# Elmélet

## Indexek

A lekérdezések optimalizálására / gyorsítására szolgáló adatstruktúrák.
(Ha nem kell az egész tábla, akkor csak az indexeket kell lekérdezni, így gyorsabb a lekérdezés).

Az index lehet:

- **Unique** - Ha a levél elemekben lévő kulcsok egyediek (unique / primary key az oszlop a táblában).
- **Nem unique** - Egy érték többször is előfordul az oszlopban.

- 2 fajtája van:
    - **B-trees**
        - A lekérdezés gyorsabb, de a tárolás lassabb.
        - Fa struktújájú adat halmaz, gyökér elem fent van, levél elemben sorrendbe rendezve vannak az értékek (by default növekvőben))
        - összekötés van a levélelemek között □ a lekérdezés tud ugrálni a levélelemek között
    - **Bitmap index**:
        - Kulcs érték található a 2 ROWID-ban (Start ROWID, End ROWID) és egy bitmap sorozattal (melyik sor kell az adatokból)
        - Amikor sok az ismétlődő elem, akkor érdemes ezt használni.
        - Jól tudja támogatni, ahol aggregációk történnek (pl. összesítés, stb.)

Index előnye: Ha egy oszlopra felépül akkor tudja gyorsítani a keresést, mert nem kell a teljes táblát lekérdezni, csak az indexet.

Ha a tábla mellé van index struktúránk, akkor a táblát karban kell tartani + tárolási többletköltsége van.

## Adatszótár

Metaadatok az adatszótárban (data dictionary: központi, csak olvasható táblák és nézetek) tárolódnak.

Data dictionary szerkezete:

- **Base tables**
- **User-Accessible Views**

Data dictionary típusai:

- **DBA** - Data Base Administrator (az adminisztrátor tudja olvasni és írni)
- **ALL** - Amihez jogosultság van
- **USER** - azokról az objektumokról, aminek én vagyok a tulajdonosa

## Adatbázis architektúra

3 nyelv, amit megért:

- **SQL** - Dekleratív nyelv, nem mondja meg, hogy hogyan kell megoldani, csak azt, hogy mit kell megoldani.
- **PL/SQL és Java** - Imperatív nyelv, megmondja, hogy hogyan kell megoldani.

Architektúra:

- Amikor a kliens kiadja az első utasítást akkor létrejön egy *user process*, ami kommunikál a *server process*-sel.
  Van egy listener a kettő között, ami figyeli mindig, hogy van-e bejövő kérés.
  Ez egy munkafolyamat (session), aminek vége, ha a felhasználó kilép vagy a kapcsolat megszakad vagy ...
- Adatbázis szerver több Instance (szolgáltatásból) épül fel.
  - Memory Structure (System Global Area)
  - Process Structure (Process Control Block)
- Adatbázis alkotó fájlok struktúrája (fizikai rész) Storage Structures

**Egy adatbázis példány mindig egy adatbázishoz kötődik.**

Lehet, hogy több adatbázis példány egy adatbázishoz kötődik.

Egy példány (Instance) nem kötődhet több adatbázishoz (Database-hez).

Kliens kapcsolódása az adatbázishoz:

- Van az adatbázis szerver, oda belép a felhasználó a szerver operációs rendszerén keresztül.
- Külön gépen van a kliens és külön gépen a szerver és a hálózaton keresztül kapcsolódik a kettő.
- Háromrétegű kapcsolaton keresztül- a programon keresztül lérjük le, ami kapcsolódik az alkalmazás szerveréhez, ami lekérdezi az adatbázist.

Az Oracle struktúra három fő csoportra osztható:

- **Tároló struktúrák**
- **Memória struktúrák**
- **Szerverfolyamatok**

## SGA

Database buffer cache - A beolvasott adadtokat tárolja, hogy ne kelljen mindig a lemezről olvasni.(Memória)

Redo log buffer - Memóriából a módosított adat bekerül ide naplózásra, utána a háttértárra kiírásra kerül.

Shared pool:

- **Library** cache - Shared SQL Area - