICT	Oversigt over data dictionary
V$DBFILE	Navn på datafilene	DBA_DATA_FILES	Navne på datafilerne
V$DATAFILE	Status på datafilerne	DBA_TABLESPACES	Navne på tablespaces
V$LOGFILE	Navn på redolog filerne		
V$LOG	Status på redologs		
V$SGA	Viser SGA allokering		
V$BGPROCESS	Viser Oracle processerne		

Alle tabelnavne der starter med V$... er virtuelle tabeller og ligger kun i memory. Alle tabelnavne der starter med DBA_... er tabeller i data dictionary.

Ref.: Oracle7 Administrators Guide appendix B.

Data Dictionary

Alle væsentlige informationer om databasen selv, dens opbygning og dens egne oversigter over hvad der findes i databasen er gemt i data dictionary, som er en række referencetabeller og views ejet af brugeren **sys**. Tabellerne indeholder oplysninger om fx tabelnavne, kolonnenavne, oprettede brugere, rettigheder, pladsallokeringer samt diverse status- og statistikinformation.

Data dictionary bruges løbende af Oracle til f.eks. optimering af SQL sætninger i forbindelse med vurderinger om et index skal bruges eller ej, at kontrollere en brugers rettigheder på en tabel eller eksistensen af en angivet kolonne o.s.v.

Data dictionary genereres, når databasen oprettes (*det sker når script filerne* `sql.bsq` *og* `catalog.sql` *i forbindelse med* `create database`*...)*. Data dictionary opdateres og vedligeholdes automatisk, når der foretages strukturelle eller allokeringsmæssige ændringer i databasen eller tabellerne. Hvis en tabel oprettes eller modificeres eller når der tildeles eller fratages rettigheder på et segment vil dette skulle noteres i data dictionary

Databaseadministratoren skal derfor kun bruge data dictionary til at søge oplysninger om databasen. Almindelige brugere uden DBA-rettigheder kan også i begrænset omfang bruge data dictionary til at søge oplysninger. Oplysninger i data dictionary fremfindes med almindelig SQL vha. SQL*Plus eller SQL*DBA.

Man kan få en oversigt over indholdet i data dictionary i tabellen `DICT`. Tabellen eller viewet kan benyttes som hjælpeværktøj til at fremfinde de relevante data-dictionary views. Her bliver der søgt efter alle views i data-dictionary, der indeholder ordet "space" i enten TABLE_NAME kolonnen eller COMMENTS kolonnen:

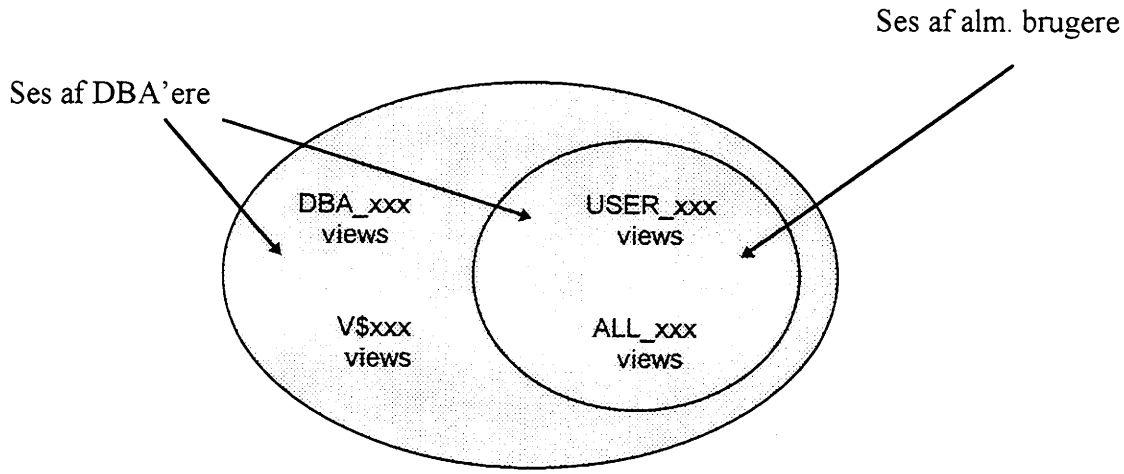
```
SQL> select substr(table_name,1,18) navn, substr(comments,1,60)
  2  from dict
  3  where upper(table_name) like upper('%&ord%')
  4  or upper(comments) like upper('%&ord%');

Enter value for ord: space

TABLE_NAME         SUBSTR(COMMENTS,1,60)
------------------ ------------------------------------------------------------
DBA_AUDIT_OBJECT   Audit trail records for statements concerning obje...
DBA_FREE_SPACE     Free extents in all tablespaces
DBA_TABLESPACES    Description of all tablespaces
DBA_TS_QUOTAS      Tablespace quotas for all users
USER_AUDIT_OBJECT  Audit trail records for statements concerning obje...
USER_FREE_SPACE    Free extents in tablespaces accessible to the user
USER_TABLESPACES   Description of accessible tablespaces
USER_TS_QUOTAS     Tablespace quotas for the user

8 rows selected.
```

Navnestandard i data dictionary

Data dictionary består som beskrevet af nogle basistabeller ejet af brugeren **sys**. Oven på disse tabeller er der bygget et system af **views**, der gør det muligt at hente tabellerne i mere læsbar form for os mennesker. Man bør derfor altid hente sine oplysninger i data dictionary views og ikke i selve tabellerne.

Data dictionary views er således opdelt i fire grupper:

- **USER_xxx:** Kan ses af alle. Her ses information om den enkelte brugers private miljø. F.eks rettighederne på brugerens egne tabeller, eller en oversigt over egne tabeller m.v..
- **ALL_xxx:** Kan ses af alle. Her ses information om andre brugers miljøer. F.eks oversigt over tilgængelige tabeller uanset ejer.
- **DBA_xxx:** Kan kun ses af brugere med DBA rettigheder Her ses information om alt i databasen. F.eks en oversigt over hvilke tabeller en bruger ejer, eller hvilker rettigheder en bruger har givet en anden og på hvad. Her ses også status inforamtion om selve database og dens plads allokeringer.
- **V$_xxx:** Kan kun ses af DBA'ere. Disse views er virtuelle og eksisterer kun i memory. V$xxx tabelleren giver typisk information om memory eller process status, men kan også indeholde information læst fra kontrol filen under opstart.



Her er en liste over de vigtigste views i data dictionary for en DBA'er:

Tabelnavn	Beskrivelse	Tabelnavn	Beskrivelse
V$DATABASE	Info om DB navn & status	DICT	Oversigt over data dictionary
V$DBFILE	Navn på datafilene	DBA_DATA_FILES	Navne på datafilerne
V$DATAFILE	Status på datafilerne	DBA_TABLESPACES	Navne på tablespaces
V$LOGFILE	Navn på redolog filerne	DBA_USERS	Oversigt over alle brugere
V$LOG	Status på redologs	DBA_TABLES	Oversigt over alle tabeller
V$SGA	Viser SGA allokering	DBA_INDEXES	Oversigt over alle index
V$BGPROCESS	Viser Oracle processerne	DBA_VIEWS	Oversigt over alle views
V$PROCESS	Viser alle processer	DBA_OBJECTS	Navne på alle ojekter (tab. ind.)
V$PARAMETER	Viser alle init.ora værdier	DBA_EXTENTS	Allokeringernes adresse og str.
V$ROWCACHE	udnyttelse af dict. cachen	DBA_FREE_SPACE	Adresse og str. på ledig plads
		DBA_SEGMENTS	Sumeret udg. af DBA_EXENTS
V$ROLL STAT		DBA_ROLLBACK_SEGS	Oversigt over alle rollback seg.
		DBA_TS_QUOTAS	Brugernes allokerings kvoter
		DBA_ROLLBACK_SEGS	Oversigt over alle rollback seg.
		DBA_ROLES	Oversigt over alle roller
		DBA_ROLE_PRIVS	Rettigheder tildelt rollerne
		DBA_SYS_PRIVS	System rettigheder tildelt
		ALL_TAB_PRIVS	Rettigheder på tabeller & views

se øvelse 2.

Listen over de vigtigste views i data dictionary for almindelige brugere:

Tabelnavn	Beskrivelse	Tabelnavn	Beskrivelse
ALL_USERS	Liste over alle brugere	USER_USERS	Bruger information om mig
ALL_TABLES	Tilgængelige tabeller	USER_TABLES	Mine egne tabeller
ALL_INDEXES	Tilhørende index	USER_INDEXES	Mine egne index
ALL_VIEWS	Tilgængelige views	USER_VIEWS	Mine egne views
ALL_SYNONYMS	Tilgængeligen synonymer	USER_SYNONYMS	Mine egne synonymer
ALL_SEQUENCES	Tilgængelige tællere	USER_SEQUENCES	Mine egne tællere
ALL_OBJECTS	Overblik over alle tilg. obj.	USER_OBJECTS	Alle mine objekter samlet
ALL_TAB_PRIVS	Rettigheder hvor jeg er med	USER_TAB_PRIVS	Rettigeheder på mine tabeller
ALL_SYS_PRIVS	System rettigheder	USER_SYS_PRIVS	Mine system privilegier
ALL_CATALOG	liste over alt hvad jeg må se	USER_TS_QUOTAS	Mine tablespace kvoter
		USER_TAB_PRIVS_RECD	Rettigheder modtaget
		USER_TAB_PRIVS_MADE	Rettigheder givet fra mig

Ref.: Oracle7 Administrators Guide appendix B.

Eksempler på information hentet fra data dictionary:

Her er et overblik over databasens opdeling, fysisk placering og størrelse, og følgende data dictionary view er et godt udgangspunkt:

```
SQL> select * from dba_data_files;

FILE_NAME
--------------------------------------------------------------------------------
  FILE_ID TABLESPACE_NAME                     BYTES     BLOCKS STATUS
--------- ------------------------------ --------- --------- ---------
/u1/oracle7/dbs/wdbsys.ora
        1 SYSTEM                         10485760      5120 AVAILABLE

/u1/oracle7/dbs/wdbrbs.ora
        3 ROLLBACK_DATA                   3145728      1536 AVAILABLE

/u1/oracle7/dbs/wdbtemp.ora
        4 TEMPORARY_DATA                  2097152      1024 AVAILABLE

/u1/oracle7/dbs/wdbuser.ora
        2 USER_DATA                      20971520     10240 AVAILABLE
```

Som det ses, er denne database opdelt i 4 små tablespaces fordelt på 4 fysiske filer.

Et andet eksempel der giver et hurtigt overblik over hvor meget der er allokeret til forkellige segmenter (f.eks tabeller, index, rollback segmenter mv.). Mere herom i afsnit 5 og 6:

```
SQL> desc dba_segments

 Name                                      Null?    Type
 ----------------------------------------- -------- ----
 OWNER                                              VARCHAR2(30)
 SEGMENT_NAME                                       VARCHAR2(81)
 SEGMENT_TYPE                                       VARCHAR2(17)
 TABLESPACE_NAME                                    VARCHAR2(30)
 HEADER_FILE                                        NUMBER
 HEADER_BLOCK                                       NUMBER
 BYTES                                              NUMBER
 BLOCKS                                             NUMBER
 EXTENTS                                            NUMBER
 INITIAL_EXTENT                                     NUMBER
 NEXT_EXTENT                                        NUMBER
 MIN_EXTENTS                                        NUMBER
 MAX_EXTENTS                                        NUMBER
 PCT_INCREASE                                       NUMBER


SQL> select sum(bytes), sum(blocks) from dba_segments;

 SUM(BYTES)  SUM(BLOCKS)
----------- -----------
    7733248        3776
```

Samme eksempel, blot lidt mere detaljeret:

```
SQL> select tablespace_name, sum(blocks), max(blocks),
  2  max(extents), count(segment_name) "ANTAL_SEG"
  3  from dba_segments
  4  group by tablespace_name;

TABLESPACE_NAME SUM(BLOCKS) MAX(BLOCKS) MAX(EXTENTS) ANTAL_SEG
--------------- ----------- ----------- ------------ ----------
SYSTEM                 3529         640           53        144
ROLLBACK_DATA           200         100            2          2
USER_DATA                47          11            4         18
```

Endnu et eksempel. Her kigger vi på rollback segmenterne:

```
SQL> select segment_name, tablespace_name,
  1         initial_extent,next_extent,pct_increase,status
  2  from dba_rollback_segs;

SEGMENT_NAME TABLESPACE_NAME INITIAL    NEXT PCT_INCREASE STATUS
------------ --------------- ------- ------- ------------ --------
SYSTEM       SYSTEM            51200   51200            0 ONLINE
RB_TEMP      SYSTEM            10240   10240            0 OFFLINE
RB1          ROLLBACK_DATA    102400  102400            0 ONLINE
RB2          ROLLBACK_DATA    102400  102400            0 ONLINE
```

Der vil blive henvist til data dictionary en hel del i resten af materialet med flere eksempler undervejs i de enkelte afsnit.

Startup og shutdown

Startup og shutdown kan **kun** udføres fra **SQLDBA**, og kun af UNIX brugeren "root" eller den UNIX bruger der **ejer** Oracle. Med andre ord er der ingen andre der kan udføre en `connect internal` kommando for at komme i kontakt med Oracle selv om den reelt ikke kører.

Opstart

Et Oracle instance kan være startet i **tre** forskellige niveauer:

- **nomount** — Oracle intance startes uden nogen form for læsning i det fysiske miljø. D.v.s. kontrol filen er lukket, og dermed også data- og redolog filerne. Det er altså kun SGA'en der bliver allokeret i memory og Oracles egne baggrundsrocesser som startes. Benyttes eksempelvis hvis der ikke er en database, og denne skal oprettes med `create database` komandoen.

- **mount** — Oracle intance startes og kontrol filen åbnes og læses. Databasen er altså fortsat lukket, da hverken datafiler eller redologs er blevet åbnet. Benyttes i forbindelse med recovery af en smadret database.

