Adatbázisok II.

1

Jánosi-Rancz Katalin Tünde

tsuto@ms.sapientia.ro

327A

Mi minden van egy (Oracle) adatbázisban?

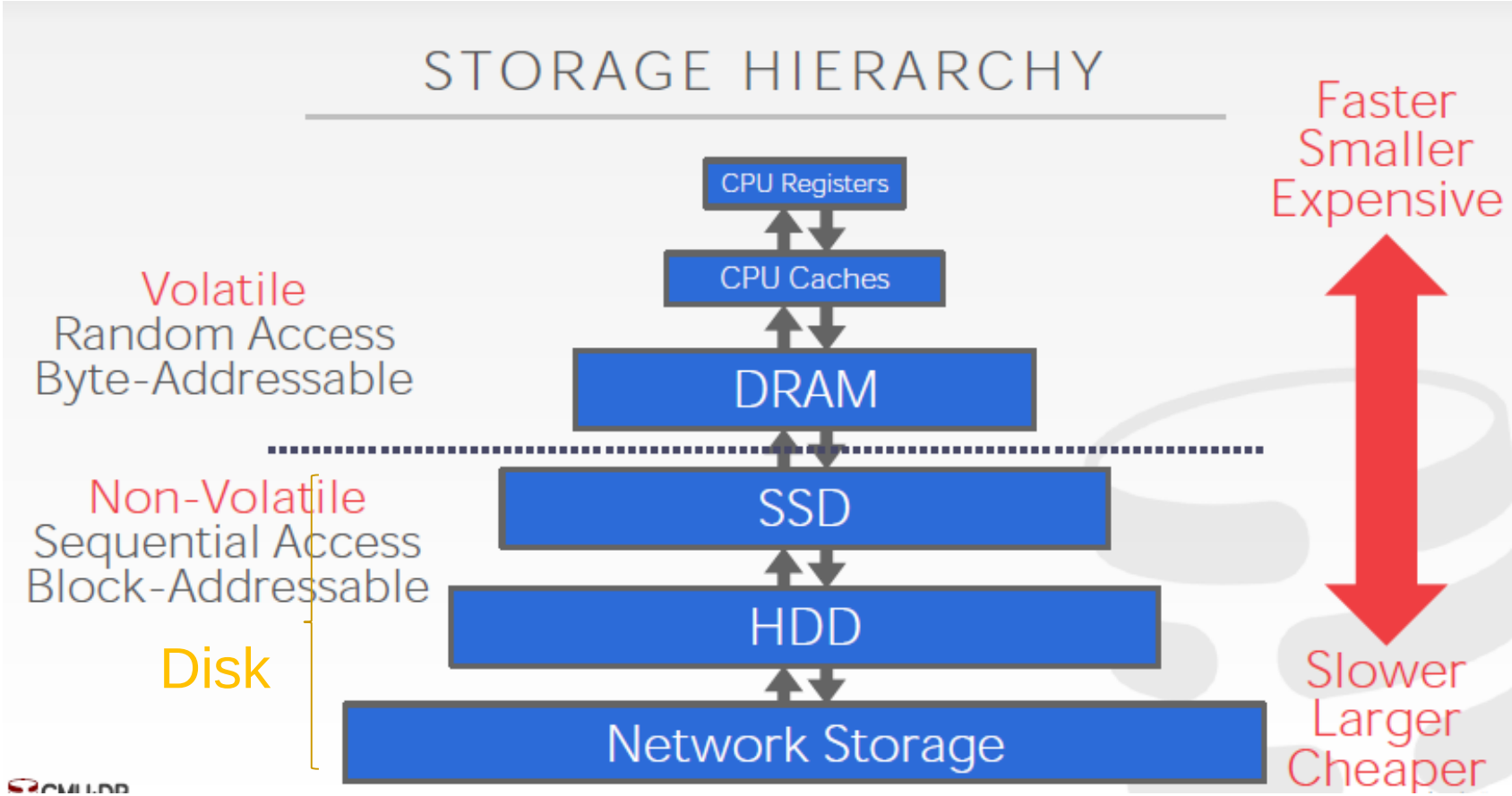
Tárolási alapfogalmak, objektumok, adatszótár nézetek

3 féle tárolóeszköz

- Illékony eszközök (Volatile device), „memory oriented” nevezzük:
  - ha a készüléket kivesszük áram alól, akkor az adatok elvesznek.
  - támogatja a gyors véletlenszerű hozzáférést bájtcímzésű helyekre. A program bármilyen bájtcímre képes ugrani, és megkaphatja az ott található adatokat.
- Nem illékony eszközök (Non-Volatile devices), „disk oriented” nevezzük:
  - a tároló eszköznek nem kell folyamatos energiát szolgáltatnia ahhoz, hogy az eszköz megtartsa a tárolt biteket.
  - ez is blokk/page címezhető. Az érték egy adott eltolásnál történő leolvasása érdekében a programnak először be kell töltenie a 4 KB-os oldalt a memóriába, amely azt az értéket tárolja, amelyet a program el akar olvasni.
  - jobb a szekvenciális hozzáférésnél (több darab adat olvasása egyszerre).
  - nem teszünk különbséget a SSD és a HDD között.
- Nem felejtő memória (Non-volatile memory device)
  - hamarosan megjelenik
  - szinte olyan gyors, mint a DRAM, de a lemez megmaradásával.

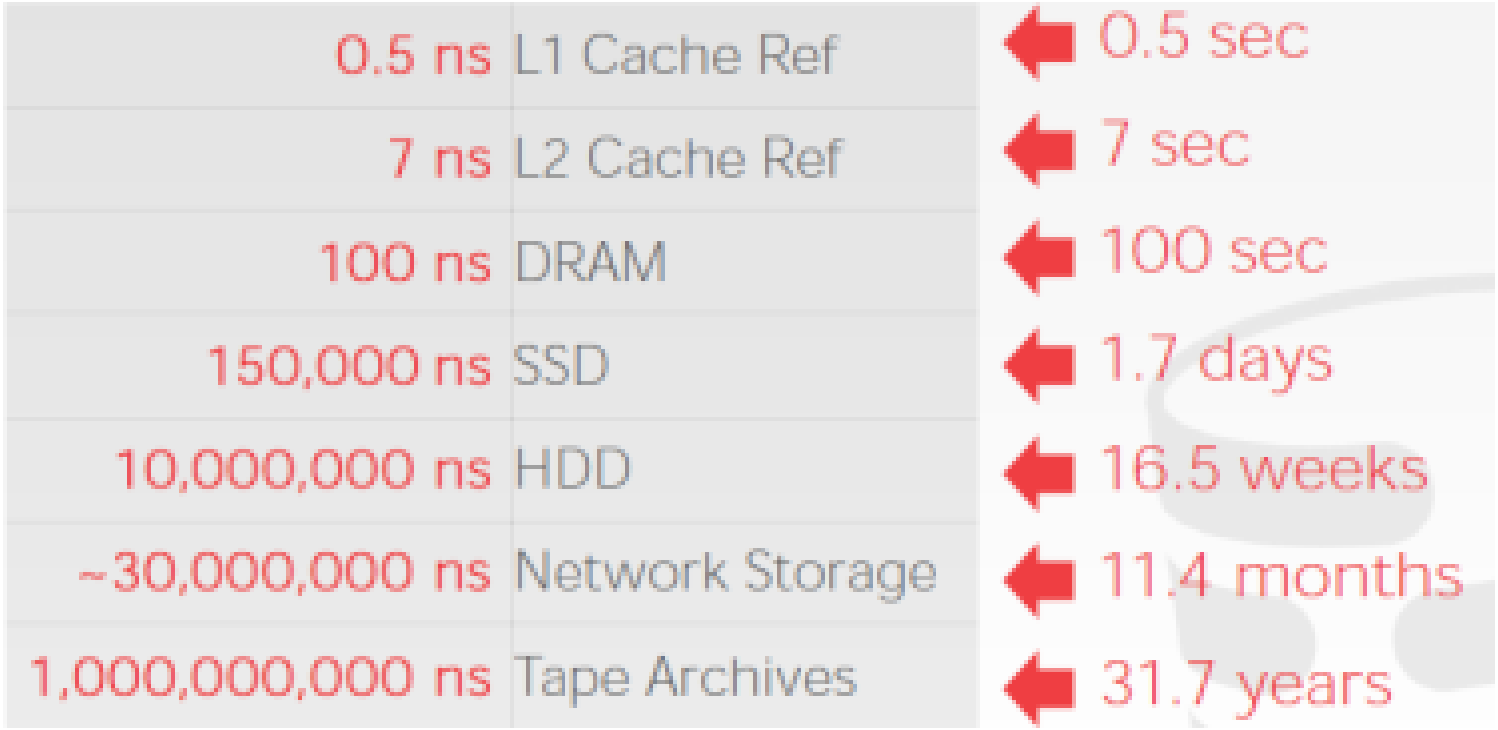
Tárolás

- Az ABKR összetevői felelnek az adatok oda-vissza mozgatásának módjáról



Elérési idő

- Ha az adatok leolvasása az L1 gyorsítótár referenciából fél másodpercig tart, akkor az SSD-ről történő olvasás 1,7 napot igényel, a HDD-ről pedig 16,5 hetet vesz igénybe.



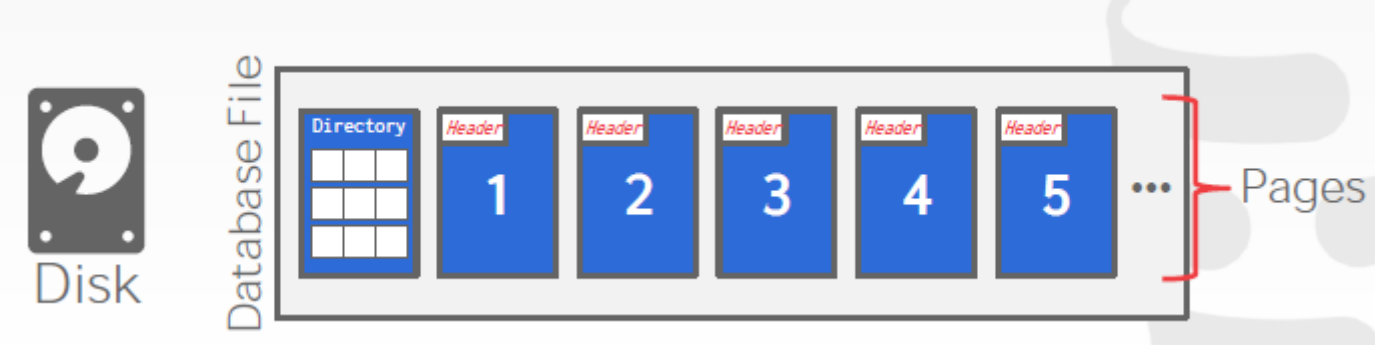
- https://gist.github.com/hellerbarde/2843375

Rendszer tervezési célok

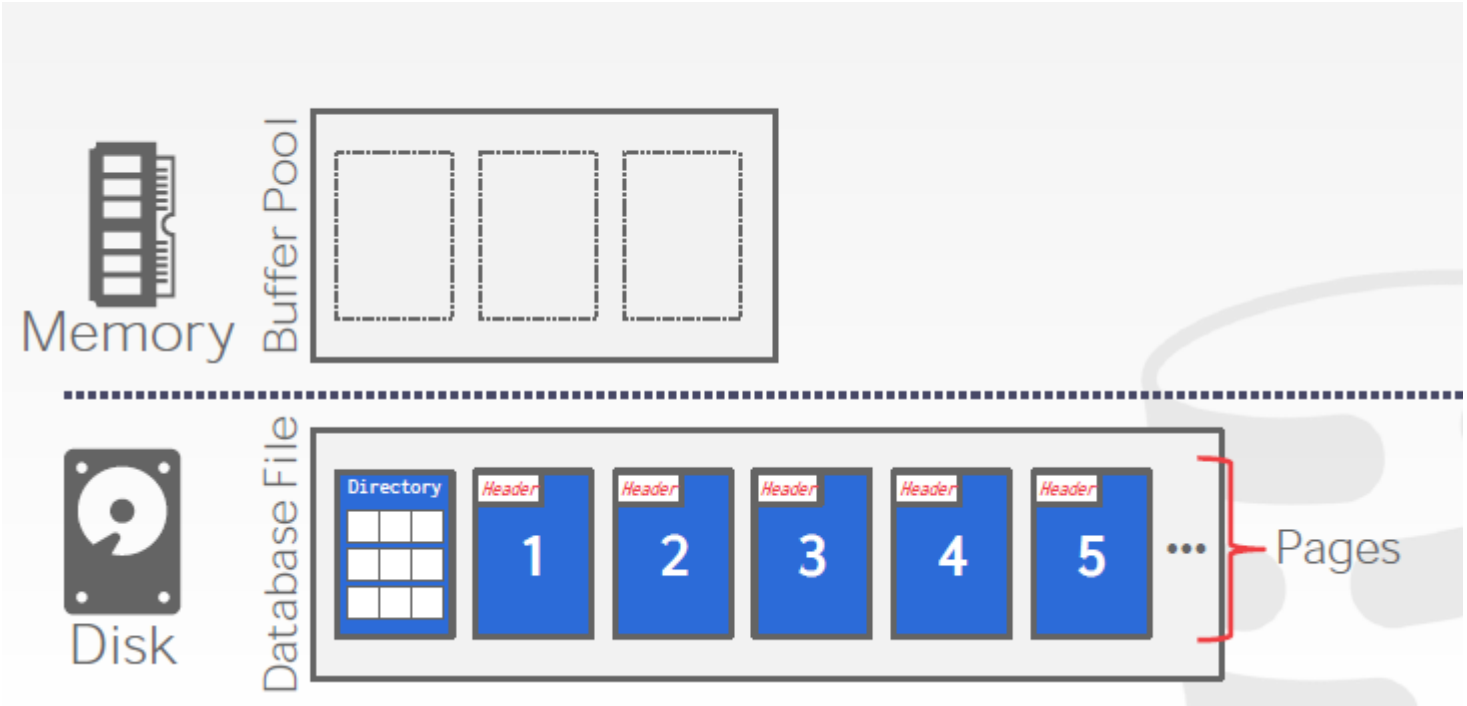
- olyan adatbázisok támogatása, amelyek képesek meghaladni a rendelkezésre álló memória mennyiséget
- a lemezen történő olvasás/írás drága
  - nem akarjuk, hogy nagy leállások legyenek
  - ha valamit a lemezről lehozunk, ne mindent lassítson le
- az ABKR képes legyen más lekérdezések feldolgozására, miközben arra vár, hogy megkapja az adatokat a lemezről.

Lemezorientált tárolás

- az adatbázis mind a lemezen van, az adatbázis fájlokban szereplő adatok oldalakba vannak rendezve, és az első oldal a könyvtár. Az adatokkal való működéshez a DBMS-nek az adatokat a memóriába kell helyeznie
- az operációs rendszer semmit sem tud ezeknek a fájloknak a tartalmáról!

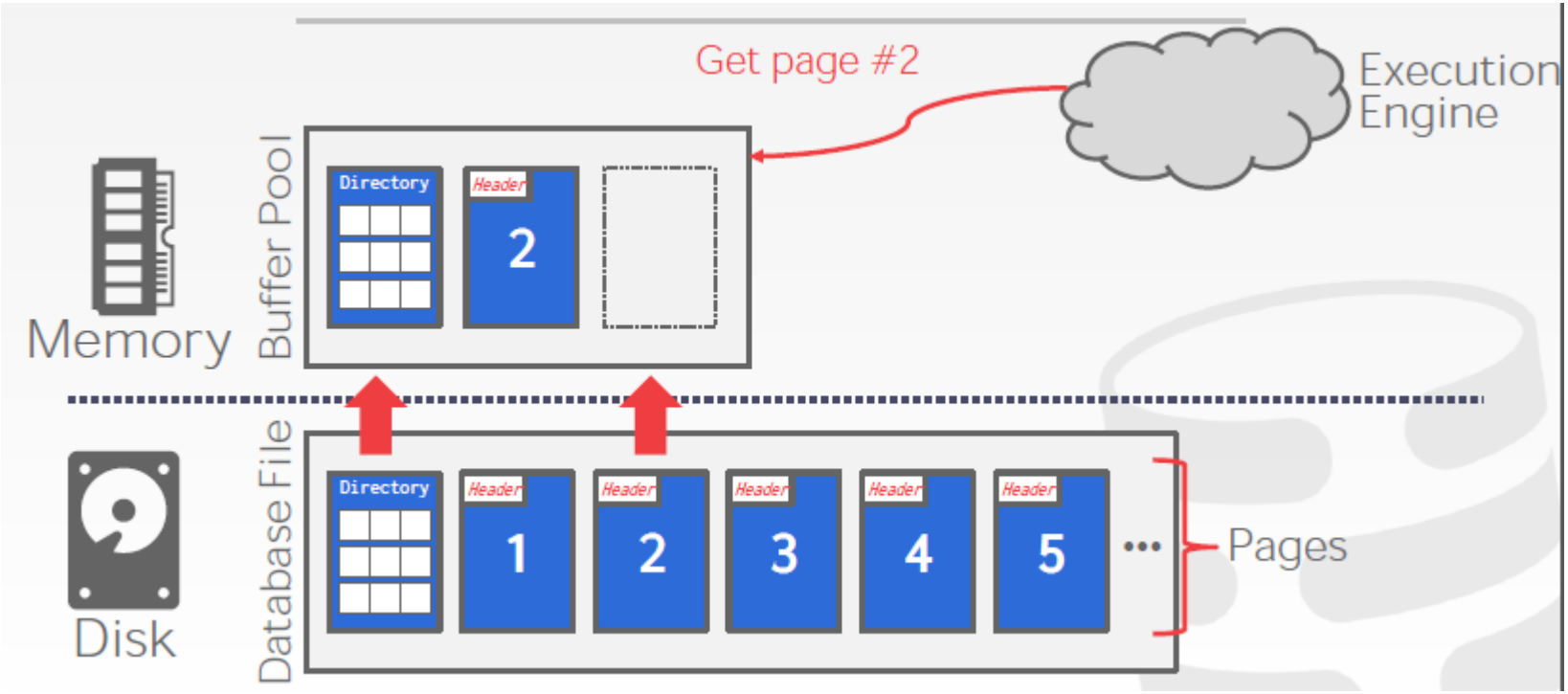


- az ABKR rendelkezik egy pufferkészlettel, amely kezeli a lemez és a memória közötti oda-vissza mozgást. Van egy végrehajtó motorja is, amely lekérdezéseket hajt végre. A végrehajtó motor egy konkrét oldalt kér a pufferkészlettől, és a pufferkészlet gondoskodik arról, hogy az oldal a memóriába kerüljön, és a végrehajtó motornak mutatót adjon a memória oldalához. A pufferkészlet-kezelő biztosítja, hogy az oldal ott legyen, amíg a végrehajtó motor ezen a memórián működik.
- mindezt az ABKR oldja meg és nem az Operációs Rendszer!



Lemezorientált ABKR

- A Page-k különféle típusú adatokat tartalmazhatnak (táblák, indexek, log records stb.)
- minden Page egyedi azonosítóval rendelkezik
- a legtöbb ABKR rögzített méretű Page-ket használ
- egy Page = egy rögzített méretű adatblokk



ABKR vs. OS?

- Nem javasoljuk az mmap (OS) használatát az ABKR-ben a helyesség és a teljesítmény érdekében!

- A DBMS (szinte) mindig ellenőrizni akarja a dolgokat önmagában, és jobb munkát tud végezni.
  - A piszkos oldalak megfelelő sorrendben való lemezre írása
  - Speciális előhívás
  - A puffercsere-politika.
  - Thread/process ütemezése

- az OS **nem** a te barátod!

- ha az OS-t tennénk felelőssé az oldalak oda-vissza mozgatásáért a lemez és a memória között, nem nyújtana jobb vezérlést és teljesítményt!!!

Fizikai és logikai szint

- Az adatbázisok vizsgálatánál elkülönítünk egy fizikai és egy logikai szintet.

- Logikai szint
  - Az adatbázis objektumai (tábla, nézet, index ... stb.)