- **open** (default) — Oracle intance startes, kontrol filen åbnes og læses for derefter at åbne både datafiler og redologs. Her vil database automatisk undersøge om databasen er konsistent. Hvis den ikke er konsistent (f.eks. efter et strømsvigt eller anden abort) vil den lave en roll-forward og recovery med udgangspunkt i seneste checkpoint ved at læse frem til den aktuelle og seneste redolog der indeholder information om transaktioner der ikke nåede ned i databasen da det gik galt. Normalt vil databasen ikke skulle foretage en roll-forward / recovery

Til 'open' kan benyttes parameteret 'EXCLUSIVE' hvilket betyder at der kun et instance kan åbne denne ene db

Ovenstående angives som parametre til selve `startup` komandoen. Eksempel:

```
SQLDBA> startup nomount
```

eller

```
SQLDBA> startup mount
```

eller

```
SQLDBA> startup open
```

der er det samme som

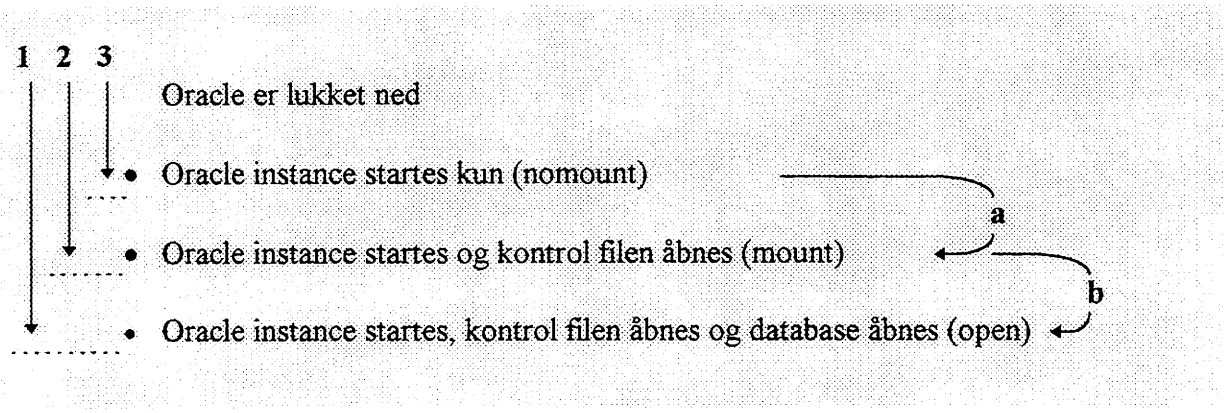
```
SQLDBA> startup
```

PFILE = $ORACLE_HOME/dbs/init$ORACLE_SID.ora

{RESTRICT}
{FORCE}
{nomount | mount | open}

- kun dba'er kan komme på

Opstarts mulighederne kan illustreres således:



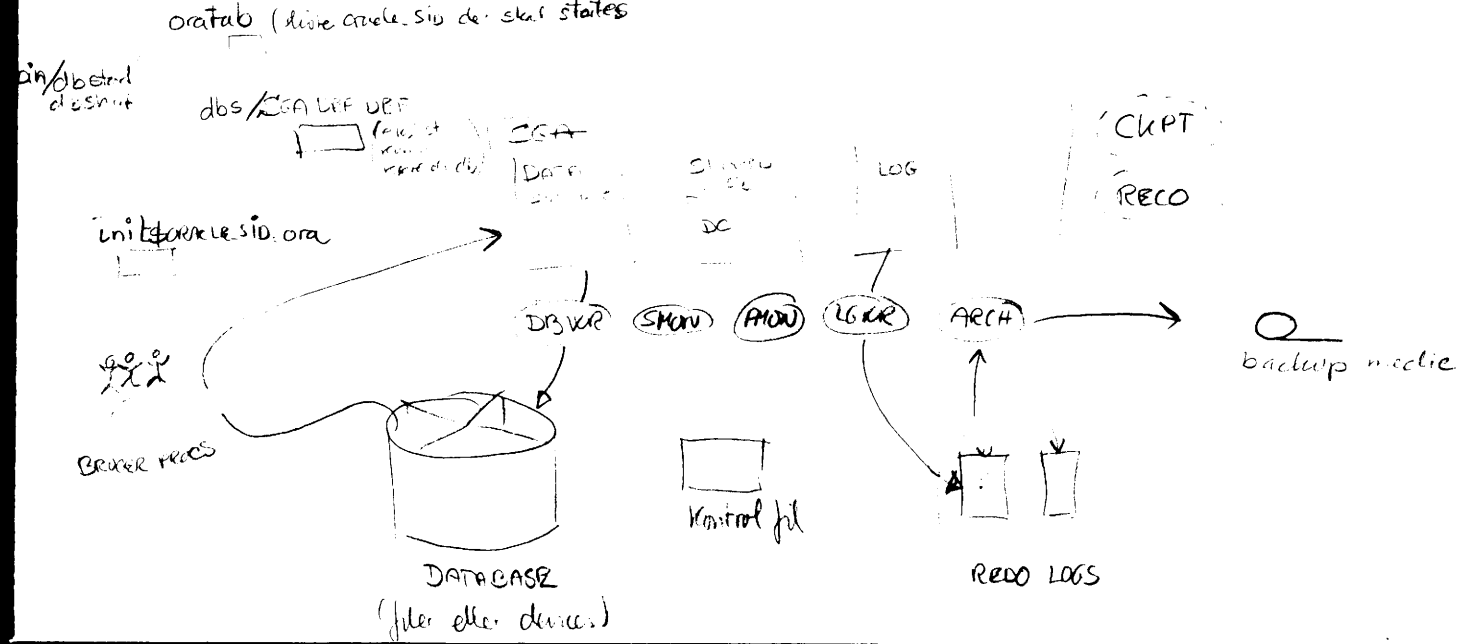
1. `SQLDBA> startup` eller `SQLDBA> startup open` kører lige igennem

2. `startup mount` kører kun igennem til andet nivaeu. Skal man videre herfra og ha' åbnet databasen må følgende komando udstedes:

   b. `SQLDBA> alter database open;`

3. `SQLDBA> startup nomount` starter som nævnt kun Oracle instance. Skal man videre herfra og ha' åbnet databasen må følgende komandoer udstedes i forlængelse heraf:

   a. `SQLDBA> alter database mount;`
   b. `SQLDBA> alter database open;`



Opstarts parametre - automatisk indlæsning

Under opstart indlæses de såkaldte init.ora parametre der angiver Oracle miljøets konfiguration. Denne fil indlæses automatisk hvis den opfylder navngivnings og placeringsstandarden, hvor den skal ligge i et helt bestemt directory og have et navn der relaterer til navnet på et bestemt instance.

Derfor er default formatet for filnavnet til init.ora filen:

```
$ORACLE_HOME/dbs/init$ORACLE_SID.ora
```

- hvilket betyder at Oracle altid vil lede i `./dbs` directory'et under det directory angivet i environment varablen `$ORACLE_HOME` for at finde og læse en fil der hedder `init$ORACLE_SID.ora`, hvis intet andet angives under opstart.

Ekempel:

```
$ORACLE_HOME = /u1/oracle7
$ORACLE_SID  = TEST
```

default parameter filen for TEST databasen er altså:

```
/u1/oracle7/dbs/initTEST.ora
```

Opstarts parametre - manuel indlæsning

Såfremt at man har flere forskallige parameterfiler, der er navngivet efter ens eget ønske, kan man angive dette som argument til `startup` komandoen. Argumentet hedder "... `pfile=`"

Eksempel:

```
SQLDBA> connect internal
SQLDBA> startup open pfile=/usr/oracle7/div_filer/Min_init.ora
```

Nedlukning

Nedlukningen udføres (som opstarten) på en enkelt databbase. Den database der lukkes ned, er det instance med det aktuelle `ORACLE_SID`.

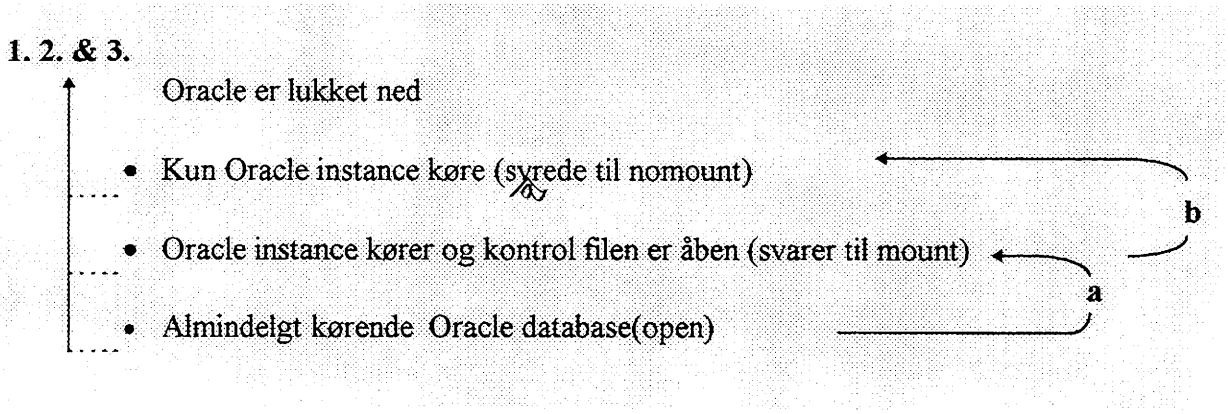
Nedlukning fra SQL*DBA hedder `shutdown` og har flg. argumenter:

- **normal** (default) — Databasen og redologs lukkes efter et sidste checkpoint, kontrolfilen opdateres og lukkes, og Oracle instance lukkes og `sgadef.dbf` filen slettes. En `shutdown normal` venter på at aktive brugere logger af og forhindrer nye brugere i at logge på databasen.

- **immediate** — Er identisk med `normal` argumentet med undtagelse af at der ikke ventes på at aktive brugere logger af. Alle sessioner afbrydes med det samme, men der udføres dog rollback på igangværende transaktioner som ikke var afsluttet. Herefter udstedes sidste checkpoint, Databasen og redologs lukkes, kontrolfilen opdateres og lukkes, og Oracle instance lukkes og `sgadef.dbf` filen slettes.

- **abort** — Svarer til at strømmen går. Dog når den lige at slette `sgadef.dbf` filen. Dvs. at der ventes ikke på at aktive brugere logger af, alle sessioner afbrydes, der udføres ikke rollback på igangværende transaktioner, stopper baggrundsprocesserne men nedlægger ikke SGA'en og udfører ikke et checkpoint. Der skal altså en instance recovery til ved næste opstart. Det sker automatisk.

Illustration af nedlukning:



Uanset der kørende niveau vil `shutdown` kommandoen lukke helt ned:

1. `SQLDBA> shutdown`

eller

2. `SQLDBA> shutdown immediate`

eller

2. `SQLDBA> shutdown abort`

Ønskes nedlukning af databasen trinvis niveau for niveau :

a. `SQLDBA> alter database close`

- Database- og redologdiske er lukket. Basen er tilgængelig for dba'ere til at udføre ændringer. (Feks. at tilføje en redologdisk eller at genskabe hele basen fra en backup.)

b. `SQLDBA> alter database dismount`

- Lukker kontrol filen men instance er stadig kørende men ikke associeret basen.

Herefter udføres en `shutdown` kommando (ref 1. 2. eller 3.) for at lukke helt ned.

Denne er blank med vilje !

Databasens objekter og segmenter

Objekter

Et database objekt er en logiske betegnelse for noget der eksisterer i databasen og ejes af en bruger. Af eksempler på objekt typer kan nævnes:

- Tabeller
- Index
- Clusteres
- Views
- Synonymer
- Sequences
- Database links
- Packages og procedurer

En oversigt over databasens objekter ses med

```
SQL> select * from dba_objects ;
```

Segmenter

Begrebet segment dækker over de objekt typer, der fysisk har optaget og allokeret plads i databasen. Eksempel på segmenter typer:

- Tabeller
- Index
- Clusteres
- Rollback segmenter (kan kun ejes af sys og system)
- Temporære segmenter (allokeres og frigives dynamisk)
- Cache Segmenter (forekommer sjældent)
- Deferred rollback segmenter

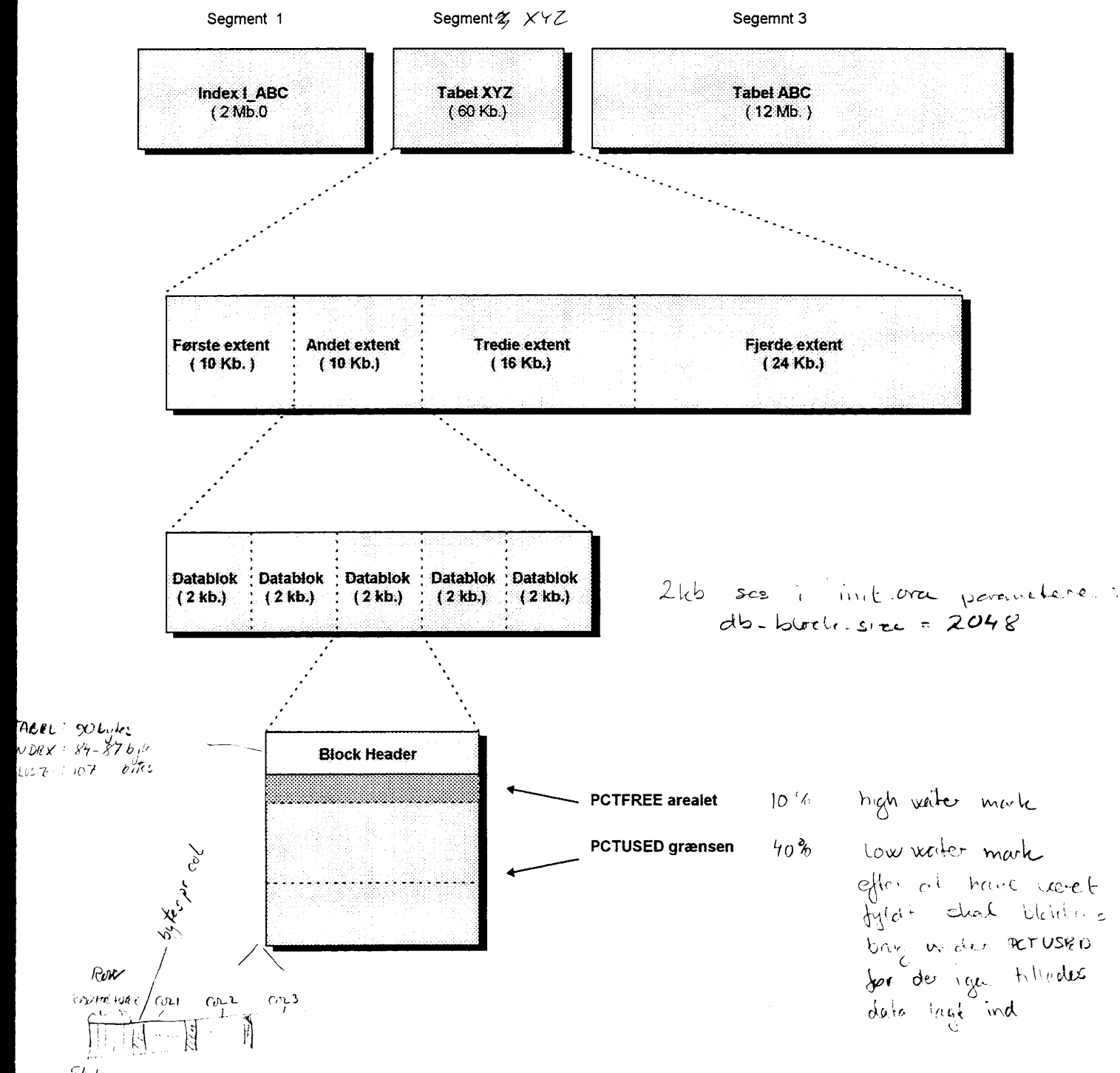
En oversigt over alle segmenter ses med:

```
SQL> select * from dba_segments ;
```

Gennerelt for alle segment typer er allokeringenheden opdelt efter "**extents**" Størrelsen på disse extents varierer fra extent til extent og er direkte afhængig af de **storage parametre** der angives ved oprettelse af de forskellige segmenter. Storage parametre gennemgåes senere i dette afsnit under de enkelte segment typers nærmere beskrivelse.