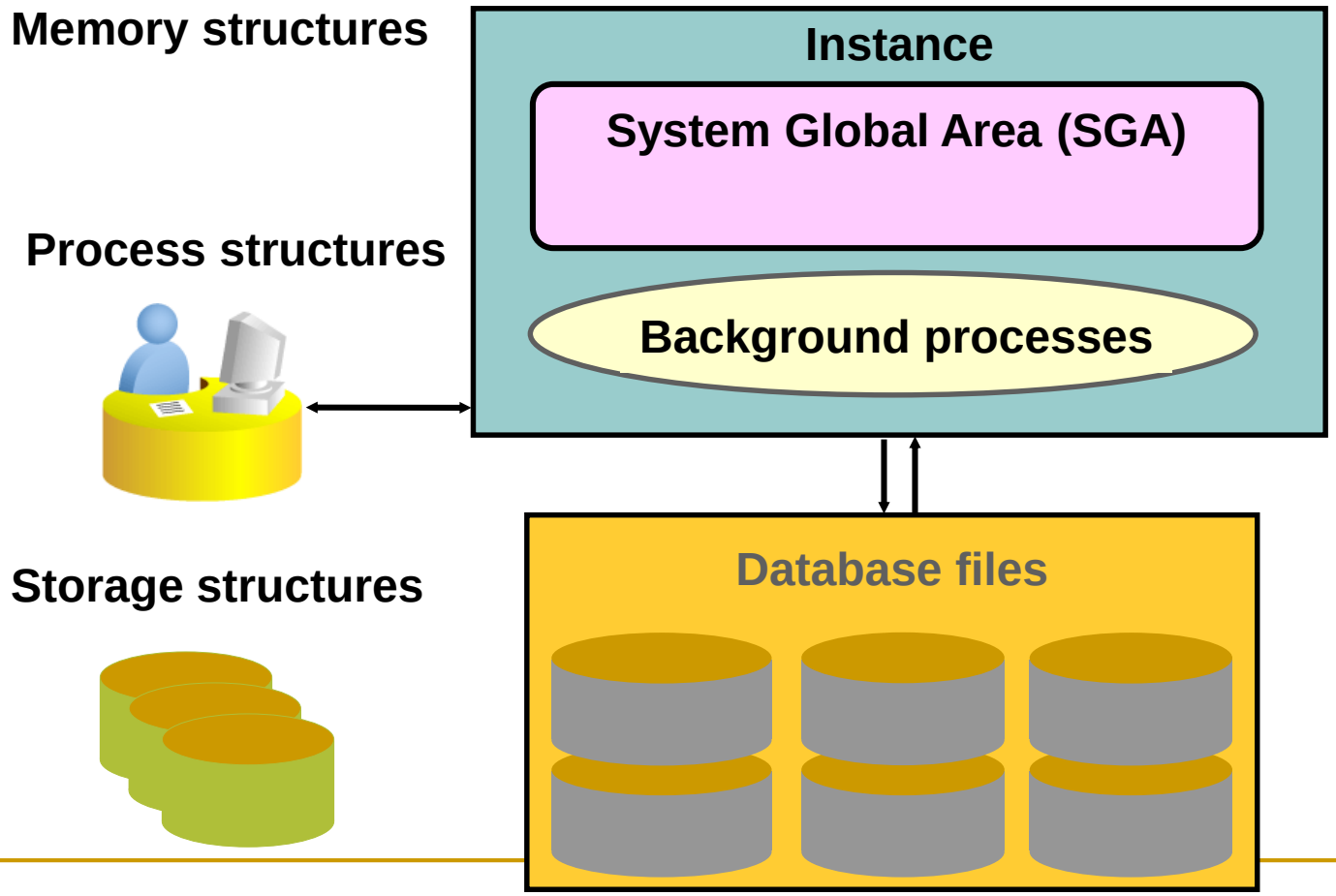
  - Logikai szinten „fogalmakkal” dolgozunk és közömbös számunkra, hogy ezek miként tárolódnak, illetve miként valósítja meg azokat a rendszer.

- Fizikai szint (fájlok)
  - Adatfájlok (ebben vannak tárolva ténylegesen az adatok -> ***.dbf**)
  - Naplófájlok (ebbe íródnak a módosításokról készült naplóbejegyzések -> ***.log**)
  - Vezérlőállományok (mindenféle információk az adatbázisról -> ***.ctl**)
  - Paraméterállomány (**init.ora**), jelszóállomány... stb.

  - A fizikai szinten konkrétan azt vizsgáljuk, hogy az egyes adatbázis-objektumok miként tárolódnak, miként implementálják az egyes adatbázis-műveleteket stb.

- A logikai adattárolás legnagyobb egységei a **tablespace**k, míg a fizikai tárolás **datafile**ok formájában történik.

Az Oracle adatbázis architektúra



Az Oracle adatbázis architektúra

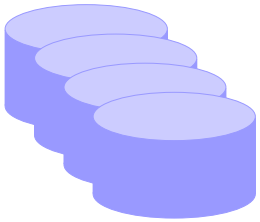
- Az Oracle adatbázis két fő részből áll:

1. Az adatbázis vagyis a fizikai struktúrák rendszere:
   - A vezérlő fájl (control file), mely az adatbázis konfigurációját tárolja
   - A helyrehozó napló fájlok (redo log files), amikben a helyreállításhoz szükséges információkat tároljuk
   - Az adatfájlok, amelyekben az összes tárolt adat szerepel
   - Paraméterfájl, amelybe olyan paramétereket tárolunk, amelyek befolyásolják egy példány méretét és tulajdonságait
   - Jelszófájl, amelyben az AB administratorok (SYSDBA) jelszavát tároljuk

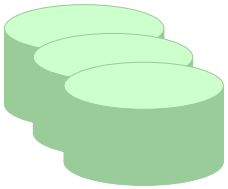
2. A példány (instance) vagyis a memória struktúrák és folyamatok rendszere:
   - A memóriában lefoglalt System Global Area (SGA) terület és azok a szerverfolyamatok, amelyek az adatbázis-műveletek végrehajtásáért felelősek.

Physical Database Structure

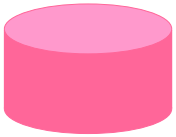
Control files



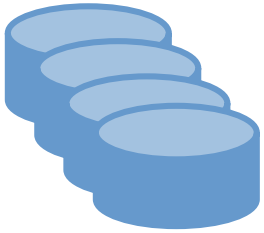
•Data files



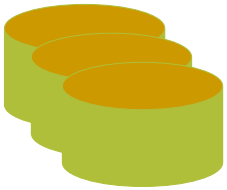
•Online redo log files



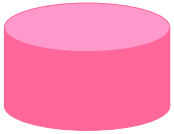
•Parameter file



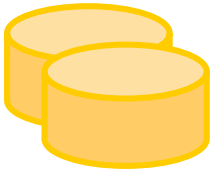
•Backup files



•Archive log files



•Password file



•Alert and trace log files

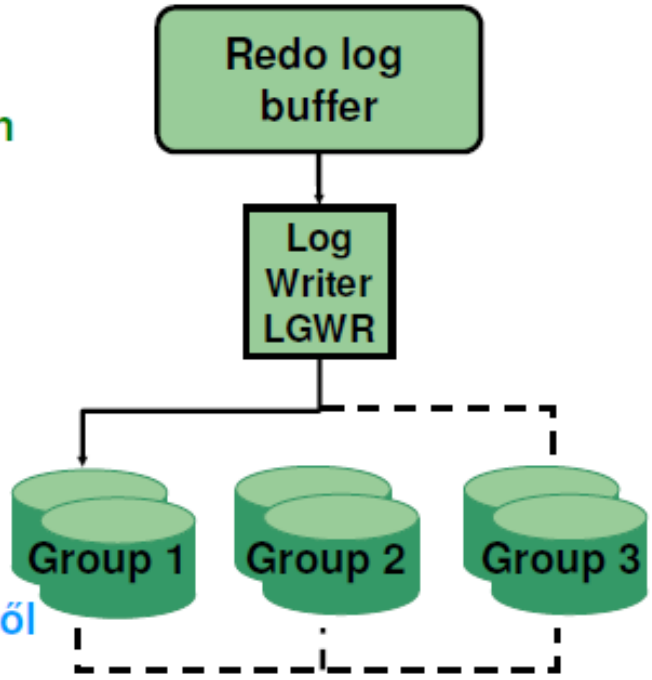
A vezérlő fájlok (Control files)

Control files

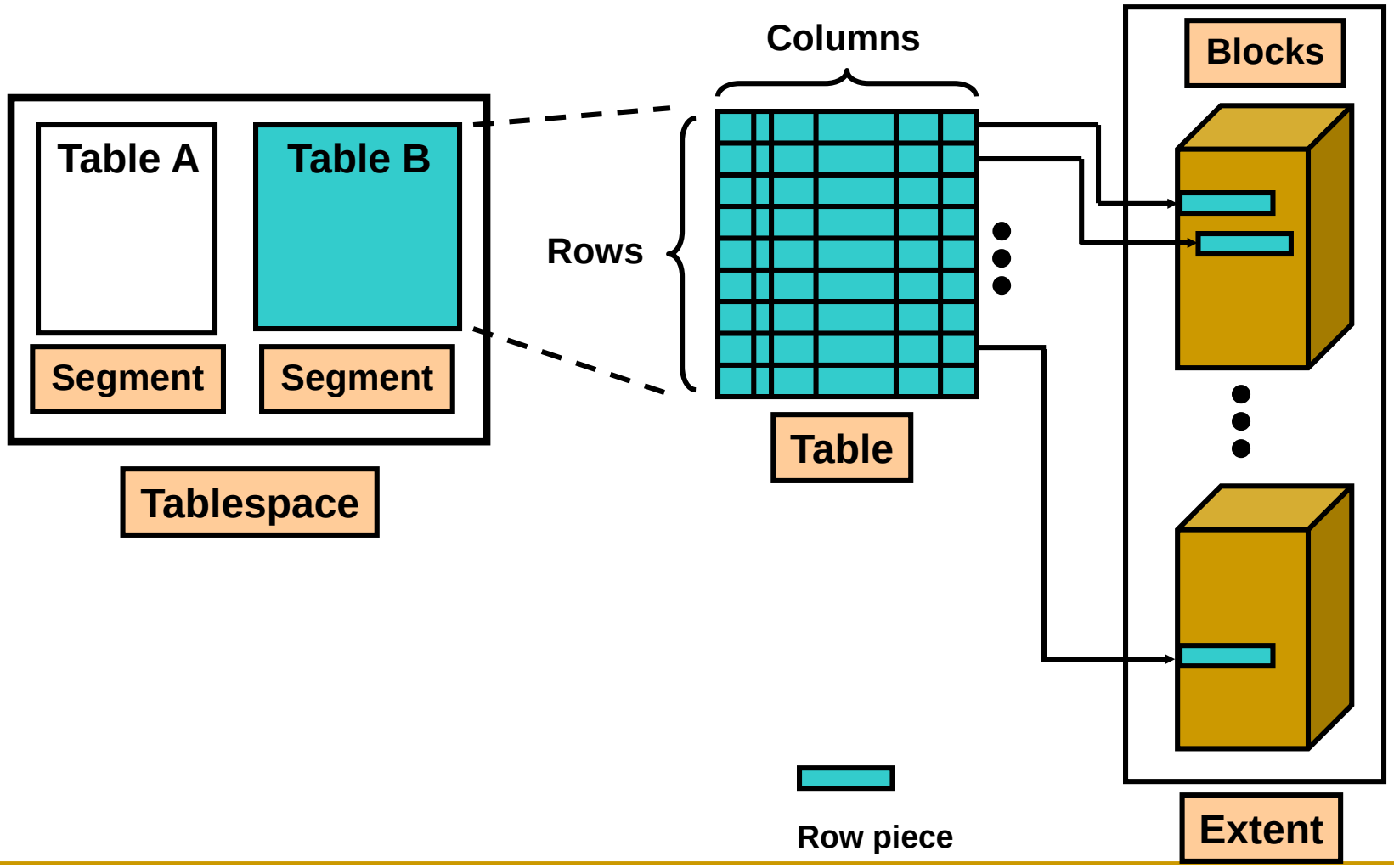
- A példány indításakor az adatbázis rákapcsolásához (mount) be kell olvasni a vezérlő fájlokat.
- Az adatbázist alkotó fizikai fájlokat határozza meg.
- Ha új fájlt adunk az adatbázishoz, akkor automatikusan módosulnak.
- A vezérlő fájlok helyét az inicializálási paraméterben adjuk meg.
- Adatvédelmi szempontból legalább három különböző fizikai eszközön legyen másolata (**multiplex the control files**). Ez is inicializálási paraméterrel adható meg. Az Oracle szerver elvégzi a többszörös másolatok szinkronizációját.

Napló állományok (Redo Log Files)

- Az adatbázis változtatásait rögzíti
- Konzisztens állapot visszaállításhoz szükséges rendszerhiba, áramszünet, lemezhiba, stb. esetén
- Adatvédelmi okokból különböző lemezeken többszörös másolatokat kell belőle szinkronizáltan kezelni.
- A REDO napló REDO fájlcsoportokból áll.
- Egy csoport egy naplófájlból és annak multiplexelt másolataiból áll.
- Minden csoportnak van egy azonosítószáma.
- A naplóíró folyamat (log writer process - LGWR) írja ki a REDO rekordokat a pufferből egy REDO csoportba, amíg vagy tele nem lesz a fájl, vagy nem érkezik egy direkt felszólítás, hogy a következő REDO csoportba folytassa a kiírást.
- A REDO csoportok feltöltése körkörösen történik.

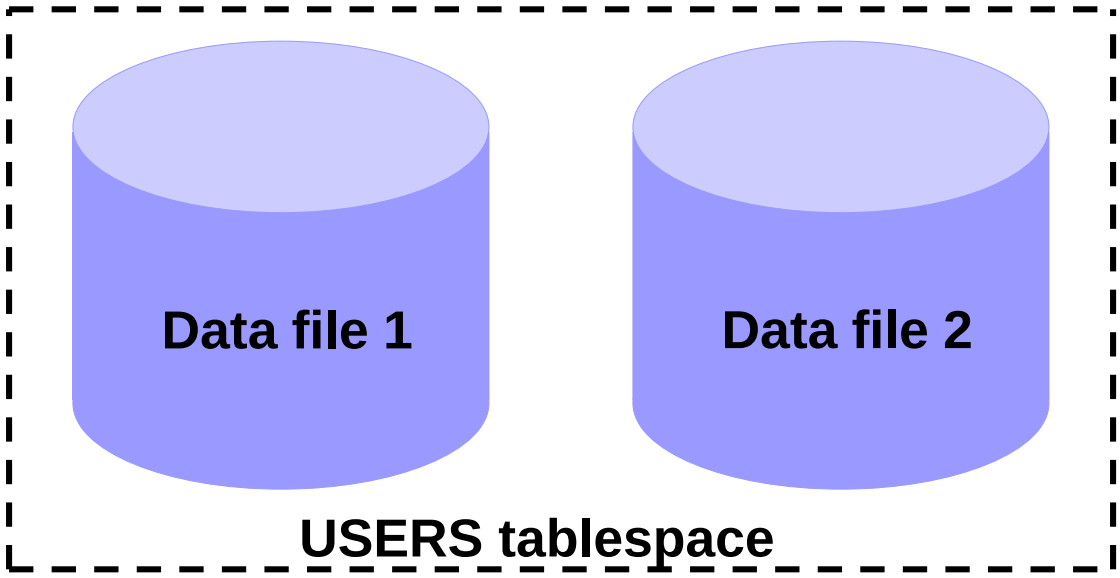


Hogyan tárolódnak a Táblák?

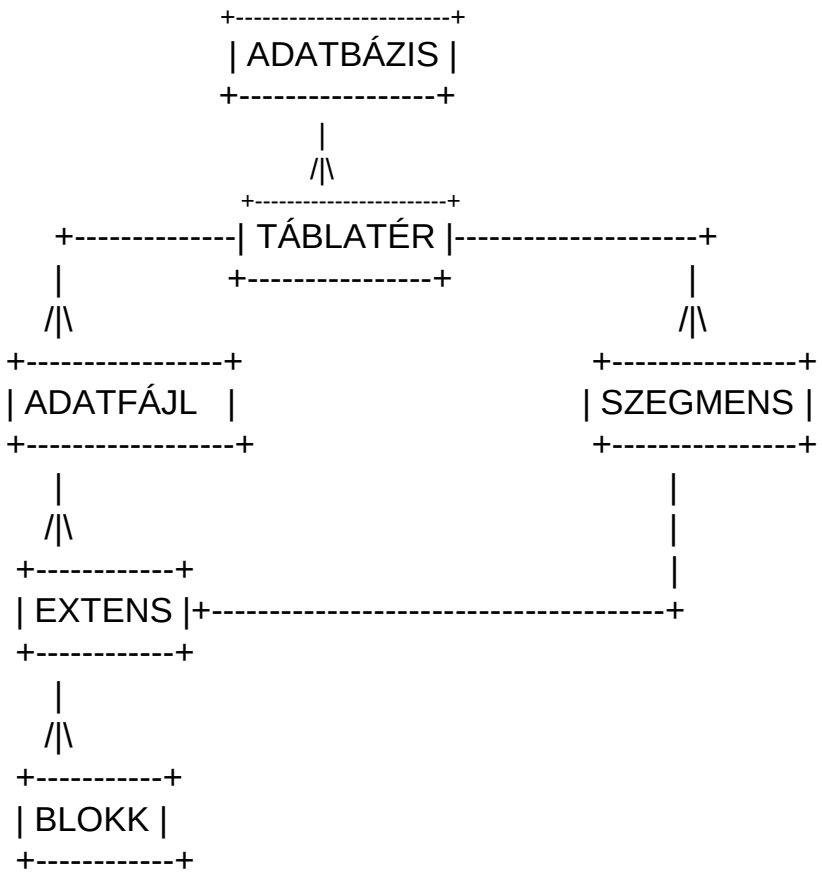


Táblaterek és adatfileok

- Egy táblatér 1 vagy több adatfileból áll
- Egy adatfile csak egyetlen táblatérhez tartozik



A tárolással kapcsolatos fogalmak és azok egymással való kapcsolata



Táblaterek I.

- Általában egy-egy adatbázis több táblatérből (tablespace) áll.
- A táblatér a „valamilyen szempontból összetartozó” adatbázis komponenseket gyűjti egybe.
- Minden adatbázishoz hozzátartozik egy SYSTEM táblatér, amely a rendszerkatalógus-táblákat tárolja.
- Általában az a jó, ha ezen a táblaterületen nem is helyezkedik el más, a katalógusokra ugyanis az Oracle-nek folyamatosan szüksége van. Ha más is használná a táblaterületet, az könnyen töredezetté válhatna, ami rontaná a teljesítményt.
- A SYSAUX táblatér SYSTEM táblatér mellett létrehozott segédterület. Több adatbázis komponens is alapesetben itt tárolódik.
- Azok az adatbázisra vonatkozó metaadatok, amelyek nem a SYSTEM táblaterületen tárolódnak, szintén idekerülnek.

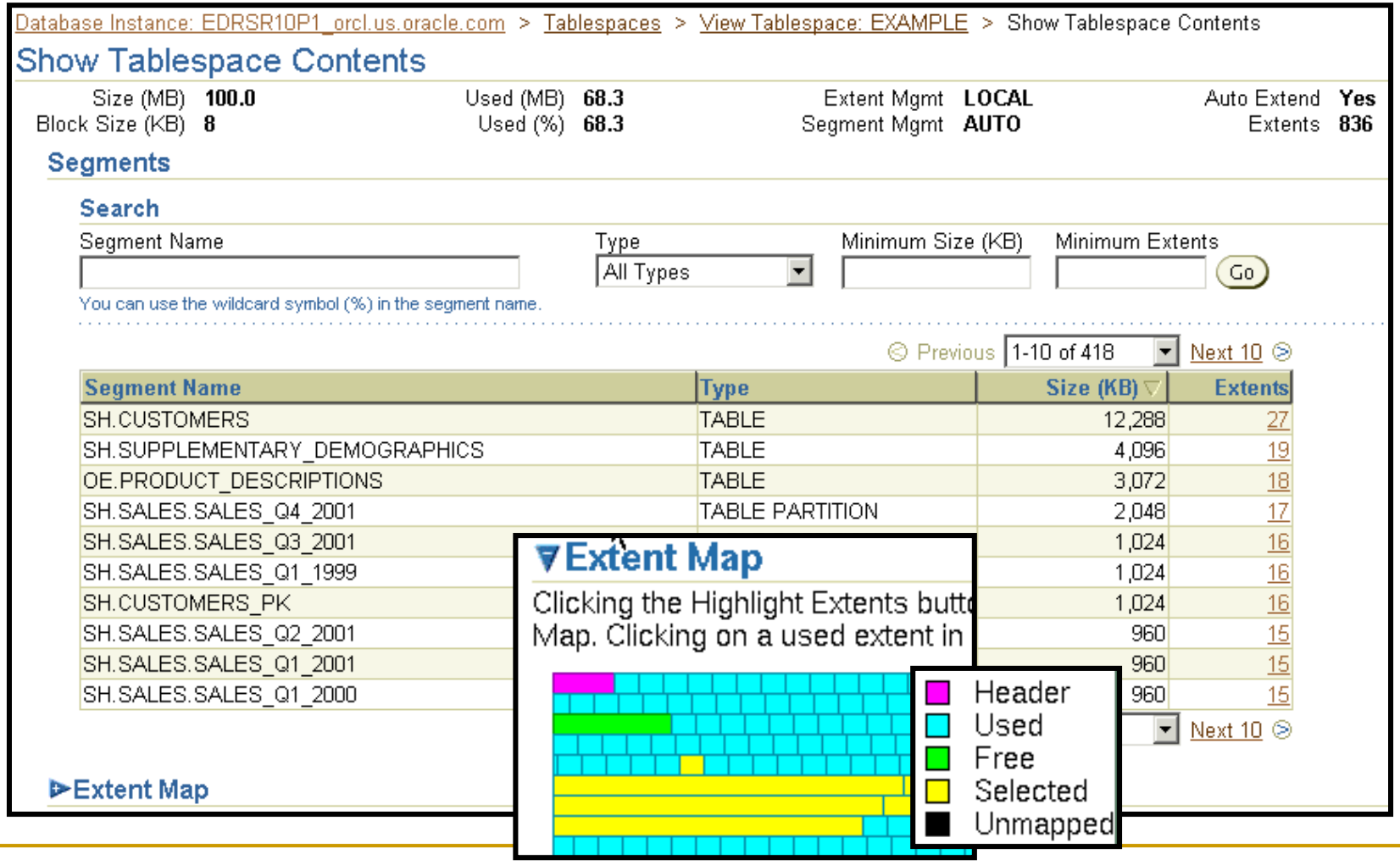
Táblaterek II.