De enkelte extents allokeres i et antal **datablokke** ad gangen. hvor en datablok typisk er enten 2K eller 4K afhængig af hvad `DB_BLOCK_SIZE` parameteren var sat til i `init.ora` filen under installation og databaseoprettelsestidspunktet.

Segmenters allokering kan illustreres således:



Tabeloprettelse

En tabel kan oprettes med kommandoen `create table` fra SQL*Plus således:

```
SQL> create table ABC (kol1  number(2) not null,
                       kol2  varchar2(10),
                       kol3  date );
```

Tabel-segmentet med tablenavnet ABC vil her blive defineret med de **default værdier** til storage parameterene, der gælder for det tablespace brugeren har som **default tablespace**. se p. 13

Det er muligt ved definitionen at angive præcist hvilke værdier, der skal gælde for tabellen med hensyn til fx størrelsen på og antallet af extents, fyldningsgrad og på hvilket tablespace, tabel-segmentet skal ligge. Eksempel:

```
SQL> create table ABC (kol1  number(2) not null,
                       kol2  varchar2(10),
                       kol3  date )
                      tablespace DATA
                      storage( initial 10K  next 10K  pctincrease 50);
```

Her angives at tabel-segmentet med tablenavnet ABC vil her blive defineret med de storage parameterene **initial** (*første extent størrelse på 10K*), **next** (*efterfølgende extent størrelse ligeledes på 10K*)og **pctincrease** (*der fortæller at extents derefter skal vokse med 50% for hver ny allokering til extents*). Tabellen bliver her oprettet i tablespace DATA.

Den vigtigste og næsten fuldstændige syntaks for **`create table`** er:

```
create table navn (kolonne definitioner,...)
tablespace navn
pctfree n
pctused n
storage ( initial n K|M
          next n K|M
          pctincrease n
          minextents n
          maxextents n ) ;
```

max 255 extents pr. tabel (OS afhængig)
Create user Jesper.
Default tablespace DATA1

create table ABC (
Alter table ABC storage (next 12M);

Forklaring til syntaksen på forrige side:

tablespace angiver det tablespace hvor tabellen skal placeres.

pctfree Størrelse, der angiver hvor mange procent af en datablok, der skal friholdes til fremtidige opdateringer af tabellens data. Standardværdien er 10.

pctused Størrelse, der angiver den minimale fyldingsgrad for en datablok.Falder fyldningsgraden under værdien for pctused, vil blokken blive inkluderet på tabellens friliste, hvorefter der kan indsættes nye tabelrækker i blokken. Standardværdi er 40.

storage Værdier for tabel-segmentets pladsallokering herunder:

- **initial** - Størrelsen i bytes af første extent. standardværdien er 10K.
- **next** - Størrelsen i bytes af andet extent. standardværdien er 10K.
- **pctincrease** - Procentsats der angiver hvor meget efterfølgende extents skalforøges i størrelse pr. gang. standardværdien er 50.
- **minextents** - Antal extents allokeret ved create table tidspunktet. standardværdi er 1.
- **maxextents** - Det maksimale antal **extents**, der kan allokeres til tabellen. standardværdi er 99 med kan variere fra system til system.

Storageparametrenes defaultværdier på et **tablespace** bestemmes ved oprettelsen af det pågældende tablespace med `create tablespace` komandoen. Storage parameterne kan desuden indgå i `create index`, `create cluster` og `create rollback segment` samt de til svarende `alter` kommandoer.

Oplysninger om pladsforbruget for et segment og de enkelte extens kan findes i :

- DBA_SEGMENTS
- DBA_EXTENTS

Oplysninger om tabellens oprettelses værdier findes i :

- DBA_TABLES

Tabel fjernelse

En tabel kan fjernes med kommandoen **drop table**:

```
drop table abc;
```

en **drop table** kan ikke fortrydes med rollback. Vær i øvrigt opmærksom på at der kan være andre objekter der refererer til tabellen i f.eks. "primær/fremede nøgle" relationer, hvorved komandoe vil blive afvist.

Tabel kopiering

En table kan kopieres på to måder:

- Den letteste måde at danne kopier af eksisterende tabeller på, er ved at bruge **create table** kommandoen i én kommando:

```
create table emp30
as select ename navn,job stilling
from emp
where deptno=30;
```

- Kopiering med i to komandoer:
-

```
create table emp30(navn     varchar2(30),
                   stilling varchar2(20)  );

insert into emp30(navn, stilling)
select ename, job
from emp
where deptno=30;
```

Ref.: Oracle7 Server Administrator's Guide, kap. 8. og Oracle7 Server SQL Language Reference Manual, s. 4-207

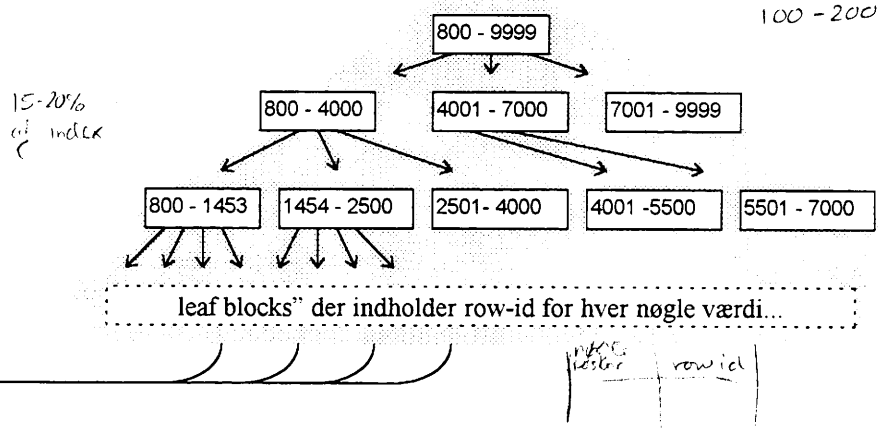
Index oprettelse

Indeks lagres separat fra de tabeller, der indekseres - som redundant information. Indeks kan oprettes og slettes på et vilkårligt tidspunkt uden at påvirke de underliggende tabeller og funktionaliteten af de applikationer, der opererer på disse tabeller. det er altså kun afviklingshastigheden ~~naturligvis~~ der påvirkes.

Oracles indeks er organiseret som b+ træer:

Tabellen post:

Postnr	Bynavn
1051	København K
1208	København K
1264	København K
1851	Frederiksberg C
2000	Frederiksberg
2100	København Ø
2600	Glostrup
2800	Lyngby
...	...
...	o.s.v...



- Nu er en post nummer tabel måske ikke lige det bedste eksempel, da der kun er et par tusinde rækker sådan en tabel. Eksemplet viser et index med meget få (ca. 2-4) braches pr. branch block af hensyn til oversueligheden. Normalt vil der være 50-200 braches pr. branch block. Derved vil antallet niveauer være mindre til en post nummer tabel.

Et indeks oprettes med kommandoen **`create index`** eller **`create unique index`**. Eksempel:

```
create unique index emp_no
on emp(empno);
```

Ovenfor oprettes indekset som unikt. Hvis denne attribut udelades, underforstås ikke-unikt indeks, der accepterer nøgledubletter. Indeks-segmentet vil blive defineret med de default storage værdier, der gælder for det tablespace brugeren har som default-tablespace. Indexet vil ligeledes oprettes i brugerens default tablespace.

$$\text{ANTAL LEAF BLOCKS} = \frac{\text{ANTAL RÆKKER} * (11 + (\text{nøgle længde}))}{(\text{BLOCKSIZE} - 84) * (1 - \frac{\text{PCTFREE}}{100})}$$

$$\text{ANTAL BLOCKS} = \text{ANTAL LEAF BLOCKS} * 1{,}20$$

20%

Den vigtigste og næsten fuldstændige syntaks for **create index** er:

```
create index navn on tabel (kolonnenavn der skal være nøgle)
tablespace navn
pctfree n
pctused n
storage ( initial n K|M
          next n K|M
          pctincrease n
          minextents n
          maxextents n ) ;
```

Parametrene **tablespace, pctfree, pctused** og **Storage** har samme betydning som forklaret under **create table** tidligere i dette afsnit

En brugers indeks kan læses med data dictionary viewet:

- user_indexes.

En oversigt over de kolonner der indgår i et indeks findes i viewet:

- user_ind_columns.

Andre views med oplysninger om indeks:

- all_indexes
- dba_indexes
- all_ind_columns
- dba_ind_columns

Fjernelse af et index

Et indeks slettes med **drop** kommandoen, der tager indeksnavnet som argument:

```
drop index emp_no;
```

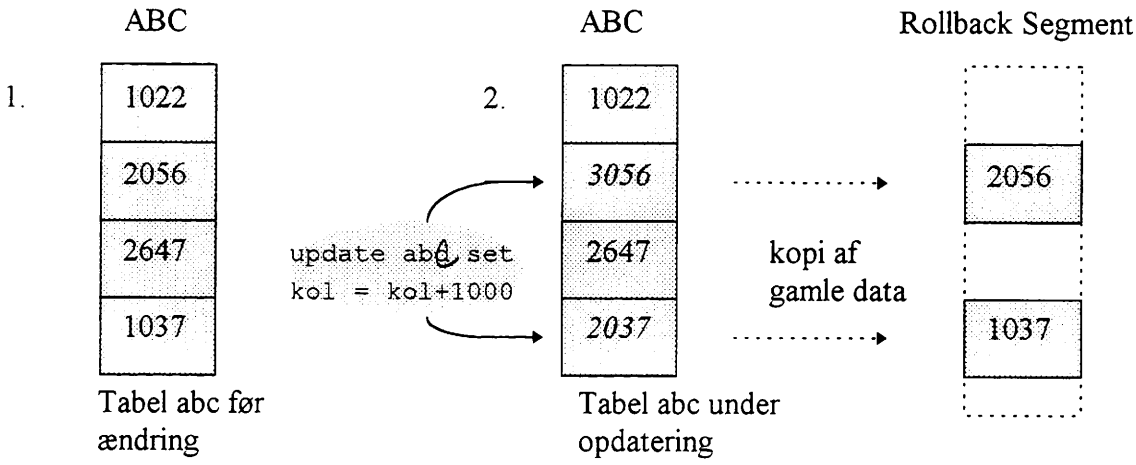
Rollback Segmenterne

En Oracle7 database er fra installaionsens side kun oprettet med et rollback segment (**system**) og et tablespace (**system**). Et rollback segment opsamler information om igangværende transaktioner og opbevarer et billede af data som de så ud inden transaktionssætningerne (`insert`, `update` eller `delete`) manipulerede med dem. En transaktion består af et antal `insert`, `update` og/eller `delete` sætninger,og selve transaktionen afsluttes med enten `commit` eller `rollback`.

Under en opdatering kopieres de gamle data (*before image data*) automatisk ud i et rollback segment for:

- at sikre **læsekonsistens** over for andre brugere der læser de data der er ved at blive ændret, men som endnu ikke er commited endnu.
- at kun føre transaktionen tilbage ved en fortrydelse (**rollback**)
- at **genetablere** databasens indhold i forbindelse med en evt recovery.

Læsekonsistens illustres således:



Storage - parametre: - 'initial' og 'next' skal være lige store (Jack)
- pctincrease vil ønsket blot være sat til 0 (0) dvs. vil være 0pct

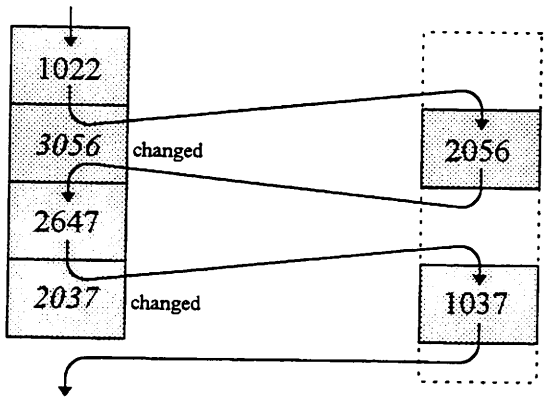
Hele første extents benyttes til header for 'Rollback Seg'
- optimal (IM)

I SQL: "set transaction use.. RS1" i ORA7 kan man for trans bestemme hvilket rollbacksegm. der skal benyttes

Hvis transktionen ikke er *commited* eller *rullet tilbage* endnu vil en anden bruger få et konsistent billede af data således:

Den anden bruger:

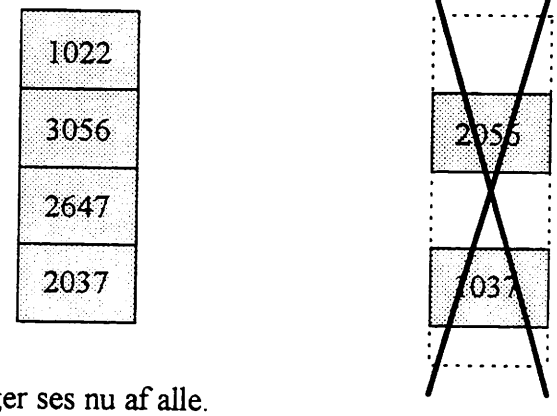
```
select * from abc;
```



Rollback segment

Hvis transaktionen **commit**'es frigives allokeringen i rollback segmentet og tabellens lås fjernes:

```
commit;
```

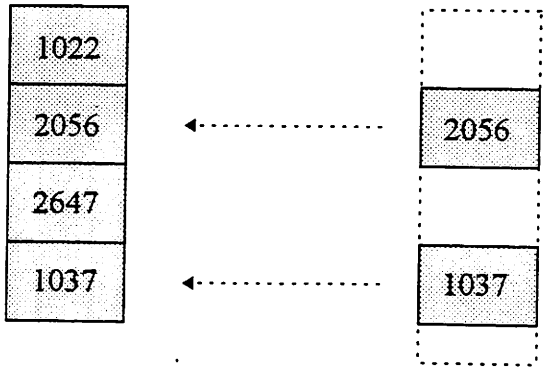


Rollback segment

- og transaktionens ændringer ses nu af alle.

Hvis transaktionen afsluttes med **rollback** kopieres de gamle data tilbage fra rollback segmentet og tabellens lås fjernes:

```
rollback;
```



Rollback segment

- og alt er som før transaktionen gik igang.

Rollback Segmenternes allokering

Et rollback segment allokerer plads efter behov, og vokser efter samme princip som en table eller index. Den forrige version af Oracle; Oracle v6, kunne rollback segmenter ikke frigive den plads de fik allokeret. Det er blevet rettet i Oracle7 med introduktionen af en ny storage parameter som hedder **optimal.** Denne storage parameter angiver en optimal størrelse som rollback segmentet skal krybe tilbage til efter en stor transaktion.

optimal bør ikke være på Rollbacksegmentet 'SYSTEM'

Et rollback segment oprettes eksempelvis således:

```
create rollback segment rs1
tablespace ts_rbs
storage ( initial 100K
          next 100K
          pctincrease 0
          minextents 2
          maxextents 99
          optimal 5M );

alter rollback segment online;
```

Performance

Jo flere transaktioner der benytter et rollback segnment jo større er sandsynligheden for at rollback segmentets header blocks bliver en flaskehals i systemet. Derfor er det en god ide at oprette et pænt antal rollback segmenter i forfhold til den forventede transaktions mængde.

Det anbefales at man maximalt har 4-5 samtidige transaktioner pr. rollback segment. Hvis man forventer at have maximalt 30 samtidige transaktioner igang vil det altså være fornuftigt at oprette 6 rollback segmenter. Med disse 6 rollback segmenter vil der maximalt være 30 transaktioner i gang hvis man sætter følgende parametre i `init.ora` filen:

```
transaktions = 30                                  -  begrænsning af transaktioner
transaktions_per_rollback_segment = 5              -  max. trans pr rollback seg.
rollback_segments = (rbs1, rbs2, rbs3, rbs4, rbs5, rbs6) - så sættes disse online
                                                          ved opstart.
```

- hermed undgår kan at benytte SYSTEM rollback segment som ikke kan reorganiseres.

Anbefalet: (for real OLTP miljø er 100kbytes nok)
Initial + next extent skal være minimum 500kbytes

Temporære segmenter

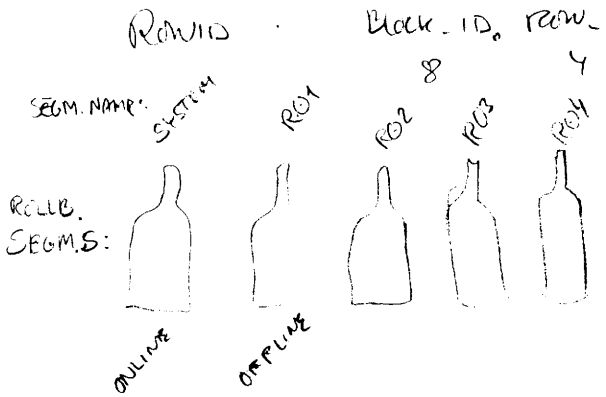
Temporære segmenter er væsentligt lettere at have med at gøre. De allokerer og frigiver plads dynamisk. Temporære segmenter benyttes når Oracle ikke selv kan håndtere større sorteringer og datamanipulationer i memory. Temporære segmenter benyttes i forbindelse med større sorteringer ved:

- `create index ...`
- `select distinct ...`
- `select ... order by`
- `select ... group by`
- `select ... union ... select ...`
- `select ... intersect ... select ...`
- `select ... minus ... select ...`
- joins
- synkroniserede subquerys

Ønsker man at allokere mere memory til sorteringer gøres dette i **`init.ora`** filen med parameteren:

```
SORT_AREA_SIZE = n
```

Temporære segmenter allokeres altid i brugerens **default temporary tablespace**. Se afsnit 7 under bruger oprettelse.



ad: p.36

$$DATA = \text{ANTAL RÆKKER} * \left(5 + \sum_{i=1}^{n} \text{col.lgth.}_i + 1\right)$$

Variabel ved VARCHAR2
Kendt ved CHAR

$$\text{ANTAL BLOCKS} = \frac{DATA}{(\text{Block Size} - \underbrace{90}_{\text{block header}}) * \left(1 - \frac{\text{PCTFREE}}{100}\right)}$$

Undersøg tabellen "DEPT":

DATA:

```
select sum(vsize(deptno)+ vsize(dname)+1 + vsize(loc)+1 + 5) from dept
```

=> 24832

Block size - 90 = 4096 - 90 = 4006 * (1 - 10/100) = * 0,9

Denne side er blank med vilje !

```
select ceil( sum( ..... )/4006 *0,9 ) from dept
```

=> 7 blokke

ceil: round up

brugt i DEPT

```
... rowid, ... from scott.dept where rownum < 10;
00000134. xxxx. xxxx
select distinct substr(rowid, 1, 8) from scott.dept
```

=>
00000134
00000135
00000136
00000137
32B
22C
32D

```
select substr(rowid, 1, 8), count(*) from scott.dept
group by substr(rowid, 1, 8)
```

134	14
135	14
136	14
137	14
32B	14
32C	14
32D	13

↑ Antal rækker pr block

6. Space management

Tablespaces

I forlængelse af de forrige afsnit vil vi her kort gennemgå oprettelsen af diverse tablespaces. Et tablespace oprettes med kommandoen:

```
create tablespace navn
datafile 'filnavn el. subdisk' size n
default storage ( initial n
                  next n
                  pctincrease n
                  minextents n
                  maxextents n);
```

Afsnit 5 gennemgik vi oprettelsen af diverse segment typer (tabeller, index, rollback segmenter m.v.), hvor der blev refereret til **default storage værdierne** hvis ikke andet blev angivet. Der er netop de værdier der sættes i forbindelse med ovenstående **`create`**- eller **`alter tablespace`** kommando.

De to views i data dictionary som indeholder relevant information om databasens tablespaces hvad angår pladsallokering og storage parametre er:

- DBA_DATA_FILES
- DBA_TABLESPACES

Hvis man ønsker sig et overblik over hvilke default storage parametre der er sat på de enkelte tablespaces er følgende et forslag:

```
SQL> select * from dba_tablespaces;

Tablespace_name Initial    Next Min_ext Max_ext Pct_incr Status
--------------- ------- ------- ------- ------- -------- -------
SYSTEM            10240   10240       1     254       50 ONLINE
DATA1             10240   10240       1     254       50 ONLINE
DATA2             10240   10240       1     254       50 ONLINE
TEMP              51200   51200       1      99      100 ONLINE
RBS              102400  102400       2     254        0 ONLINE
```

Her listes alle status- samt default storage værdier pr. tablespace.

Det er altid en god ide at få et overblik over databasens opdeling, fysisk placering og størrelse. Her er følgende view et godt udgangspunkt:

```
SQL> select * from dba_data_files;

FILE_NAME
------------------------------------------------------------------------
  FILE_ID TABLESPACE_NAME                 BYTES     BLOCKS STATUS
--------- ------------------------------ --------- --------- ---------
/u1/oracle/dbs/wdbsys.ora
        1 SYSTEM                       10485760      5120 AVAILABLE

/u1/oracle/dbs/wdbrbs.ora
        3 RBS                           3145728      1536 AVAILABLE

/u1/oracle/dbs/wdbtemp.ora
        4 TEMP                          2097152      1024 AVAILABLE

/u1/oracle/dbs/wdbuser.ora
        2 DATA1                        20971520     10240 AVAILABLE

/u1/oracle/dbs/wdbuser.ora
        5 DATA2                        40971520     20480 AVAILABLE
```

Som det ses, er denne database opdelt i 5 små tablespaces fordelt på 5 fysiske filer

Hvor meget af denne database er allerede brugt (allokeret til forkellige segmenter f.eks tabeller, index, rollback segmenter mv.) og hvor meget ledig plads er der tilbage?

Følgende giver et hurtigt overblik over hvor meget der er allokeret totalt set:

```
SQL> select sum(bytes), sum(blocks) from dba_segments;

 SUM(BYTES)  SUM(BLOCKS)
-----------  -----------
    7733248         3776
```

Samme eksempel lidt mere detaljeret:

```
SQL> select tablespace_name, sum(blocks), max(blocks), max(extents),
  2  count(segment_name) "ANTAL_SEG"
  3  from dba_segments
  4  group by tablespace_name;

TABLESPACE_NAME                SUM(BLOCKS) MAX(BLOCKS) MAX(EXTENTS) ANTAL_SEG
------------------------------ ----------- ----------- ------------ ---------
SYSTEM                                3529         640           53       144
RBS                                    200         100            2         2
DATA1                                   47          11            4        18
o.s.v...
```

- man kunne evt. tilføje en "where sætning", der undgår data dictionary tabellerne:

```
SQL> select tablespace_name, sum(blocks), max(blocks), max(extents),
  2  count(segment_name) "ANTAL_SEG"
  3  from dba_segments
  4  where owner <> 'SYS'
  5  group by tablespace_name;

TABLESPACE_NAME                SUM(BLOCKS) MAX(BLOCKS) MAX(EXTENTS) ANTAL_SEG
------------------------------ ----------- ----------- ------------ ---------
SYSTEM                                 142           8            4         9
DATA1                                   47          11            4        18
o.s.v...
```

Er der mange tabeller eller index der er fragmenteret ? Her tæller vi hvor mange segmenter, der har mere end 10 extents (udvidelser):

```
SQL> select count(*) from dba_segments
  2  where extents > 10
  3  and owner <> 'SYS';

 COUNT(*)
---------
        2
```

- lad os finde ud af hvilke:

```
SQL> select segment_name, owner from dba_segments
  2  where extents > 10;
  3  and owner <> 'SYS';

SEGMENT_NAME   OWNER
-------------  -----------
EMP3           SCOTT
KUNDER         JENS
```

Vi undersøger EMP3 tabellen lidt nærmere og benytter nu DBA_EXTENTS viewet:

```
SQL> desc dba_extents
 Name                            Null?    Type
 ------------------------------- -------- ----
 OWNER                                    VARCHAR2(30)
 SEGMENT_NAME                             VARCHAR2(81)
 SEGMENT_TYPE                             VARCHAR2(17)
 TABLESPACE_NAME                          VARCHAR2(30)
 EXTENT_ID                       NOT NULL NUMBER
 FILE_ID                         NOT NULL NUMBER
 BLOCK_ID                        NOT NULL NUMBER
 BYTES                                    NUMBER
 BLOCKS                          NOT NULL NUMBER
```

Beregningskolonnen "END" er indsat for at kunne se, om de allokerede extents ligger i forlængelse af hinanden ved at sammenligne denne værdi med næste rækkes BLOCK_ID:

```
SQL> select substr(segment_name,1,15) navn, extent_id, file_id
  2         block_id, blocks, block_id+blocks "END"
  3  from dba_extents
  4  where segment_name = upper('&navn')
  5  and owner = upper('&ejer')
  6  order by file_id, block_id;

Enter value for navn: emp3
Enter value for ejer: scott

NAVN            EXTENT_ID   FILE_ID  BLOCK_ID     BLOCKS        END
--------------- --------- --------- --------- ---------- ---------
EMP3                    0         2        38          2        40
EMP3                    1         2        40          2        42
EMP3                    2         2       129          3       132
EMP3                    3         2       132          4       136
EMP3                    4         2       136          6       142
EMP3                    5         2       142          9       151
EMP3                    6         2       204         14       218
EMP3                    7         2       218         21       239
EMP3                    8         2       239         32       271
EMP3                    9         2       271         48       319
EMP3                   10         2       319         72       391
```

Vi ser at EMP3 tabellen har 11 extents (udvidelsesallokeringer). Alle extents ligger pænt ved siden af hinanden med undtagelse af extent 1 & 2 og 5 & 6. Vi kan også se at denne tabels PCTINCREASE må ha' været sat til 50%.

Læser default 8 blokke frem ad gangen (ora.ini:)

På samme måde kan ledig plads undersøges. Her kan der forekomme free space arealer, der ligger i forlængelse af hinanden uden at disse bliver samlet. Dette kaldes "honeycomb fragmentering". Fra version 7.0.16 og frem kan Oracle selv finde ud af at samle flere free space arealer som et.

Vi ser i DBA_FREE_SPACE viewet:

```
SQL> desc dba_free_space

Name                            Null?    Type
 ------------------------------ -------- ----
 TABLESPACE_NAME                 NOT NULL VARCHAR2(30)
 FILE_ID                         NOT NULL NUMBER
 BLOCK_ID                        NOT NULL NUMBER
 BYTES                                    NUMBER
 BLOCKS                          NOT NULL NUMBER
```

```
SQL> select tablespace_name, file_id, block_id,
  2  blocks, block_id+blocks "END"
  3  from dba_free_space
  4  order by tablespace_name, file_id, block_id;

TABLESPACE_NAME                 FILE_ID  BLOCK_ID    BLOCKS       END
------------------------------ --------- --------- --------- ---------
RBS                                   3       202      1335      1537
SYSTEM                                1      3531      1590      5121
TEMP                                  4         2        10        12
TEMP                                  4        12      1013      1025
DATA1                                 2        49         5        54
DATA1                                 2        54         5        59
DATA1                                 2        59         8        67
DATA1                                 2       308       100       408
DATA1                                 2       577        20       597
DATA1                                 2       597        30       627
DATA1                                 2       627       910      1537
DATA1                                 2      2178      8063     10241
o.s.v...
```

Der er altså den såkaldte "honney comb" fragmentering i denne database - dvs eksemplet stammer fra en en version tidligere end version 7.0.16. (*for eksemplets skyld*)

Til sidst kigger vi på rollback segmenterne:

```
SQL> select segment_name, tablespace_name,
  1         initial_extent,next_extent,pct_increase,status
  2  from dba_rollback_segs;

SEGMENT_NAME TABLESPACE_NAME INITIAL_EXTENT NEXT_EXTENT PCT_INCR STATUS
------------ --------------- -------------- ----------- -------- --------
SYSTEM       SYSTEM                  51200       51200        0 ONLINE
RB_TEMP      SYSTEM                  10240       10240        0 OFFLINE
RB1          RBS                    102400      102400        0 ONLINE
RB2          RBS                    102400      102400        0 ONLINE
```

Parameteren PCTINCREASE skal altid være 0 på rollback segmenter

7. Brugere, privilegier og sikkerhed

For at kunne logge på databasen, kræves et **Oracle-bruger-id** med tilhørende password. Brugeradministration på en Oracle database omfatter følgende arbejdsopgaver:

- at oprette brugere i databasen
- at tildele/ændre passwords
- at tildele og vedligeholde systemprivilegier for brugere
- at tildele og vedligeholde rettigheder på tabeller
- at tildele default tablespaces og temporary tablespaces til brugere
- at tildele eventuelle allokerings kvoter i tablespaces

Brugeroprettelse

En bruger på databasen oprettes med kommandoen **`create user`**. Her er den simple syntaks:

```
CREATE USER navn IDENTIFIED BY password ;
```

-og den fulde syntaks:

```
CREATE USER navn IDENTIFIED BY password
DEFAULT TABLESPACE ts_navn
TEMPORARY TABLESPACE ts_navn
QUOTA n ON ts_navn ;          <-----   (n kan også være ordet "unlinited")
```

Her er et eksempel: Opret en bruger WINSTON med password CHRUCHILL:

```
CREATE USER winston IDENTIFIED BY churchill
DEFAULT TABLESPACE data2
TEMPORARY TABLESPACE temp
QUOTA 5M ON data2 QUOTA unlimited ON temp;
```

Vær intet 'quota' er default 0 bytes

Systemprivilegier

Efter oprettelse af en bruger med kommandoen `CREATE USER`, har den nye bruger ingen systemprivilegier. Systemprivilegier giver rettigheder til udførelse af forskellige kommando-typer. I Oracle7 findes ca. 80 system-privilegier. Disse kan ses i Oracle7 Server SQL Language Reference Manual under "`GRANT`". Det er sjældent at man i en almindelig installation har behov for at udspecificere systemprivilegier den detaljeringsgrad som Oracle7 giver mulighed for. Derfor er der nogle default roller (**`roles`** der gennemgåes på næste side), som hver især dækker over et antal systemprivilegier.