- Az UNDO táblateret elsősorban a tranzakciók végrehajtásánál használja az Oracle, ha nem akarjuk manuálisan kezelni a visszagörgetési táblaterületeket.
- Ez a táblaterület biztosítja, hogy a sikertelen tranzakciókat visszagörgethesse a rendszer.
- Egy éppen változás alatt lévő tábláról szintén ennek segítségével kaphatnak konzisztens nézetet a felhasználók.
- A felhasználóknak egy külön USER táblaterületet szoktak létrehozni.
- Ezenkívül legalább egy ideiglenes (TEMPORARY) táblaterületet is létre szoktak hozni az ideiglenes objektumok számára. Ha ez nincs, az Oracle a SYSTEM táblaterületen tárolja az ideiglenes adatokat.
- Sok esetben érdemes még egy ideiglenes táblaterületet definiálni csak a rendezések számára. Rendezést használnak: a DISTINCT, GROUP BY, ORDER BY műveletek, a halmazműveletek, de a statisztikák gyűjtésénél, indexek felépítésénél szintén szükség van rendezésre.

Táblaterek III

- Az adatbázisunk méretét háromféleképpen növelhetjük:
  - új tablespacet hozunk létre
  - egy már létező tablespacehez új datafilet adunk hozzá
  - egy már létező datafile méretet növeljük (engedélyezhetjük, hogy az adatbázis ezt dinamikus hajtsa végre, amint szüksége van több területre)

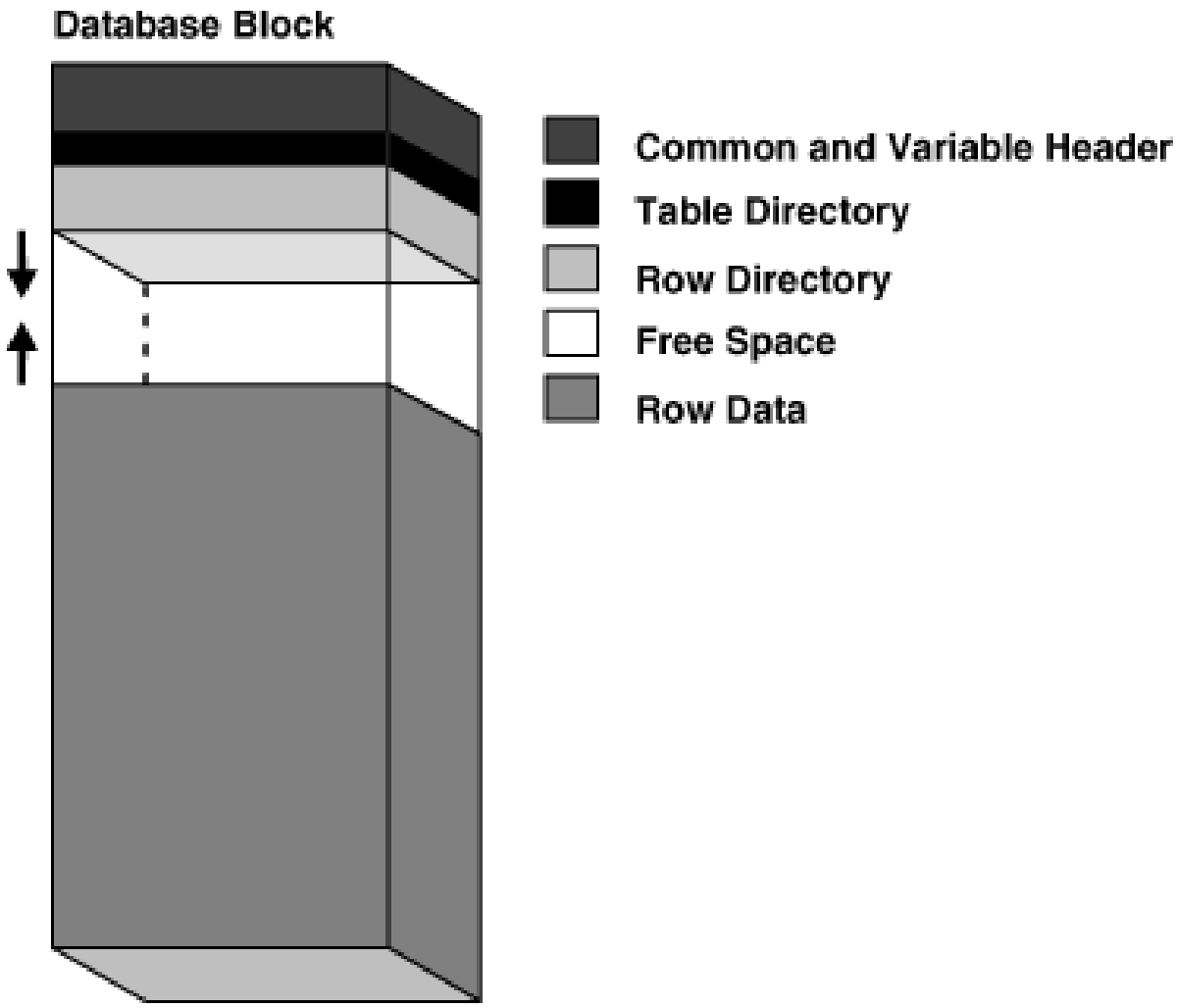
Viewing Tablespace Contents



Adattárolás logikai megvalósítása

- A legkisebb adategység, amit az Oracle kezelni képes a blokk (8k) . Egy-egy blokk bizonyos mennyiségű bájtnak felel meg az adattárolón.
- Az Oracle által kezelt blokkok mérete általában különbözik a háttérben működő operációs rendszer blokkméretétől, viszont annak csak többszöröse lehet.
- Az extensek a háttértárolón folytonosan elhelyezkedő adatblokkokból állnak.
- Egy-egy szegmens egy-egy objektumnak felel meg. Egy szegmens egy fizikailag tárolt objektum. Particionált táblák és indexek esetén minden partíció külön szegmensen helyezkedik el.
- Az Oracle a létrehozáskor egyetlen extensen tárolja a szegmenst, ám ahogy az bővül, a rendszer újabb és újabb extenseket foglal le, amelyek az esetek nagyrészében nem folytonosan helyezkednek el.
- Egy szegmens nem tartozhat több táblaterülethez, az viszont előfordulhat, hogy több adatfájlon tárolódik. Ezzel szemben egy extens mindig egy adatfájlba kerül.

Blokkok I.


Database Block
Common and Variable Header
Table Directory
Row Directory
Free Space
Row Data

Blokkok II.

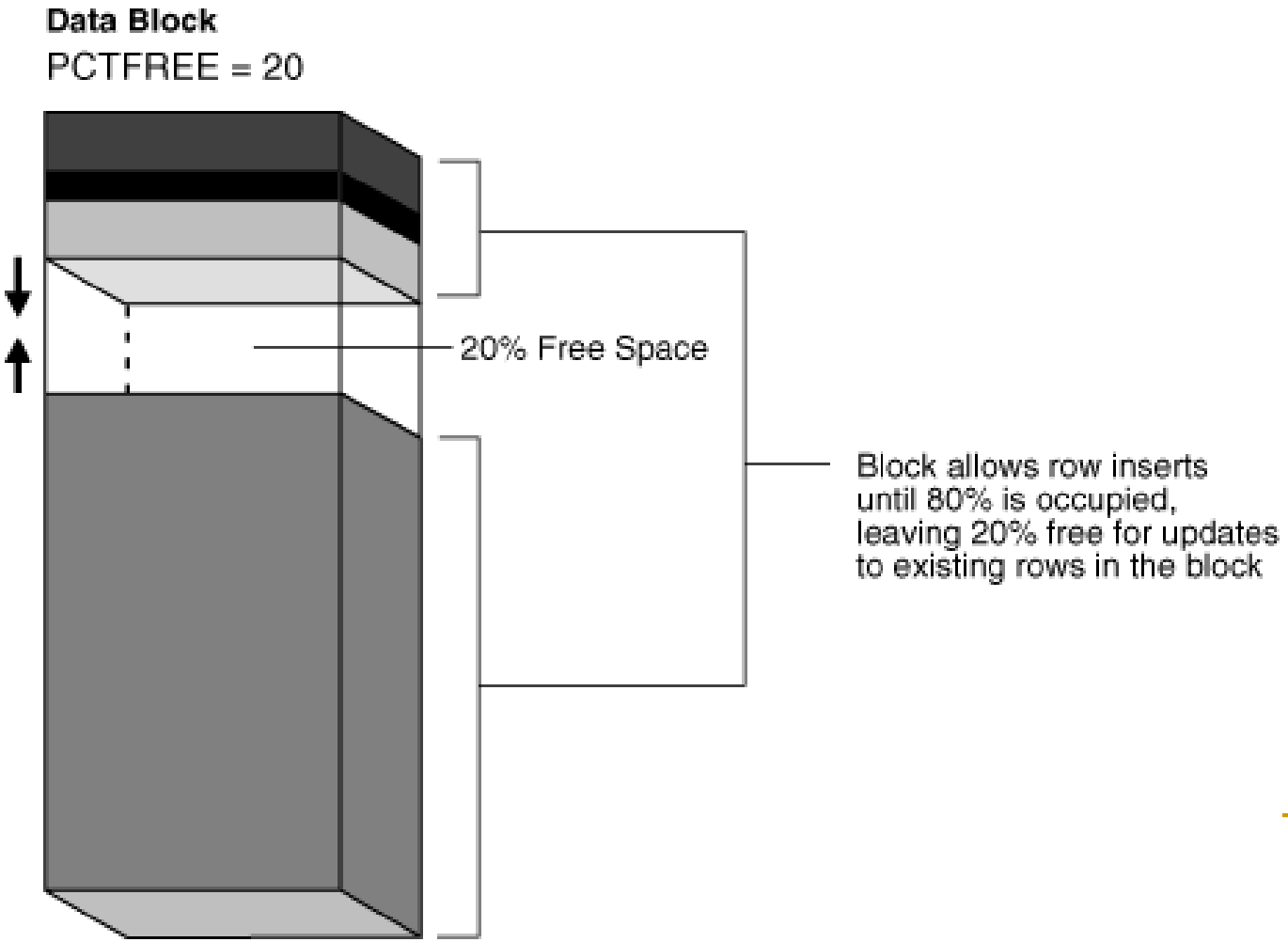
- A fejléc általános adatokat tárol, mint például a blokk címe, annak a szegmensnek a típusa (adat, index stb.), amihez tartozik stb.
- A tábla könyvtár arról a tábláról tartalmaz információkat, amelyeknek sorai a blokkban tárolódnak.
- A sor könyvtár a blokkban található sorokról tárol információkat, például a címüket. Ha törlődnek a blokk sorai, az Oracle nem szabadítja fel ezt a területet. Csak akkor hasznosítja újra, amikor új sorok kerülnek a blokkba.
- Az előző három részre angolul overhead-ként is hivatkoznak.
- A sor adat rész tartalmazza magukat a sorokat. A sorok „átlóghatnak” más adatblokkokba.
- A szabad területet módosításnál, illetve új sorok beszúrásánál használja a rendszer. A tranzakciók a sorok lock-olásánál szintén használják ezt a területet.