Både roller og systemprivilegier tildeles med kommandoen **`GRANT`**:

Blot et eksempel tre systemprivilegier en bruger skal have som minimum. Her tildeles brugeren WINSTON rettigheder til at:

- logge på databasen
- oprette egne tabeller
- oprette synonymer

```
SQL> GRANT CREATE SESION, CREATE TABLE, CREATE SYNONYM TO winston;
```

Der eksisterer 80 rettigheder

For at gøre tingene nemmere er der som nævnt lavet 3 roles der dækker et normalt behov for at kunne skælne imellem bruger typer. Rollerne hedder:

- CONNECT - indeholder alle de systemprivilegier som en almindelig slutbruger har brug for.
- RESOURCE - ~~indeholder CONNECT plus~~ det som en udvikler har brug for. F.eks retten til at eje og oprette ~~tabeller og index.~~
  - Indeholder automatisk ubegrænset QUOTA på alle tablespace og procedure og trigger
- DBA - indeholder næste alle rettigheder.

Der har stået i manual de sidste 4år at disse 3 roller vil forsvinde!

Brugeren kan læse sine privilegier i viewene

- USER_SYS_PRIVS
- USER_ROLE_PRIVS
- ROLE_SYS_PRIVS

'create session" rettighed bør være tilstrækkelig for den almindelige bruger

```
select * from role_sys_privs where role in ('CONNECT','RESOURCE');
```

```
Select * from role_sys_privs where role = 'CONNECT'
```

ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE
CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW

Der findes kun disse 3 i ORA6 mens der i 0817 findes de 80

dde

Roles

Roles er navngivne sæt af privilegier. Roles gør det nemmere og mere overskueligt at administrere de mange system-privilegier. Brugen af roles begrænser det samlede antal grants betydeligt.

En role kan også indeholde objekt-privilegier (omtales efterfølgende).

En role oprettes med kommandoen **CREATE ROLE**.

Eksempel:

```
CREATE ROLE journal;
```

Objekt-privilegier

En oversigt over objekt-privilegier med de objekttyper de omfatter kan se således ud:

ALTER	table, sequence
EXECUTE	procedure
DELETE	table, view
INDEX	table, cluster
INSERT	table, view
REFERENCES	table
SELECT	table, view, sequence, snapshut
UPDATE	table, view
ALL	

- Foreign Key

Eksempel: Giv brugeren abc privilegier til at søge og indsætte i tabellen adresser.

```
GRANT select,insert ON adresser TO abc;
```

desc dba_ts_quotas ;

TABLESPACE_NAME
USERNAME
BYTES
MAX_BYTES = QUOTA
BLOCKS
MAX_BLOCKS

PRIVLEGE : 'Grant any privilege' virker ikke !

Fratage privilegier

Privilegier fratages med kommandoen REVOKE.

Eksempel: Fjern rettighederne ALTER og DELETE på tabellen budget fra rollen regnskab.

```
REVOKE alter,delete ON budget FROM regnskab;
```

Eksempel: Fjern retten til at oprette tabeller fra brugeren abc.

```
REVOKE create table FROM abc;
```

REFERENCE:
Oracle7 Server Administrator's Guide, kap. 12.
Oracle7 Server SQL Language Reference Manual s. 4-225.

Oracle utilities vil, hvis der ikke er angivet oracle-bruger, forsøge med
OPS$<OS-bruger-ID>/OPS$<OS-bruger-ID>

8. Databaseoprettelse

Her er et eksempel på en manuel database oprettelse. Databasen kommer til at hedde S, og scriptet skal køres fra SQLDBA:

```
set termout on
set echo on
spool /usr/oracle7/dump/opret_S.log

connect internal

startup nomount pfile=/usr/oracle7/dbs/initS.oprl

create database S controlfile reuse
  logfile  '/dev/redolog1S'  size 10238K reuse,
           '/dev/redolog2S'  size 10238K reuse
  datafile '/dev/db_system_S' size 25592K reuse;

connect system/manager

create rollback segment rs0;
alter rollback segment rs0 online;

create tablespace RBS
  datafile '/dev/db_rbs_S' size 51192K reuse
  default storage (initial 100k
                   next 100k
                   pctincrease 0
                   minextents 2
                   maxextents 200
                   optimal 1M);

create tablespace TEMP
  datafile '/dev/db_temp_S' size 104440K reuse
  default storage (initial 20k next 40k pctincrease 100);

create tablespace DATA1
  datafile '/dev/db_data1_S' size 563192K reuse
  default storage (initial 10k next 10k pctincrease 50);

create tablespace DATA2
  datafile '/dev/db_data2_S' size 255992K reuse
  default storage (initial 10k next 10k pctincrease 50);

create tablespace IND
  datafile '/dev/db_ind_S' size 204792K reuse
  default storage (initial 10k next 10k pctincrease 50);

create rollback segment rs1 tablespace rbs;
create rollback segment rs2 tablespace rbs;
create rollback segment rs3 tablespace rbs;
create rollback segment rs4 tablespace rbs;
create rollback segment rs5 tablespace rbs;
```

fortsættes....

```
fortsat...
connect internal
shutdown

startup

connect system/manager

drop rollback segment rs0;

create user scott identified by tiger;

grant connect, resource to scott;        <----- benytter roles i stedet for privs.

alter user sys temporary tablespace temp;
alter user system temporary tablespace temp;
alter user scott default tablespace data1 temporary tablespace temp;

spool off
```

I ovennævnt oprettelsesscript bliver der refereret til 2 init.ora filer under de forskellige startup;

`/usr/oracle7/dbs/initS.opr1` (benyttes kun en gang ved create database delen)
`/usr/oracle7/dbs/initS.ora` (som er default og ligger korrektplaceret med rette navn)

De enkelte init.ora filer ser således ud:

`/usr/oracle7/dbs/initS.opr1`

```
#
# init.ora Copyr (c) 1987-1996 Oracle. Benyttes kun af opret_S.sql og
# bruges kun iforb. med create database...
#
control_files = ?/dbs/ctrl_S.dbf,/etc/save.d/ora6control/ctrl_S.dbf

db_name = S
db_block_buffers = 200
db_block_size = 2048
shared_pool_size = 3000000

init_sql_files = (?/dbs/sql.bsq,\
?/rdbms/admin/catalog.sql,\
?/rdbms/admin/expvew.sql)

language=Danish_Denmark.WE8ISO8859P1
nls_sort=true

user_dump_dest = /usr/oracle7/dump
background_dump_dest = /usr/oracle7/dump
```

/usr/oracle7/dbs/initS.ora (default)

```
#
# init.ora Copyr (c) 1987-1996 Oracle
# Dette er default init.ora til S databasen.
#
control_files = ?/dbs/ctrl_S.dbf,/etc/save.d/ora8control/ctrl_S.dbf

db_name = S
db_file_multiblock_read_count = 8
db_block_buffers = 1500
db_block_write_batch = 5
db_block_size=2048
db_files=30
shared_pool_size=3000000

ddl_locks = 2300

# Context area:
open_cursors = 300
context_area = 16384
context_incr = 8192

# Enable audit faciliteten:
audit_trail = TRUE

# NLS:
language=Danish_Denmark.WE8ISO8859P1
nls_sort=true

# Redo logfiles:
log_buffer = 256000
log_allocation = 10000
log_checkpoint_interval = 10000
log_files=10

processes = 100

# Rollback segment opsaetning, saa system IKKE bruges:
rollback_segments = (rs1, rs2, rs3, rs4, rs5)
transactions_per_rollback_segment = 5
transactions = 25

row_cache_cursors = 25
row_cache_enqueues = 500
row_cache_buffer_size = 400

sequence_cache_entries = 50
sort_area_size = 131072

background_dump_dest = /usr/oracle7/dump
user_dump_dest = /usr/oracle7/dump
```

Spejlet (2 stk) →

default
Læser 8 blokke hver gang den første læses (readahead) →

→

Kan ikke ændres/sættes undtagen ved den startdos hvorved DB er oprettet

Default 100
Dette er meget/for højt !

Overholder ikke → DK registerlov

for hver 10000 os → blokke checkpoint

default 1,1 × processes → transactions

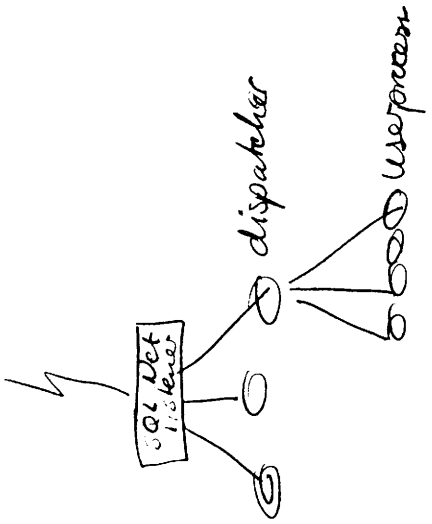
Sætte er meget lille → gerne ex. 1Mbytes

Husk at alle fremtidige brugere, der ikke skal eje ~~tabeller~~ procedure i databasen, bør oprettes således:

```
create user xxx identified by yyy;
grant connect to xxx;
alter user xxx temporary tablespace temp;
```

-og alle fremtidige brugere, der også skal eje ~~tabeller~~ procedure i databasen:

```
create user xxx identified by yyy;
grant connect, resource to xxx;
alter user xxx default tablespace ts_navn
      temporary tablespace temp_ts
      quota unlimited on ts_navn to xxx;
```



Ved multithreaded server vil dispatcher kunne knytte forskellige brugere på skift til samme userprocess

9. Backup, restore og recovery

Der er tre fundamentalt forskellige måder, hvorpå man kan tage backup af en Oracle database.

1. ORACLE-programmerne **exp/imp** - Logisk backup
2. UNIX-programmet **dskback** - Fysisk offline backup
3. Archiving - Fysisk online backup (hot)

Det er naturligvis muligt at vælge en kombination af ovenstående. Hvilken backupstrategi der vælges til den enkelte installation vil afhænge af flere forhold, bl.a.

- Backupmediets størrelse (streamer/video)
- Antal opdateringer (backuphyppighed)
- Behovet for at kunne 'fiske' data, fx enkelte tabeller
- Behovet for en genetablering af basen, der genopretter data helt op til tidspunktet for nedbrud (archiving)

Backupstrategien bør fastlægges i samarbejde med leverandør.

Logiske backup (exp og imp)

Oracle tilbyder to programmer, **exp** og **imp**, som kan bruges til hhv. eksport af data fra databasen og import af data til databasen. **Exp**- programmet lægger kopien af de udvalgte tabeller i en UNIX-fil, hvorfra den kan lægges tilbage igen med **imp**-programmet. **exp/imp**-programmerne har følgende egenskaber:

- Det er muligt at tage kopi af:
  - enkelte tabeller
  - alle tabeller, views, synonymer m.m. for enkelte brugere
  - hele databasen

- Oracle 6.0 og senere versioner, giver mulighed for tilvækst backup.En tilvækstbackup tager kun kopi af de tabeller, der har ændret sig siden sidste backup.

- Exp- og imp-programmet afvikles på en kørende database. Inkonsistens kan opstå, hvis der samtidig med exp-programmet kører forgrenede transaktioner på basen, eftersom programmet eksporterer én tabel ad gangen.

- Imp giver mulighed for, at data udlægges sekventielt. Dette medfører en bedre performance (se senere under oprydning).

- En eksport fylder mindre end en diskkopiering, da data lagres på komprimeret form. Fx eksporteres indeksdefinitioner i stedet for de faktiske indeks.

- Eksporten kan lægges direkte ud på et backupmedie.

Nedenfor et eksempel på de spørgsmål exp-programmet stiller under en interaktiv afvikling af programmet:

```
Enter array fetch buffer size: 80000 >
Bufferstr. ved overførsel til UNIX-fil

Export file: expdat.dmp >
UNIX filnavn til eksportfil

E(ntire), U(sers), or T(ables): U >
Hele databasen, alle en brugers objekter eller ud valgte tabeller

Export Grants (Y/N) N >
Eksportere rettigheder

Export table data (Y/N): Y >
Eksportere tabeldata eller kun definitioner

Compress extents (Y/N): Y >
komprimere extents i ét initial extent
```

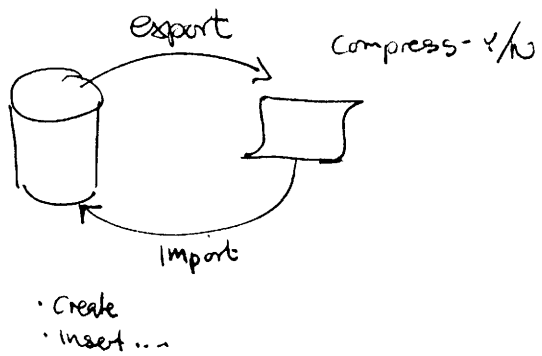
Programmet vil herefter løbende meddele hvad der sker under eksporten.

Alternativt kan man angive værdier for de forskellige nøgleord i én kommandolinie. Denne metode er velegnet til fx shellscripts.

Eksempel: Eksportér scott's objekter inkl. grants

```
# exp system/password FILE=scott.dmp GRANTS=y OWNER=scott
```

Archiving består i at redologdiskenes indhold kopieres ud på et andet medie, fx en streamer. Da redologdiskene logger alle ændringer der sker på basen, er en kopi af redologdiskene et godt værktøj til at genetablere basen efter et nedbrud. Archiving kræver dog en del administration til gengæld for sine fortræffeligheder.



Fysisk offline backup (i UNIX)

UNIX sikkerhedskopiering af de rå subdiske kan ske med **dskback**-programmet. Sikkerhedskopiering af rå diske er hurtigere end eksport, men kræver en større indsigt i diskenes organisering og Oracles opbygning. En diskbackup bruger mere plads end en export.

- En total backup af databasen taget med dskback skal omfatte:
  - databasedisken(e)
  - redologdiskene
  - en kopi af kontrolfilerne

Eksempel: Diskbackup til videodrev + backup af kontrolfiler

```
SQLDBA> shutdown

# dskback -B -c -v      /dev/databaseDE
                        /dev/user1DE
                        /dev/redolog1DE
                        /dev/redolog2DE
                        /dev/redolog3DE
                        /dev/video

# tar cvf /dev/flop     /usr/ora7/dbs/control1DE.dbf
                        /etc/save.d/oracontrol/control1DE.dbf
                        /etc/save.d/oracontrol/control2DE.dbf
```

cpio
arpy (NT)
orchserv (NT)

Online backup (hot archiving)

Archiving består i at redologdiskenes indhold kopieres ud på et andet medie, fx en streamer. Da redologdiskene logger alle ændringer der sker på basen, er en kopi af redologdiskene et godt værktøj til at genetablere basen efter et nedbrud. Archiving kræver dog en del administration til gengæld for sine fortræffelighed.

Forudsætningen for at kunne køre med archiving er følgende:

- at databasen er sat i "log archive mode" (ALTER DA... archivelog) (init.ora: log_archive_start=true)
- at ARCH processen kører
- at destinationen for ARCH er sat (init.ora:

Når dette er sat, tages der backup af et tablespace ad gangen ved at sætte det i en slags "offline" status for alle skrivninger (men ikke for læsninger) for derefter at kopiere tablespacetes datafil(er) ud på tape . Når kopieringen er afsluttet sættes det pågældende tablespace "online" igen for skrivninger. Denne process gentages for hvert tablespace.