Sorok láncolása és vándorlása

- Ha egy sor nem fér be egy adatblokkba, például médiafájlok esetén, az Oracle a szegmenshez tartozó láncolt adatblokkokban tárolja az egyes részeket.
- Ha egy sor módosítás hatására túl nagyra nő, s már a szabad területen sem fér el, akkor a rendszer az egész sort egy új adatblokkba mozgatja. Ezt nevezik sorok vándorlásának (migrating).
- Az eredeti helyen csak a sor új helyének címe tárolódik. A vándorlás növelheti az I/O műveletek számát.

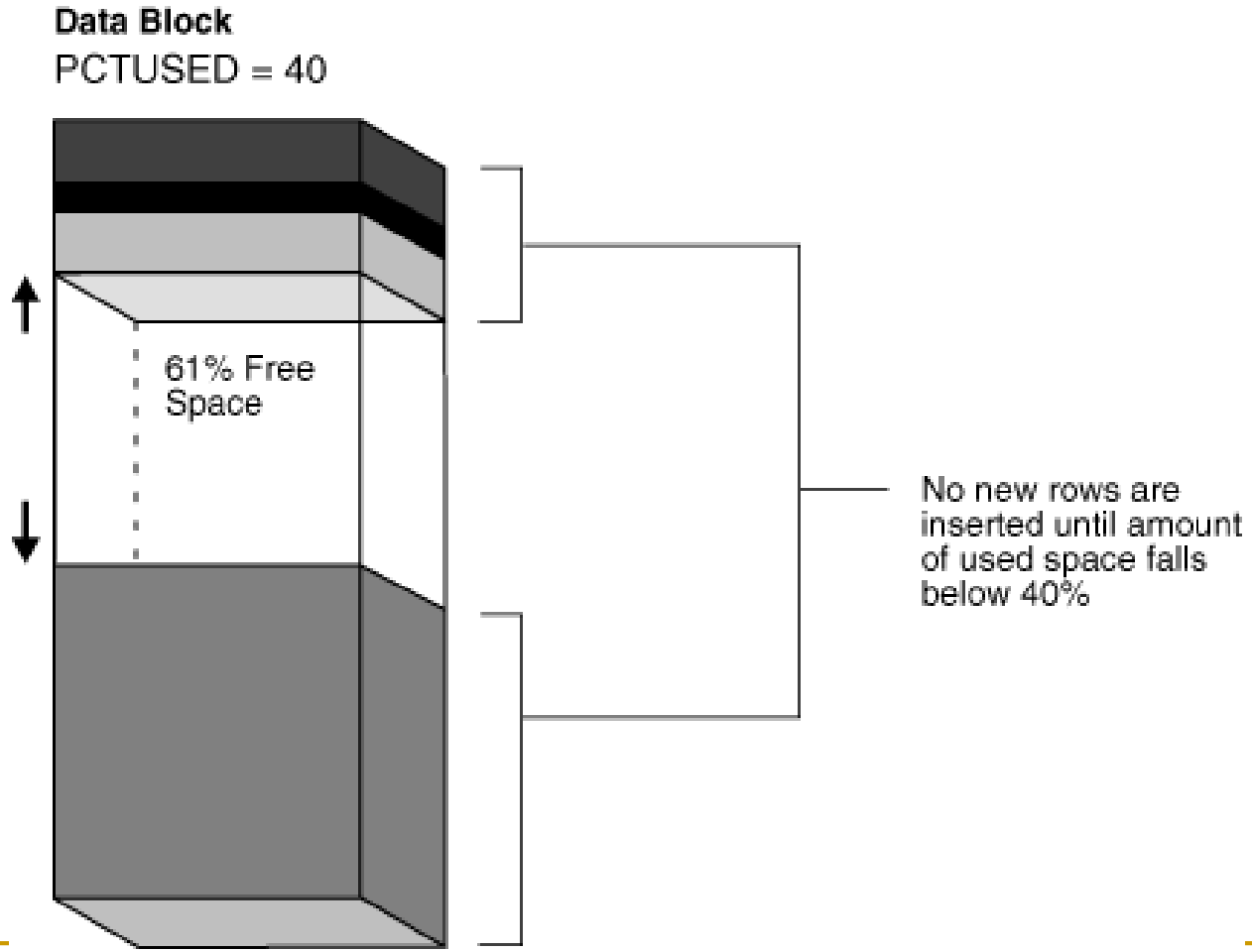
PCTFREE paraméter

Manuálisan felügyelt tablespaceknél két paramétert használhatunk az adatblokkok szabad területéhez történő hozzáférés vezérlésére PCTFREE : -vel beállíthatjuk, hogy az adatblokk hány százalékát akarjuk update-ekre fenntartani

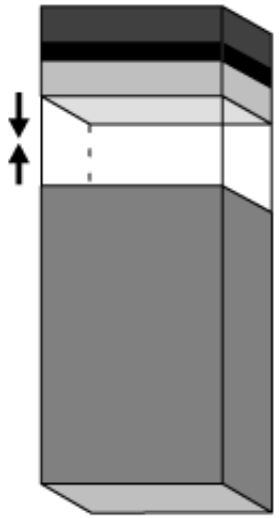


PCTUSED paraméter

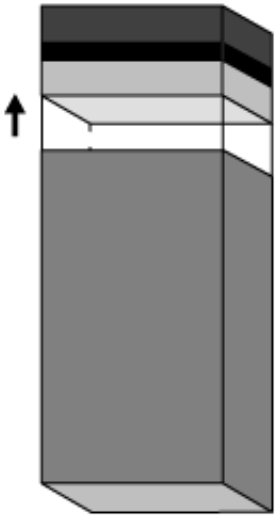
A *PCTUSED* paraméter egy minimum értéket ad a használt területre, amíg nem kezdeményezhetünk új instertet. Ha ez alá csökken a blokk foglaltsága, akkor szabadnak nyilvánítja a blokkot, és ismét enged beleírni



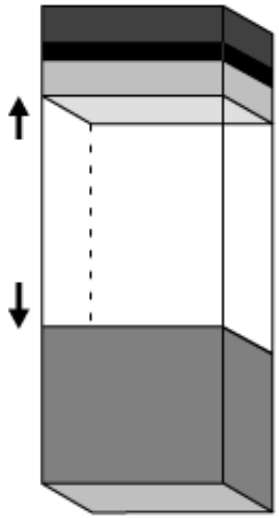
Data Block
PCTFREE = 20, PCTUSED = 40



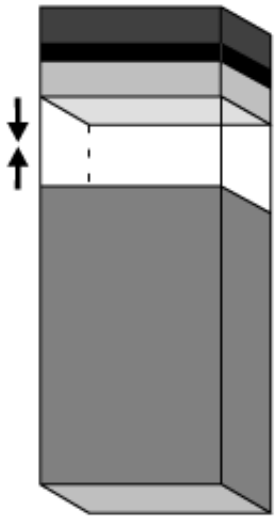
1 Rows are inserted up to 80% only, because PCTFREE specifies that 20% of the block must remain open for updates of existing rows.



2 Updates to exisiting rows use the free space reserved in the block. No new rows can be inserted into the block until the amount of used space is 39% or less.

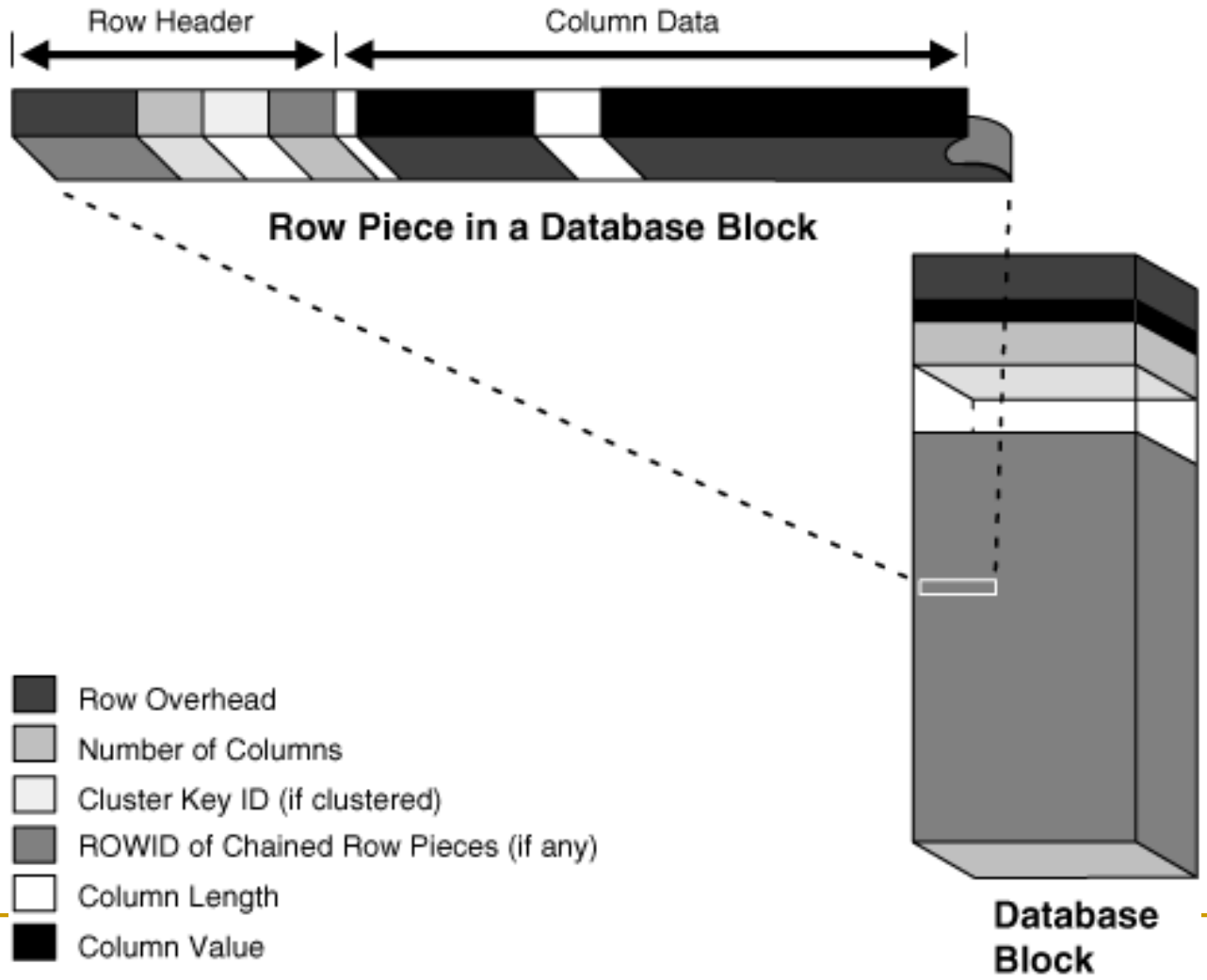


3 After the amount of used space falls below 40%, new rows can again be inserted into this block.



4 Rows are inserted up to 80% only, because PCTFREE specifies that 20% of the block must remain open for updates of existing rows. This cycle continues . . .