Der skal iøvrigt være flere tablspaces i databasen. Se eksemplet på nste side...

Eksempel:

I init.ora filen tilføjes:

```
log_archive_dest = /u2/backup_disk
log_archive_start = true
log_archive_format = 'log%s_%t.arc'
```

Databasen lukkes ned til første niveau:

```
SQLDBA> alter database close;

SQLDBA> alter database dismount;
```

herefter sættes databassen i rette mode:

```
SQLDBA> alter database archivelog;

SQLDBA> alter database open;
```

og databasen er klar til at kunne håndtere en online /hot backup startegi. Tag I øvrigt en total backup af database miljøet hurtigst muligt herefter.

Databasens redologs bliver nu løbende arkiveret, og der kan nu tages backup af databasen periodisk, f.eks en gang om ugen. En database backup tages således:

```
SQL> alter tablespace ts_navn begin backup;
```

- så tages en fysisk kopi af tablespacets datafil(er)...

```
SQL> alter tablespace ts_navn end backup;
```

Disse tre trin gentages for hvert tablespace i databasen.

Fejl og recovery

Der skelnes mellem forskellige typer af fejl, der kan opstå under afvikling af programmer. Hvilken form for genetablering der anvendes, afhænger naturligt nok af fejlsituationen.

Brugerfejl
Fx at en bruger utilsigtet sletter tabeller eller tabelrækker.Data kan retableres fra en eksport.

SQL fejl
Fx at en bruger forsøger at afvikle en SQL-sætning, der fejler. Oracle annullerer automatisk de ændringer, der er en følge af SQL-sætningen. Er der tale om en manglende recource, kan den DBA-ansvarlige udvide ressourcen.

Procesfejl
En bruger- eller en skyggeproces står af. Oracles baggrundsproces PMON opdager den forsvundne proces og ruller alle processens uafsluttede transaktioner tilbage. Alle ressourcer processen har brugt frigives.

Instance fejl
Oracle databasen går ned p.g.a. strømsvigt, system crash eller fejl i Oracle RDBMS. Et almindeligt symptom er, at en af baggrundsprocesserne terminerer. Ved opstart af databasen laves der automatisk Instance Recovery. Se beskrivelse af Instance Recovery i næste afsnit.

Mediefejl
En disk der bruges til Oracle ødelægges, fx ved head crash. Den DBA-ansvarlige må først genetablere den ødelagte disk, og dernæst genetablere databasen.

Recovery

1) Rolling forward
- verificering af at alle kommittede transaktioner er skrevet til databasediske
- verificering af at alle ændrede data på såvel data
- som rollback-segmenter er skrevet til databasedisken
- ændringer der burde være på databasedisken overføres

2) Rolling back
- verificering af at transaktioner der er fortrudt med rollback ikke findes på databasedisken
- verificering af at transaktioner hvorpå der ikke er udført commit ikke findes på databasedisken
- ændringer der ikke burde være på databasedisken fjernes

Da alle transaktioner siden sidste **checkpoint** står i den aktuelle **online redolog**, kan hele genetableringen af databasen ske automatisk.

Er der tale om en termineret baggrundsproces, må den databaseansvarlige lukke basen basen ned og starte den op igen for at aktivere de automatiske recovery procedurer.

Recovery efter mediefejl

Hvis der er en hård fejl på en eller flere af diskene, er der under Oracle 6.0 grundlæggende 3 forskellige måder at håndtere den nødvendige recovery. Måderne vil naturligvis afspejle, hvorledes backuppen er taget.

1. Indlæsning af en konsistent rå diskbackup, taget af alle diske, mens basen var lukket ned - med tilhørende kontrolfiler. Databasen bliver ført tilbage til backuptidspunktet.

2. Import af en fuld databaseeksport. Databasen skal kreeres, før importen kan foretages. Databasen vil blive ført tilbage til eksporttidspunktet. Denne metode kræver at man er i stand til at rekonstruere eventuelle udvidelser til databasen foretaget siden den oprindeligt blev defineret.

3. Indlæsning af en rå diskbackup af alle diske med tilhørende kontrolfiler samt pårulning af transaktioner fra de arkivere de redolog diske. Metoden kræver, at redolog diskene er blevet arkiveret med archiving faciliteten. Databasen vil blive ført op til det tidspunkt, da databasen gik ned.

Ref.: Oracle7 Server Administrators's Guide, kap. 19.

10. Overvågning

Monitor-programmet (under SQL*DBA) kan vise forskellige informationer om et kørende Oracle-system, bl.a.:

- Hvilke tabeller, der er i anvendelse.
- Hvilke brugere, der er aktive.
- Antallet af fysiske/logiske læse/skrive-operationer for de enkelte processer.
- Informationer om rollback segmenterne.
- Hvilke låse processerne har sat
- Hvor mange aktive transaktioner er i gang

Monitor-programmet kaldes fra SQL*DBA således:

```
> connect system/<password>
```

[CTRL][g] eller **F4** og **F5** på en vt220 terminal skifter til menuen
Vælg Monitor

De menupunkter der har aktuel interesse er:

System I/O
Viser den procentvise I/O-aktivitet på de enkelte processer inkl. baggrundsprocesserne. Der gives både et øjebliksbillede indenfor hvert opdateringsinterval, samt et akkumuleret billede fra start af databasen.

Process
Viser alle baggrunds- og bruger-processer der kører på databasen.
For hver proces vises:
- ORACLE proces id
- UNIX proces id
- UNIX brugernavn
- Om der ventes på en LATCH
- Terminalen processen udføres på
- ORACLE program der udfører processen

Table
Viser hvilke tabeller der accesseres.
For hver tabel vises:
- Session id
- Tabelejer
- Tabelnavn

File I/O
Viser hvorledes antallet af logiske læsninger er I forhold de fysiske. Dette kaldes "Hit ratio" og værdien skal helst være over 0.75 - 0.80

Rollback Segments
Viser status på rollback segmenterne, hvor mange active transaktioner der er i gang på hvilke rollback segmenter

Session
Viser informationer om hver bruger-session.
For hver bruger vises fx:

- Session id
- Proces id
- ORACLE brugenavn
- Sidst udførte SQL-statement

Ref.: Oracle7 Server Utilities User's Guide, kap. 13.

11. Særlige Oracle7 features

Constraints

Constraints er nogle integritetsregler, der kan defineres i tabellerne i forbindelse med tabeloprettelsen. Sådanne regler benyttes f.eks til at lægge noget consistenscheck og forretningslogik over på database serveren fremfor I applikationen. Dette er for at skabe en bedre balance I mellem applikation og database f.eks. i client/server miljøer.

Af constarints kan nævnes:

- Primær/fremmedenøgle relationer imellem tabeller
- kolonne checks (f.eks en kolonnes værdi må ikke overskride en bestemt værdi)
- Entydighed I en kolonne (Unique)
- Default værdier
- Not null som vi alle kender

Ved hjælp af constraints er det også muligt at åbne op for andre værktøjer og applikationer, da man herved har sikret sammenhængen i relationer og integritet i databasen.

Procedurale objekter i databasen til afvikling I database kernen

Der er mulighed for at afvikle procedurer (både hændelsesafhængige og uafhængige) samt funktioner som man selv kan definere med PL/SQL - et proceduralt sprog omkring SQL. Objekterne kan således deles op:

- Stored procedures (skal ekseveres explicit)
- Triggers (hændelsesafhængige procedurer tilknyttet tabeller)
- Packages (et sæt af sammenhørende procedurer)
- Funktions (procedurer der returnerer en værdi)

Det er her man for alvor kan hente performance i et client/server miljø.

Andre facilliteter i Oracle7

- **MTS** Multi threaded Server konfigurationer, dergøt det muligt at dele en skyggeprocess (der i denne sammenhæng hedder en server process) mellem flere brugere. Herved er der mulighed for at spare memory, men der skal være over 20 samtidige brugere på databasen for at kunne mærke en forbedret performance.

- **Distribuerede databaser** - Oracle har en option i Oracle7 der gør det muligt at køre på flere database fra samme applikation. Hvis der er tale om læsninger kan det klares med de traditionelle database link definitioner. Er der tale om distribuerede transaktioner benyttes en "two-phase commit" algoritme på godt og ondt. Algoritmen sikrer at der enten committes på alle involverede databaser eller også rulles der tilbage på alle involverede databaser. Med andre ord; ingen distribueret transaktion vil opleve en commit på en database samtidigt med en rollback på en anden database.

12. Appendix

Ofte stillede spørgsmål

Sp: ***Hvordan ser man om en Oracle database kører?***
Sv: Man følger denne fremgangsmåde:

1. Gå ud i UNIX.
2. Indtast følgende kommando: `ps -ef ^ grep smon ^ wc -l`
3. Systemet kvitterer med en af følgende værdier:
   - Hvis 1 betyder det, at basen ikke kører.
   - Hvis 2 (eller højere) betyder det, at basen kører.

Sp: ***Hvordan ser man hvilke brugere, der er logget på databasen?***
Sv: Man kan bl.a. se dette gennem en forespørgsel i Data Dictionary.
Dette gøres således:

1. Gå i SQL*Plus som DBA (system/manager)
2. Indtast nu denne kommando efter ledeteksten
3. `SQL> select distinct username from sys.v_$session;`

Sp: ***Hvad er et tablespace?***
Sv: Et tablespace er en administrativ lagerenhed i en Oracle database.
Et tablespace kan bestå af en eller flere filer og/eller rå diske. En Oracle database vil altid have minimum et tablespace, der kaldes SYSTEM. På mange Oracle 6 databaser, installeret af DDE, findes der kun det ene tablespace kaldet SYSTEM. Hvis dette er tilfældet vil såvel systemdata, som brugerdata ligge i dette tablespace. Det vil i givet fald være dette tablespace, der skal overvåges mht. til pladsforbrug.

På de fleste Oracle 7 databaser vil der foruden tablespace SYSTEM også være andre tablespaces. Hvis der er andre tablespaces, vil der som regel være et eller flere tablespaces, der forbeholdt brugerdata. Dette/disse hedder ofte sådan noget som USERS og USERS1 ol.; men kan i princippet hedde hvad som helst.

Sp: ***Hvordan ses, hvor meget plads der i alt er i basen?***
Sv: Man kan se dette gennem en forespørgsel i Data Dictionary.
Dette gøres således:

1. Gå i SQL*Plus som DBA (system/manager)
2. Indtast nu denne kommando efter ledeteksten:
3. ```
   SQL> select tablespace_name, sum(blocks), sum(bytes)
        from sys.dba_extents
        group by tablespace_name;
   ```
4. Der vises nu den totale plads i blokke og bytes for hvert tablespace.

Sp: ***Hvordan ses, hvor meget fri plads, der i alt er i basen?***
Sv: Man kan se dette gennem en forespørgsel i Data Dictionary.
Dette gøres således:

1. Gå i SQL*Plus som DBA (system/manager)
2. Indtast nu denne kommando efter ledeteksten:
3. ```
   SQL> select tablespace_name, sum(blocks), sum(bytes)
        from sys.dba_free_space
        group by tablespace_name;
   ```
4. Der vises nu den totale friplads i blokke og bytes for hvert tablespace.

Sp: ***Hvordan finder man ud af, hvor meget plads, der er forbrugt i databasen.***
Sv: Dette er meget enkelt: Man tager de tal, man har fundet for det totale pladsforbrug for hvert tablespace, og herfra trækker man de tal, man har fundet for den totale friplads på hvert tablespace.

Sp: ***Hvad vil det sige, at en database er fragmenteret?***
Sv: At en database er fragmenteret betyder, at data ikke ligger samlet og dermed udnytter pladsen optimalt. Dette kan dog give visse administrative og performance-mæssige udfordringer til DBA'en

Sp: ***Hvordan kan man se, hvor fragmenteret basens friarealer er?***
Sv: Man kan se dette gennem en forespørgsel i Data Dictionary.
Dette gøres således:

1. Gå i SQL*Plus som DBA (system/manager)
2. Indtast nu denne kommando efter ledeteksten:
3.
```
SQL> select tablespace_name, count(*)
     from sys.dba_free_space
     group by tablespace_name;
```
4. Der vises nu det totale antal frie områder for hvert tablespace.

Sp: ***Hvor store skal fragmenteringen være før der skal gøres noget?***
Sv: Det afhænger bl. a. hvor stor lagermediet er og hvor meget databasen benyttes.
Mange DDE kunder foretager jævnligt en såkaldt reorganisering af databasen, fx. en gang årligt eller halvårligt.

Egne noter:

Øvelser

1. Introduktion (for at se os lidt omkring i Oracle miljøet)

1. Log ind på systemet med det unix brugernavn, instruktøren opgiver.
   - Kald **sqlplus** som dba brugeren **system/manager**
   - Udfør forespørgslen:

```
select username from dba_users;
```

   Du skulle meget gerne få en liste over alle oprettede brugere i databasen...

2. For at se forskellen i mellem en dba-bruger (f.eks system/manager) og en almindelig bruger:

   Log på eller "connect" som en almindelig bruger (navn opgives af instruktøren). Foretag den samme forespørgsel som i delopgave 1. Hvad er resultatet? (du skulle meget gerne få en fejl der fortæller at tabellen ikke findes)

3. Log på eller "connect" som **system/manager** igen...
   - Hvor mange tabeller og views ejer brugeren **sys** ?
   - (se i følgende datadictionary views: **dba_tables, dba_views,** og **dba_objects**)

select count(*) from dba_objects where owner = 'SYS' and (object_type = 'VIEW' or object_type = 'TABLE') => 394

owner table_name

owner view_name

object_type view table

64

330

4. Log ud af sqlplus igen.

5. Log ind i sqldba.

6. Afgiv kommandoen **connect system/manager**.
   - Hvad er resultatet ?
   - udfør delopgave 1 her i SQLDBA

7. Udfør kommandoen **connect internal.**
   - Hvad er resultatet ? (her skulle du blive afvist af sikkerhedmæssige hensyn)

└ insufficient privileges

Egne noter:

set long 10000
select text from dba_views where view_na = 'DBA_VIEWS';

select table_na from dba_tables where owner = 'SYS'
minus
select object_name from dba_objects where owner = 'SYS' and object_type = 'TABLE';
⇒ -default_auditing_options
Denne tabel kan ikke 'desc'!

select distinct unique object_type from dba_objects group by object_type;

CLUSTER
INDEX
PACKAGE
PACKAGE BODY
PROCEDURE
SEQUENCE
SYNONYM
TABLE
VIEW

Opgaver

2. Oracle7 arkuitektur

1. Hvad er indeholdt i et "instance" ? SGA og Oracle baggrunds processer
2. Hvornår og hvorledes navngives et "instance" ? db_name i oracle.ini
3. Hvad er indeholdt i SGA'en ? Data buffer area (Shared SQL area + Data dictionary) Shared P
   log buffer area
4. Med 3 instances på en maskine - Hvorledes finder en bruger frem til det rigtige ?
   ORACLE_SID
5. Baggrundsprocessen ARCH er valgfri. - Hvorfor og hvad benyttes denne til ?
   Archiver redolog - indhold til er tape - backup
6. Hvilken Oracle baggrundsprocess rydder op hvis en bruger smides af systemet af en eller anden grund ?
   PMON
7. Hvad er det mindste antal redologs en Oracle7 skal have ?
   2
8. Hvad kunne være et argument for at have flere redologs end minimum ?
   1. ARCH kører ikke arkiverer hurtigt nok til at den kan tømme filen
9. Hvad er det mindste antal kontrol filer i en Oracle 7 installation ?
   1
10. Hvad kunne være et argument for at have flere kontrol filer end minimum ?
    Spejling giver større sikkerhed
11. Hvor mange datafiler skal minimum anvendes hvis;
    - der er et tablespace (SYSTEM) ? 1
    - der er 5 tablespaces ? 5
12. Hvor mange rollback segmenter skal der minimum være hvis;
    - der er et tablespace (SYSTEM) ? 1
    - der er 5 tablespaces ? 2
    Ved flere tablespaces min 1+1 !
13. Kan der være flere tablespaces i en datafile eller sub/rådisk ? Nej
14. Hvilken information lagres i kontrol filen ? checkpoint
    redolog
    filnavne
15. Hvad er data dictionary ?
    Databasens egen database
16. Hvilke(t) data dictionary view(s) giver overblik over databasen fysiske og logiske struktur ?
    DBA_TABLESPACES

Egne noter:

- LGWR ikke behov for overskrivn
2. Ved stort antal kan ARCH undlades fordi redo-log'er
ligger for ex en hel uge, hvor der så er en backup af db

Øvelser

2. Oracle7 arkitektur

1. Gå ud i UNIX igen

2. Hvordan er dine **ORACLE_HOME** og **ORACLE_SID** variable sat?
   /usr/ora7 MTS

3. Log ind i **sqlplus** som dba-bruger; **system/manager**.
   - det gik vel fint ?

4. Afslut sqlplus

5. Ret din **ORACLE_SID** variabel til **xyz**.
   - Log ind i **sqlplus** som **system/manager**
   - Hvad sker nu ? - gik det godt ?
   ERROR: ORA-01034: ORACLE not available
   ORA-07429: smsgsg: shmget() failed to get segment

6. Ret dit 'Oracle-miljø' tilbage til de oprindelige værdier.
   - Log på **sqlplus** som **system/manager** igen.

7. Hvor mange tablespaces er der i denne database ?
   select * fra dba tablespaces => 5

8. På hvilke fysiske filer eller subdiske ligger databasen ?
   select * from V$dbfile =>

9. Hvor stor er databasen totalt ?
   Sum af segmenter select sum(bytes) from V$datafile; => 80146432

10. Hvilke filer eller subdiske er afsat til redologs ?
    select * from V$logfile →

11. Hvor mange rollback segmenter er oprettet her i ?
    Select * from DBA_ROLLBACK_SEGS; 5 rows

12. Hvor ligger databasens kontrolfiler ? (findes i den relevante init.ora fil)

13. Hvor stor er SGA'en p.t. ?
    select * from V$SGA;
    sum(value)

14. Kører ARCH processen p.t. ?
    select * from V$BG PROCESS
    where name='ARCH'

DBA_TABLESPACES:
- NAME
- INITIAL_EXTENT
- NEXT_EXTENT
- STATUS

segment_name	tablespace_name
SYSTEM	SYSTEM
R01	RBS
R02	RBS
R03+4	RBS

Fixed Size 38086
Variable Size 4042316
Database Buffers
Redo Buffers
4915936

PADDR = 20018BCC
NAME = ARCH
DESCR = Archival

ps -ef | grep -MTS => bla ora_arch_MTS

Egne noter:

SYSTEM
RBS
TEMP
TOOLS
USERS

SUPERMAX ORA-01119 V Error · 2 : No such file or directory

1 /dev/mtssys
2 /dev/mtsrbs
3 /usr/ora7/dbs/mtstmp
4 /usr/ora7/dbs/mtstools
5 — /mtsusers
6 /db7test/browse20-tbl
7 /usr/ora7/dbs/mtstmp2

1 /dev/mtsr1
2 /dev/mtsr2
3 /dev/mtsr3

Opgaver

3. Data dictionary

1. I hvilket tablespace findes data dictionary ? SYSTEM

2. Data dictionary er opdelt i 4 hovedgrupper
   - Hvilke ?
   - Hvordan identificeres de enklet views fra de enkelte grupper ?
   - Hvad er forskellen i millem de enkelte grupper
   - Hvad er helt specielt for V$xxx tabellerne ?

   USER_ ...
   ALL_ ...
   DBA_ ...
   V$_ ...

   ? eksisterer kun i memory

3. Kan en almindelig bruger se alle 4 grupper ? Nej

4. Hvordan opdateres data dictionary ?
   Automatisk når der foretages strukturelle eller allokeringsmæssige ændringer i DB eller tabeller

Øvelser

3. Data dictionary

1. Log på **sqlplus** med **system/manager** hvis den er blevet logget af siden sidst.

2. Hvilke tabeller ejers af brugeren 'SCOTT' ?
   select table_name from DBA_TABLES where OWNER='SCOTT'

3. Hvor meget af databasen er allokeret til index i forhold tabeller ?
   select segment_type, sum(bytes), sum(blocks) from dba_segment group by segment_type

Typisk er det 1/3

4. Er der tabeller med mere end 3 extents ?
   - Hvis det er tilfældet, hvem ejer disse ? → SYS, ORA3, TEST, ORA5, ORA1, ORA2
   select segment_name, extents, owner from dba_segments where extents > 3 and segment_type = 'TABLE';

5. Hvor mange objekter findes der i databasen totalt ud over data dictionry ?
   select count(object_name) from dba_objects where owner != 'SYS' => 70 (Ejes af SYS)

6. Hvor mange brugere er oprettet i databasen ?
   select count(*) from dba_users → 54

7. Hvor mange rollback segmenter er allokeret i databasen ?
   select count(*) from dba_segments where segment_type = 'ROLLBACK';

8. Er alle rollback segmenter i brug ?
   select distinct status from dba_rollback_segs group by status; => ONLINE
   JA !

fortsættes.....

Egne noter:

desc dba_extents

owner
segment_name
segment_type — (CACHE, CLUSTER, INDEX, ROLLBACK, TABLE)
tablespace_name
extent_id — not null
file_id — " "
block_id — " "
bytes
blocks — ~~not~~ null

```
select segment_name, owner, count(*), from dba_extents
where segment_type = 'TABLE'
group by segment_name, owner
having count(*) > 3
order by 3 desc;
```

=> 13 rows

BONUS
DEPARTMENT
DEPT
DUMMY
EMP
JOB
LOCATION
SALGRADE

	sum(bytes)	sum(blocks)	
CACHE			
CLUSTER			
INDEX	3207168	783	1
ROLLBACK		3188	.
TABLE	13058048		4

desc dba_rollback_segs

SEGMENT_NAME
OWNER
tablespace_name
segment_id
file_id

status

3. Data dictionary (*fortsat*)

9. Hvor mange DBA brugere findes der ?
   select * from dba_role_privs where granted_role = 'DBA' =>
10. Hvor mange v$tabeller findes der ?
    select count(*) from DICT where table_name like 'V$%' => 61 (62)
11. Udskriv en liste der viser hvilke typer af segmenter, der findes i basen.
    select distinct segment_type from dba_segments group by segment_type
12. Prøv at finde cache-segmentet, rollback-segmentet og eventuelle temporary-segmenter.
    select * from DBA_... where segment_type = 'CACHE'
13. Hvor mange extents er der i databasen i forhold til segmenter ? et segm ~ 2 extents
    select count(*), sum(extents) from dba_... => 521 761
    select count(distinct segment_name), count(*)
    from dba_extents
    — Temporary segments
    — Deferred rollback segments

dba_sgm.sql

/usr/ora7 /rdbms/admin/catdbsgm.sql :

create synonym DBA... for SYS.DBA...

Egne noter:

desc dba_role_privs

GRANTEE
GRANTED_ROLE
ADMIN_OPTIONS
DEFAULT_ROLE

SYS
DBA, CONNECT, RESOURCE, PMLRULE
NO/YES
NO/YES

		ADM	DEF
SMM	DBA	NO	YES
STEEN	"	NO	YES
SYS	"	YES	YES
SYSTEM	"	YES	YES

CACHE
CLUSTER
INDEX
ROLLBACK
TABLE

Hvilke rettigheder er tilknyttet en rolle:

desc role_sys_privs

role
privilege
admin_options

CLUSTER:

Opgaver

4. Startup og Shutdown

1. Hvornår læses init.ora filen ? under opstart
2. Hvordan finder Oracle frem til den rigtige init.ora hvis man ikke specifikt angiver dette ?
   Benytter env $ORACLE_SID og $ORACLE_HOME $ORACLE_HOME/dbs/init$ORACLE_SID.ora
3. Hvordan angiver man i øvrigt at man ønsker at starte op på basis af en særlig init.ora fil ?
   parameter pfile = < >
4. Hvornår bruges connect internal og hvorfor ? for startup, den startup kræver at man er 'connected'. Linien SQL> viser om der er forbindelse til en database eller ej. Den kan kun udføres når
5. Hvad er forskellen mellem **startup mount** og **startup nomount** ?
   kontrolfil er åbnet men ikke database, ellers ikke startet
6. Hvad er den hurtigste genvej til at lukke databasen uden at skulle recover ved næste opstart ? shutdown immediate
7. Oracle kan være kørende i tre niveauer. Forklar hvad de enkelte niveauer benyttes til ?
8. Hvad skal der gøres på en kørende Oracle hvis **sgadefxxx.dbf** filen slettes ?
   kill 8mon

Egne noter:

Opgaver

5. Databasens objekter/segmenter og pladsallokering

1. Der er et maximum på 60 samtidige transaktioner på en database.
   - Hvor mange rollback segmenter vil du oprette ? 12
   - Udfyld følgende init.ora parametre i denne forbindelse:

   transactions = 60
   transactions_per_rollback_segment = 5
   rollback_segments = (rbs1, rbs2, rbs3, rbs4, ..., rbs12)

2. Hvad benyttes de temporære segmenter til ? Ved større sorteringer/datamanipulationer som der ikke er plads til i memory
   - Hvem opretter disse ? Oracle DB selv
   - Hvad sker der med et temporært segment når det ikke skal bruges længere

3. Her kommer der en sej opgave:
   Under oprettelse af et index på en tabel der er 500 Mb stor, hvor vi har estimeret at indexet kommer til at fylde ca. 200 Mb angives denne kommando:

```
create unique index I_stor_emp on stor_emp(empno)
tablespace data2
storage (initial 200 M next 20 M pctincrease 50)
```

   ... og et stykke tid efter får man følgende fejl på trods af at storage parametrene er korrekte:

```
ORA-01556 Maximum number of (99) extents reached
          cannot create object.
```

   ora-01630 / I unix shell: oerr ora 01630
   - Hvad er galt og hvad gør man ???

4. Hvad skal pctincrease være ved oprettelse af rollback segmenter ?
   0 pct. Ønsker heraf må selv den til, kender ☺
5. Vil en ændring i storage-parametrene have en effekt på allerede allokerede extents i en tabel ? Nej

6. Hvad vil du sætte storage parametrene til, hvis du skal oprette et unikt index på en tabel med ca. 10.000 rækker, og hvor nøglen er ca. 7 bytes lang ? p 40
   (PS: databasens blokstørelse er 2 K)

Unix shell: bc (calculator

ANTAL RÆKKER $\frac{10.000 \text{ rækker} * (7 + 11)}{(2 \text{ kbyte} - 84 \text{ bytes}) \times 1,20} = \frac{180\,000}{2048} \sim 294\,600\,000$ = 300M bytes

storage (initial 300M
next 300M
pctincrease 0
minextents 2
maxextents
);

Egne noter:

- For at oprette et index skal der foretages en sortering som forudsætter/kræver et temporært segment af en vis størrelse

-> "max # extents (%s) reached in temp segment in tablespace %s"
Cause: A temp segment tried to extend past max extents
Actn: If maxextents for tablespace is less than the sysd max, you can raise that. Otherwise, raise pctincrease for the tablespace

Øvelser

5. Databasens objekter/segmenter og pladsallokering

1. Log på som en almindelig bruger (opgives af instruktøren)

2. Opret en tabel '**test**' med flg. definition:

   Kolonner: afdnr number(2)
   afdnavn varchar2(14)
   bynavn varchar2(13)
   Storage: initial 8K
   next 3K
   pctincrease 50

3. Undersøg tabellens storagedefinition ved at forespørge på viewet **user_tables**.
   (bemærk eventuelle afrundinger)

4. Kopiér indholdet fra scott's **dept**-tabel over i tabellen test.

   Brug følgende SQL: `insert into test select * from scott.dept`

5. Kopiér indholdet af test-tabellen til sig selv gentagne gange - stop, når der indsættes 4096 rækker.

6. Udskriv en liste, der for hvert extent tilhørende tabellen **test** viser **extent_id**, extentstørrelsen i **blokke** og extent størrelsen i **bytes**. Oplysningerne findes i data dictionary viewet **user_extents.**

7. Noter antallet af blokke for de første 5-7 extents.
   Har de den størrelse du forventede? Nej extent_id 5 burde være 30

Undersøg om du selv har en tabel der hedder **emp**. Hvis den ikke findes under dit brugernavn, kør da scriptet; **demobld.sql**.
(findes i et bibliotek under `$ORACLE_HOME/sqlplus/demo`)

Fortsættes...

Egne noter:

ORA-00955: name is already used by an existing object
drop table test
Table created

TEST	TOOLS	pct_free 10	pct_used 40	ini_trans 1	max_trans 255
	initial_extent 8192	next_extent 4096	min_extents 1	max_extents 249	pct_increase 50
	~~num_rows~~	empty_blocks 0	c		

0	2	8192
1	1	4096
2	2	8192
3	3	12288
4	5	20480
5	10	40960
6	15	61440

5. Databasens objekter/segmenter og pladsallokering *(fortsat)*

9. Opret et unikt indeks på **empno** kolonnen i din **emp** tabel.
   - Prøv at indsætte den samme post i tabellen to gange. Hvad sker?

10. Opret et indeks på **ename**-kolonnen i **emp**-tabellen.
    Prøv at udføre følgende 2 select-sætninger:

```
select ename from emp;

select ename from emp where ename > ' ';
```

    - Hvad er forskellen i uddata?

11. Opret et indeks på **sal**-kolonnen i **emp**-tabellen.
    Prøv at udføre følgende select-sætning:

```
select sal*12 from emp where sal*12 > 2000
```

    - Er uddata sorteret?
    - Prøv at omskrive select-sætningen så indekset bruges.

$ORACLE_HOME/rdbms/admin/utlxplan.sql
create table PLAN_TABLE
explain plan for <statement>
set statement_id = 'xxx' into output

p 4-294 Ora7 Server "SQL Language Ref. Manual"

```
Select LPAD(' ', 2*(Level-1)) || operation operation, options, object_name
position
From output
start with id = 0 and statement_id = 'XXX'
connect by prior id = parent_id and statement_id = 'XXX'
```

Vedl training benyt 'Analyze table xxx COMPUTE STATISTICS'
og 'select * from user_tables where table_name = 'xxx''

Egne noter:

create unique index i_empno on emp(empno);
=> index created

insert into emp select * from emp where empno = 7370;
=> ORA-00001: unique constraint (ORA3.I_EMPNO) violated

create index i_ename on emp(ename);

Øvelser

6. Spacemanagement

1. Udfør følgende SQL:

```
select ename,sal,length(sal),vsize(sal) from emp;
```

   Ser du nogen forskel i de to sidste kolonner ?