Sorok formátuma és mérete I.

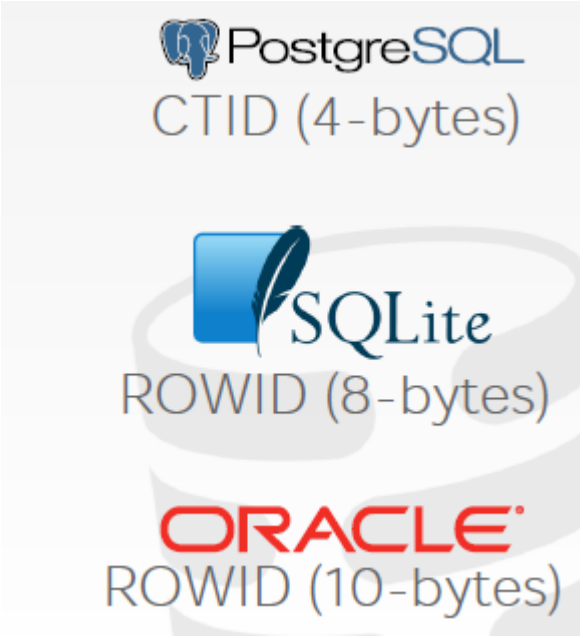


Sorok formátuma és mérete II.

- Egy sor legfeljebb 255 mezőt tartalmazhat. Ha ennél több mező szerepel, a rendszer a maradékot igyekszik ugyanabban a blokkban láncolva elhelyezni (infra-block chaining).
- Egy-egy sor fejléce a sorok oszlopairól, az egymáshoz láncolt sordarabokról (row pieces) – ha vannak –, klaszter esetén a klaszter kulcsáról tárol információkat.
- Ha a teljes sort tartalmazza a blokk, a fejléc legalább 3 byte hosszú.
- Ezek után minden egyes oszlopnál tárolódik az oszlop hossza (ha 255 byte-nál rövidebb 1 byte-on, ha hosszabb 3 byte-on) és a mező értéke.
- Ha változó hosszúságú adatról van szó, az adat által elfoglalt hely a változtatásoknak megfelelően módosul.
- NULL érték esetén az Oracle csak az oszlop hosszát tárolja (zero).
- A sor végén szereplő NULL értékek esetén még az oszlop hossza sem tárolódik. Ezt érdemes figyelembe venni a táblák attribútumainak felsorolásakor a CREATE utasításban.

Sorazonosítók

- A DBMS-nek szüksége van egy módszerre az egyes sorok nyomon követésére.
- Minden egyes sorhoz egyedi rekord-azonosító tartozik.
  - Leggyakoribb: page_id + eltolás / nyílás
  - Tartalmazhatja a fájlok helyinformációit is.
- Egy alkalmazás nem támaszkodhat ezekre az azonosítókra, hogy bármit is jelentenek.



ROWID I.

- A ROWID spec. típus, tulajdonképpen egy pointer, ami egy sornak a fizikai helyére mutat (melyik adatfájlban van, melyik blokkban, hányadik rekord).
- Egy sor mindaddig megőrzi azonosítóját, míg nem törlődik, ezért hasznos lehet SELECT, UPDATE, DELETE utasításokban. A ROWID-n történő hivatkozás a sorok elérésének leggyorsabb módja.
- Az indexekben a kulcsérték(ek) mellett szintén a ROWID tárolódik.
- Ez nincs a táblában tárolva, de mégis úgy viselkedik: pszeudo oszlop
  - **Példa:**
  - SELECT **rowid**, empno, ename, sal FROM emp;
- A kiterjesztett ROWID (extended ROWID) formátuma:

  OOOOOOFFFBBBBBBRRR, itt:
  - OOOOOO - az objektum azonosítója
  - FFF - fájl azonosítója (táblatéren belüli relatív sorszám)
  - BBBBBB - blokk azonosító (a fájlon belüli sorszám)
  - RRR - sor azonosító (a blokkon belüli sorszám).

ROWID II.

```
SELECT ROWID,
        SUBSTR(ROWID,1,6) ″OBJECT″,
        SUBSTR(ROWID,7,3) ″FILE″,
        SUBSTR(ROWID,10,6) ″BLOCK″,
        SUBSTR(ROWID,16,3) ″ROW„
 FROM ugyfel;
```

- A SUBSTR(ROWID,1,6) visszaadja a ROWID attribútum értékét az első karaktertől 6 karakter hosszúságban. A lekérdezés segítségével a különböző részeit különíthetjük el a ROWID-nek.

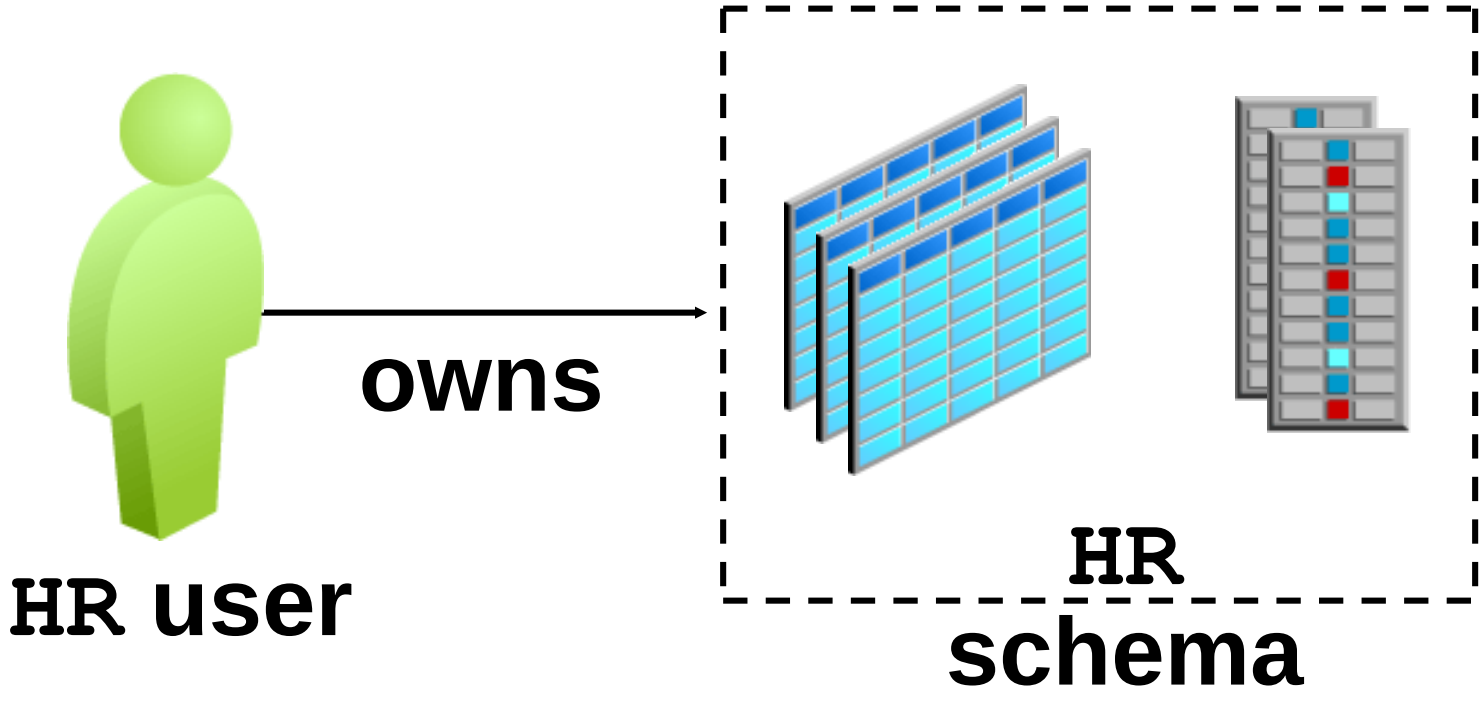
Extensek I.

- A táblák (table) jelentik az Oracle adatbázisban az alapvető adattárolási egységet.
- Tábla (vagy egyéb szegmens) létrehozásakor a rendszer egy kezdeti extenst (initial extent) allokál a tablespacen belül. Vezérlésére PCTFREE és PCTUSED paramétereket használunk. (Partícionált, Egymásba ágyazott, Ideiglenes, Külső táblák)
- Ha ez betelik, az Oracle újabb extenst allokál. A tárolási paraméterekhez (storage parameters) tartozó PCT_INCREASE paraméter segítségével adható meg, hogy hány százalékkal nőjön a következő extens mérete.
- A MINEXTENTS, MAXEXTENTS paraméterek a minimálisan lefoglalandó, illetve maximálisan lefoglalható extensek számát tárolják.
- A táblaterületek helykezelése történhet az adatszótárak segítségével, illetve lokálisan bitmap-ek felhasználásával. Az Oracle az utóbbit ajánlja nagyobb hatékonysága miatt.
- Általában egy szegmens extensei, ha felszabadulnak sem kerülnek vissza a közös táblatérbe, csak a szegmens törlése után.
- Klaszterek esetén, ha törlünk egyet az objektumok közül, a szegmenshez tartozó extensek még akkor sem „szabadulnak fel”.