2. Hvor mange tabeller i databasen har mere end 4 extents p.t.?

3. Undersøg om databasens **freespace** er fragmenteret.

4. Er der to sammehængende freesapace arealer der lige så godt kunne ha' været et stort ?

5. Lav en liste der viser pladsallokeringen (sumeret pr. tablespace) for alle tablespaces i databasen, størelsen på den største segment allokering, samt det største antal extents heri. (*se på nogle af eksemplerne på side 49 som inspiration*)

6. Hvor mange bytes pr. blok er der i tabellen TEST (den fra en tidligere øvelse)

   (*hint: Brug **vsize(xxx)** funktionen pr kolonne **summeret** pr. blok. En blok identificeres med **substr(rowid,1,8)** fiunktionen*)

Egne noter:

Øvelser

7. Brugere, privilegier og sikkerhed

1. Hvad er forskellen imellem standard-rollen "CONNECT" og standard-rollen "RESOURCE" - Brug enten DBA-manualen kap. 12 eller kig evt. i viewet: DBA_ROLES og DBA_SYS_PRIVS.
   resource har extra 'create procedure/trigger' nogler 'alter session', c... db link', c... ses
2. Udskriv en liste over de eksisterende brugere med et bruger-id der starter med "ORA..."og vis hvilke systemprivilegier (eller roller) disse har.

3. Opret en ny Oracle-bruger med dine initialer som brugernavn og password.
   create user ge identified by ge
4. Den nye bruger skal kunne logge sig på databasen, oprette tabeller, views, synonymer m.v., men skal ikke være "udvikler-typen" (*d.v.s. brugeren må ikke kunne oprette procedurer og triggers*)
   grant connect to ge
5. Den nye bruger skal have en begrænset allokeringsrettighed på 30 K i tablespace'et "TOOLS".
   90
   alter user ge default tablespace TOOLS quota 90K on TOOLS
6. Sørg for at den nye bruger altid opretter tabeller i tablespace'et TOOLS som default, hvis intet andet angives i forbindelse med **create table** kommandoer.
   alter user ge default tablespace TOOLS
   Talleved
7. Glem heller ikke at tage hensyn til den nye brugers eventuelle sorteringer.
   (*De temporære segmenter må jo helst ikke blive oprettet i SYSTEM tablespace*)
   alter user ge temporary tablespace TEMP
8. Forespørg mod viewet DBA_TS_QUOTAS og DBA_USERS, og se om tingene ser fornuftige ud på den nye bruger.
   tablespace_name : TOOLS
   bytes : 0
   max_bytes : 92208
   blocks : 0
   max_blocks : 23
   username, user_id, password, default tablespace, temporary_tablespace, created, profile
   DEFAULT
9. Ret password på den nye bruger.
   alter user ge identified by ge5678
10. Log på / connect som den nye bruger.

11. Opret 3 nye tabeller under den nye bruger og kald dem hhv. **tabel_s**, **tabel_u** og **tabel_d**. Tabellerne må gerne være tomme.

12. Opret privilegier på disse 3 tabeller, således at der tildeles hhv. **select, update** og **delete** privilegier til hhv. **tabel_s**, **tabel_u** og **tabel_d**. Privilegierne skal gives til alle brugere i databasen.

Fortsættes...

Egne noter:

1 Select * from role_sys_privs where role in ('CONNECT', 'RESOURCE');

ROLE	PRIVILEGE	ADM
CONNECT	ALTER SESSION	
—1.—	CREATE CLUSTER	
	CREATE	
	...	
RESOURCE	CREATE CLUSTER	
	CREATE PROCEDURE	
	SEQUENCE	
	TABLE	
	TRIGGER	

'c synonym', 'create view'

2. select username, user_id, ~~profile~~ from dba_users where username like 'ORA%';

dba_sys_privs : GRANTEE, PRIVILEGE, ADMIN_OPTION

ORA 9 ..., UNLIMITED TABLESPACE

dba_role_privs : GRANTEE, GRANTED_ROLE, ADM_OPTION, DEFAULT_ROLE

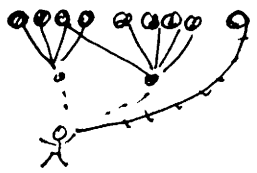
```
select u.username, p.privilege
from dba_users u , dba_sys_privs p
where u.username like 'ORA%'
and u.username = p.grantee
```

```
select * from dba_role_privs where grantee like 'ORA%';
ORA:...        CONNECT
:              RESOURCE
ORA 9          CONNECT
ORA 9          RESOURCE
```

DBA_SYS_PRIVS ○

ROLE_SYS_PRIVS

DBA_ROLE_PRIVS



← 80 stk

← roller

← bruger/roller

7. Brugere, privilegier og sikkerhed *(fortsat)*

13. Connect som **orax/orax** hvor **x** er det tal opgivet af instruktøren.

14. Afprøv privilegierne i forhold til de tre tabeller. (*D.v.s. forsøg med kommandoerne select, insert, update og delete på alle de 3 tabeller*)

15. Log på / connect som den nye bruger igen.

16. Kopiér **scott's emp**-tabel til den nye bruger. (*Angives som scott.emp. Prøv evt. at lave et synonym der dækker dette*)

17. Lad tabellen vokse ved at kopiere tabellens data i tabellen selv.

18. Kontrollér at disk-resourcen ikke kan overskrides. (*Se i USER_TS_QUOTAS eller DBA_TS_QUOTAS*)

19. Connect som system/manager og sæt disk-resourcen op til 100K.

20. Log på / connect som den nye bruger igen og afprøv delopgave 17. igen.

21. Connect som system/manager

22. Opret en rolle der kun kan logge på databasen (CREATE SESSION)

23. Opret endnu en bruger på databasen

24. Tildel rollen fra delopgave 22 til brugeren fra delopgave 23

25. connect som den bruger fra delopgave 23 og prøv at oprette en tabel. *(det skulle helst ikke kunne lade sig gøre*)

Egne noter:

Øvelser

8. Databaseoprettelse

Opret en helt ny database...
(*alle datafiler, kontrol filer og redologs oprettes i* **`/db7baser`** *directory'et som filer i filsystemet*)

find disse først !

Her lidt hints:

1. Find på et nyt ORACLE_SID til dig selv.

2. Opret en **init.ora** fil.
   Husk:
   - control_files = /db7baser/xxxxx
   - db_name=xxx,
   - db_block_size=2048 eller *4096*
   - init_sql_files=(/usr/ora7/dbs/sql.bsq, /usr/ora7/rdbms/admin/catalog.sql)
   Find selv andre relevante parametre hertil.

3. Opret selve databasen med **`create database`** komandoen.
   - Datafilen der skal danne fundament til SYSTEM tablespace skal være max. 10 Mb.
   - Redologs skal ikke være for store (max 500Kb. pr. stk.)
   - Opret et passende antal tablespaces - og glem nu ikke rollback segmenterne !

4. Tilret init.ora filen således at den bringer dine nye rollback segmenter **online** ved hver opstart fremover.

5. Kør det script der opretter alle DBA synonymerne under system/manager.

6. Opret en bruger med "connect" rettighed. - Husk defaults, temporary segmenterne og quotas på et relevant tablespace for denne bruger

7. Kør **`/usr/ora7/sqlplus/demo/demobld.sql`** scriptet under denne nye bruger.

   - check at alt ser fornuftigt ud... Se om du kan selecte fra **emp, dept** m.v...

Egne noter:

Øvelser

9. Logisk backup og restore (exp/imp)

Følgende delopgaver foretages på din ny-oprettede database fra sidste øvelse:

Logisk backup:

1. Connected som system/manager; tag da en export af den brugers tabeller du oprettede i pkt 6 og pkt. 7 i sidste øvelse.
   Eksportfilen skal ligge i **/tmp** directory'et

2. Når eksporten er foretaget, importer da **dept** tabellen ind igen.

3. Gå ind i sqlplus som din bruger fra pkt 6. i sidste øvelse. Hvordan ser din **dept** tabel ud?

4. Afhjælp dobbeltlagringen.
   Kan klares på to måder:
   - enten drop dept tabellen og importer den igen.
   - eller opret en ny temporær tabel med `create table ... as select ...`,
     drop dept og rename den temporære tabel til **dept** igen.

5. Lav en kopi af **emp** tabellen med `create table xxx as select * from emp`

6. Sørg for at den nye tabel udvides allokeringsmæssigt til 3-4 extents med
   `insert into ... select * from ...`

7. Tag en total export af databasen. Husk at samle (reorganisere) fragmenterede tabeller til enkelte extents. Eksportfilen skal igen ligge i **/tmp** directory'et

8. Drop tabellen fra delopgave 5 og 6.

9. Importér tabellen du droppede i delopgave 8.

10. Gå ind i sqlplus og se hvordan allokeringen nu ser ud på tabellen du importerede i delopgave 9.

11. Prøv evt. at importere til en "index-fil" og se indholdet her i (En udokumenteret feature i manualerne. Angives med `imp system/manager file=... indexfile=...`)

Fortsættes med fysisk backup....

Egne noter:

Øvelser

9. Fysisk backup, hot backup og recovery

Følgende delopgaver foretages på din ny-oprettede database. Hvis der er tidspress, spring da direkte til delopgave 6:

1. Luk din database ned og tag en fysisk backup af dit database miljø
   - Klares i UNIX med **mv** kommandoen hvis der er tale om *filer* eller **dskback** hvis databasen er oprettet på *rå/subdiske*.

2. Start databasen op igen og lav nogle opdateringer på din database.

3. Luk databasen igen.

4. Prøv derfter at slette en af dine datafiler, redolog-filer eller control-filen for så at forsøge at start databasen op igen.
   - Noter fejlmeddelelsen

5. Få databasen op på højkant igen og se hvordan opdateringerne fra delopgave 2 har det.

   - - - - - - - - - - - - - - - -

6. Din database skal nu køre i ARCHIVE LOG MODE. Baggrundsprocessen ARCH skal køre og sættes op med en arkiverings destination i `/TMP`. Find på et fornuftigt navne-format til destinations filerne.

7. Når alt er oppe igen, tag en hot/online backup af databasen tablespace for tablespace.

8. Lav nogle opdateringer på din database.

9. Tving et "checkpoint" og "log switch" igennem på databasen og gentag delopgave 8 + 9 et par gange...

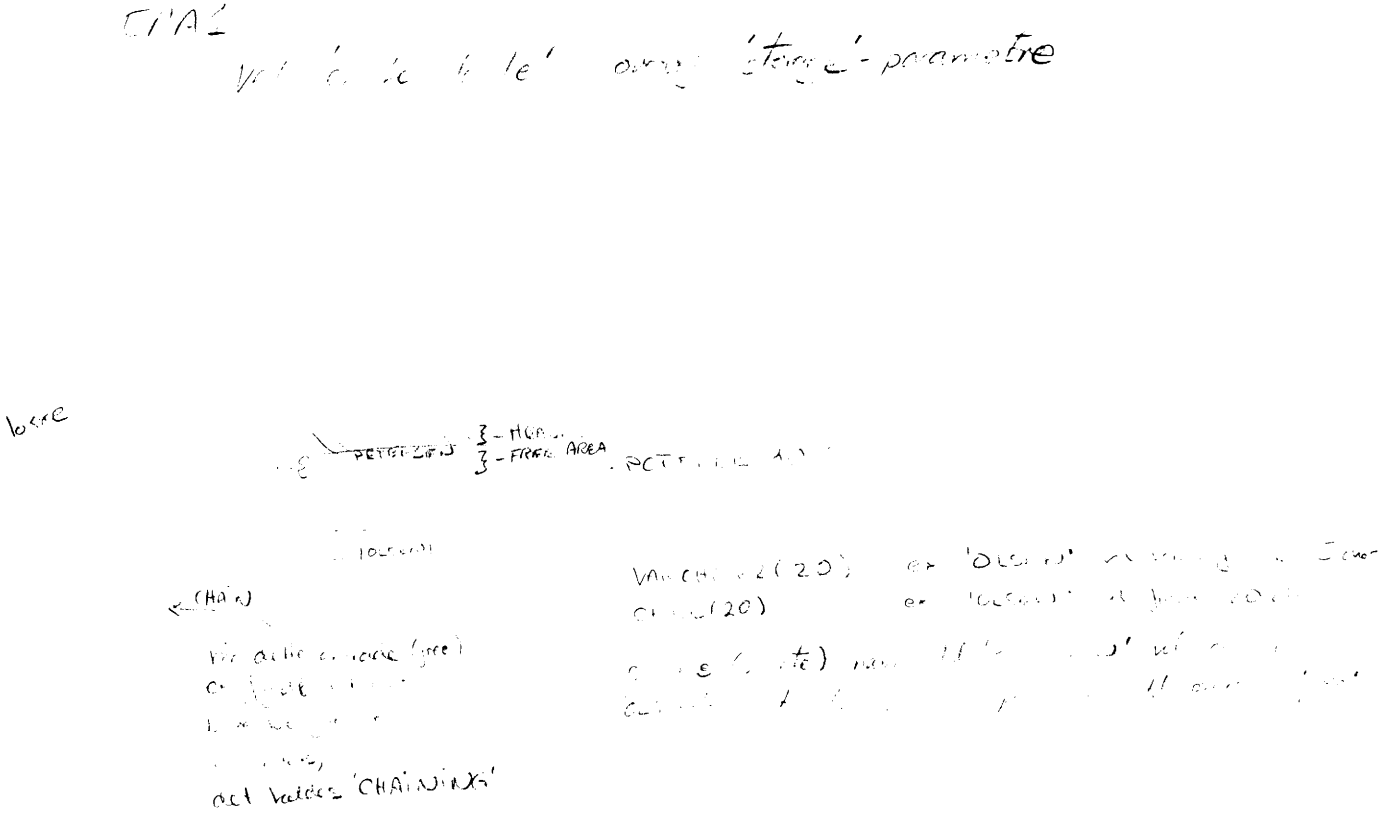
10. Simulér et diskcrash ved at slette den datafil, der benyttes af det tablespace hvori du foretog opdateringer i delopgave 8.

11. Luk databasen ned hurtigst muligt !

12. Restore datafilen (eller datafilerne) og start databasen op på normal vis.
    - Måske kan databasen recover uden din hjælp.
    - Ellers hjælp den med at finde de arkiverede redologs i recovery processen.

13. Check om alle dine opdateringer fra delopgave 8 er på plads igen.

CREATE VIEW
AS SELECT.
:
WITH CHECK OPTION; ← Tillader ikke 'update' hvis det medfører at række forsvinder ud af dette view!



SCLDCA :

Esc K (Show keys)

Ctl G (Menu)

Dansk Data Elektronik A/S
Herlev Hovedgade 199
DK 2730 Herlev
Tel.: (+45) 42 84 50 11
Fax: (+45) 42 84 52 20