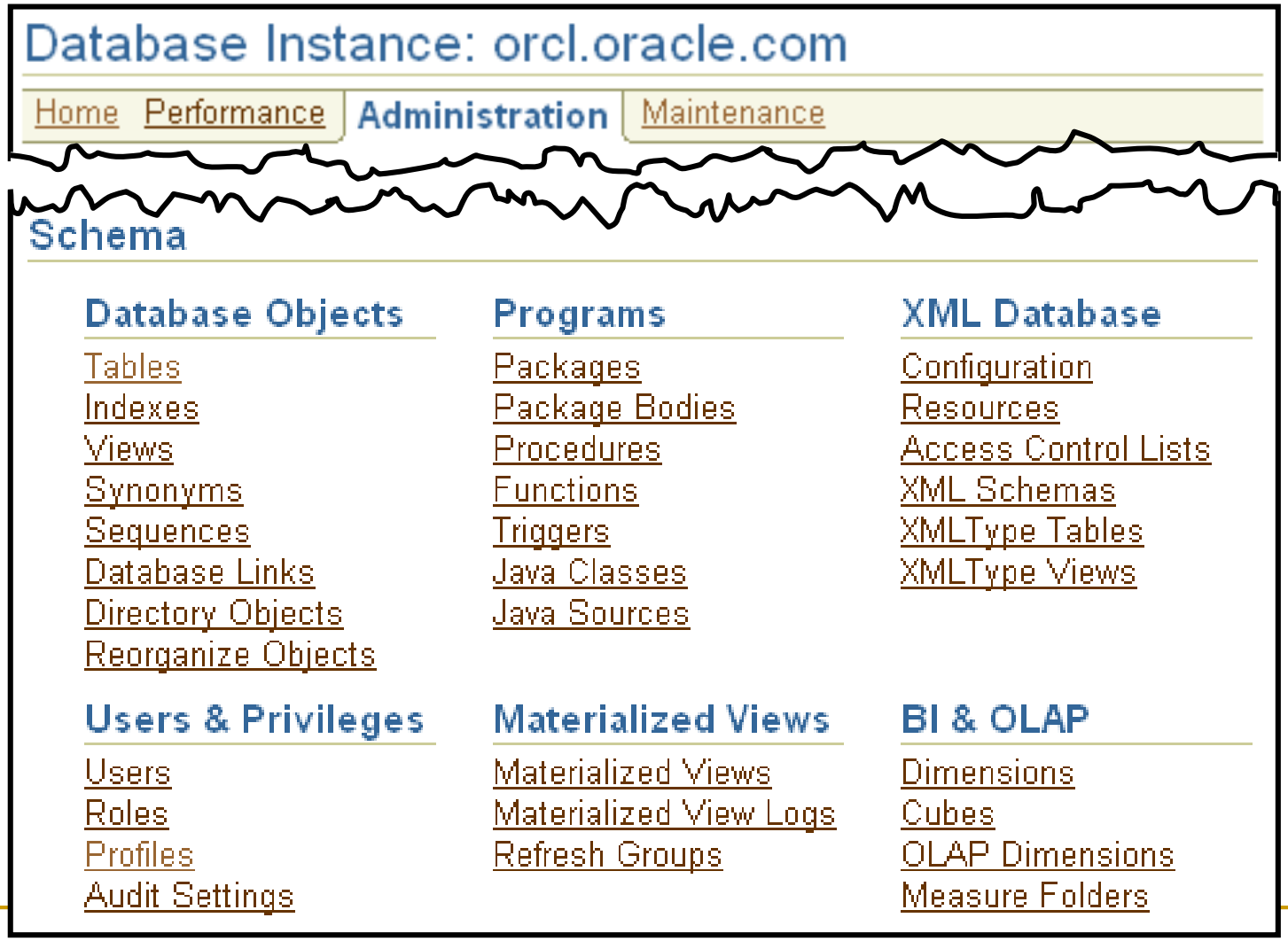
Extensek II.

- Temporális szegmensekben, ha az ideiglenes szegmens törlődik, a rendszer automatikusan visszaadja a szabad extenseket a táblaterületnek.
- A speciálisan rendezésre fenntartott szegmensek nem törlődnek egy-egy rendezés után.
- Adatszótárakon alapuló helykezelés esetén az Oracle a különböző extensekből felszabadult folytonos területeket nem vonja össze automatikusan, míg lokális helykezelés esetén igen.

Mi az a séma?



Séma objektumok elérése



Séma, objektumok

- Egy-egy séma (**Schema**) az egy-egy felhasználó tulajdonában lévő objektumokat tárolja, neve megegyezik a felhasználó nevével.
- Egy produkciós AB-ban, ez a felhasználó nem egy személyt reprezenál hanem **egy applikációt!!!**
- A sémabeli objektumokra <sema_nev>.<objektum_nev> formában lehet hivatkozni.
- Példa: scott.emp.
- A sémák és tablespacek között nincs semmilyen összefüggés: egy tablespace tartalmazhat objektumokat több különböző sémából, illetve egy séma objektumai is tárolódhatnak különböző tablespacekben
- Legfontosabb sémában található objektumok:
  - klaszterek; kényszerek; adatbázis hivatkozások; adatbázis triggerek; dimenziók; külső eljáráskönyvtárak; indexek és indextípusok; Java osztályok; materializált nézetek és a hozzájuk tartozó logok; objektum táblák, objektum típusok és objektum nézetek; operátorok; szekvenciák; tárolt függvények, eljárások és csomagok; szinonimák; táblák és indexelt táblák; nézetek.
- A séma és az objektumok a logikai szinthez tartozó fogalmak.

Az adatszótár (**data dictionary**)

- Egy-egy adatbázishoz tartozó adatszótár többek között a következő infomációkat tárolja:
  - az összes adatobjektum definíciója, az általuk felhasznált memória mérete, blokkok száma,
  - a felhasználók neve, jogosultságai,
  - melyik felhasználó módosította az egyes objektumokat stb.

- Az adatszótárhoz tartozó adatok alaptáblákban tárolódnak, az Oracle ezekben követi a változásokat, ám a felhasználók ritkán férnek hozzá ezekhez, mert a rendszer egy belső reprezentációt használ.

- Az adatokat az Oracle által létrehozott nézeteken keresztül lehet megtekinteni. Ezek általunk is olvasható formában jelenítik meg az alaptáblák adatait.

- Az alaptáblák tartalma a felhasználók által nem módosítható.

USER_, ALL_, DBA_ prefixek

- A felhasználói **USER**_ prefixű nézetekben a felhasználó csak a saját sémájához tartozó adatokat láthatja.
- A nézetek felépítése hasonló a többi nézetéhez, ám itt általában hiányzik az OWNER oszlop.

- Példa: USER_SEGMENTS.
- A kiterjesztett felhasználói **ALL**_ prefixű nézeteken keresztül mindazon adatok megtekinthetők, amelyekhez az aktuális felhasználónak hozzáférése van.
- Az adatbázis adminisztrátori **DBA**_ prefixű nézetek az egész adatbázisról adnak átfogó képet.

SYS és SYSTEM felhasználó

- Mindkét felhasználó automatikusan létrejön az adatbázis létrehozásakor, mindkettő adatbázis-adminisztrátori jogkörrel.
- Minden adatszótárhoz tartozó alaptábla és nézet a SYS felhasználó birtokában van a SYS nevű sémán. Döntő fontosságú annak biztosítása, hogy ezekhez más ne férhessen hozzá.
- A SYSTEM felhasználó további adminisztrációhoz szükséges táblákat és nézeteket definiálhat, melyeket az Oracle különböző szolgáltatásai használhatnak.

- CREATE TABLE tipus_proba(...)
- TABLESPACE users
- PCTUSED 50 PCTFREE 20 INITRANS 1 MAXTRANS 255
- STORAGE (INITIAL 32K MINEXTENTS 1 MAXEXTENTS 200 PCTINCREASE 0
- FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT);

- STORAGE(...): hogyan viselkedjenek az extensek, mekkorák legyenek, hogyan bővüljenek stb.
- - INITIAL: első extens mérete
- - NEXT: következő extens mérete
- - PCTINCREASE: milyen mértékben növekedjenek az extensek (%-ban) az előzőhöz képest. (50 azt jelenti, hogy másfélszerese lesz a következő)
- - MINEXTENTS: minimális extens darabszám (a tábla létrehozásakor ennyit automatikusan létrehoz)
- - MAXEXTENTS: maximális extens darabszám
- stb.

Feladatok I.

- Adattárolással kapcsolatos fogalmak (DBA_TABLES, DBA_DATA_FILES, DBA_TABLESPACES, DBA_SEGMENTS, DBA_EXTENTS, DBA_FREE_SPACE):

1. Adjuk meg az adatbázishoz tartozó adatfájlok nevét és méretét méret szerint csökkenő sorrendben.
2. Adjuk meg, hogy milyen táblaterek lettek létrehozva az adatbázisban, az egyes táblaterek hány adatfájlból állnak és mekkora az összméretük. (tablater_nev, fajlok_szama, osszmeret)
3. Mekkora a blokkok mérete a USERS táblatéren?
4. Melyik a legnagyobb méretű tábla szegmens az adatbázisban és hány extensből áll? (A particionált táblákat most ne vegyük figyelembe.)
5. Melyik a legnagyobb méretű index szegmens az adatbázisban és hány blokkból áll? (A particionált indexeket most ne vegyük figyelembe.)

Feladatok II.

6. Melyik a legnagyobb méretű LOB szegmens az adatbázisban és mekkora az első extensének mérete?
7. Melyik a legnagyobb méretű tábla partíció az adatbázisban és mekkora lesz a következő extensének a mérete?
8. Adjuk meg adatfájlonként, hogy az egyes adatfájlokban mennyi hely lett lefoglalva összesen.
9. Melyik felhasználó objektumai foglalnak összesen a legtöbb helyet az adatbázisban?
10. Van-e a NIKOVITS felhasználónak olyan táblája, amelyik több adatfájlban is foglal helyet?
11. Melyik táblatéren van az ORAUSER felhasználó dolgozo táblája?

Feladatok III.

- A táblák oszlopai (DBA_TAB_COLUMNS):

12. Adjuk meg azoknak a tábláknak a tulajdonosát és nevét, amelyeknek van 'Z' betűvel kezdődő oszlopa.
13. Adjuk meg azoknak a tábláknak a nevét, amelyeknek legalább 8 darab dátum típusú oszlopa van.
14. Adjuk meg azoknak a tábláknak a nevét, amelyeknek 1. és 4. oszlopa is VARCHAR2 típusú.

Feladatok IV.

- ROWID adattípus formátuma és jelentése:

- A ROWID megjelenítéskor 64-es alapú kódolásban jelenik meg. Az egyes számoknak (0-63) a következő karakterek felelnek meg: A-Z -> (0-25), a-z -> (26-51), 0-9 -> (52-61), '+' -> (62), '/' -> (63)
- Pl. 'AAAAAB' -> 000001

15. Írjunk PL/SQL függvényt, ami a fenti 64-es kódolásnak megfelelő számot adja vissza. A függvény paramétere egy karakterlánc, eredménye pedig a kódolt numerikus érték legyen. (Elég ha a függvény maximum 6 hosszú, helyesen kódolt karakterláncokra működik, hosszabb karakterláncra, vagy rosszul kódolt paraméterre adjon vissza -1-et.)
16. Ennek a fv-nek a segitségével adjuk meg egy táblabeli sor pontos fizikai elhelyezkedeset. (Melyik fájl, melyik blokk, melyik sora.) Például az ugyfel tábla azon sorára, ahol az ügyfél neve ‘MELAK'.

Feladatok V.

17. A NIKOVITS felhasználó CIKK táblájának adatai hány blokkban helyezkednek el?
18. Az egyes blokkokban hány sor van?

Feladatok VI.

- Adatbázis objektumok (DBA_OBJECTS):

19. Kinek a tulajdonában van a DBA_TABLES nézet illetve a DUAL tábla?
20. Kinek a tulajdonában van a DBA_TABLES illetve a DUAL szinonima?
21. Milyen típusú objektumai vannak az ORAUSER nevű felhasználónak az adatbázisban?
22. Hány különböző típusú objektum van nyilvántartva az adatbázisban? Melyek ezek? (két külön lekérdezés)
23. Kik azok a felhasználók, akiknek több mint 10-féle objektumuk van?
24. Kik azok a felhasználók, akiknek van triggere és nézete is?
25. Kik azok a felhasználók, akiknek van nézete, de nincs triggere?
26. Kik azok a felhasználók, akiknek több mint 40 táblájuk, de maximum 37 indexük van?

Felhasznált irodalom

- https://15445.courses.cs.cmu.edu/fall2019/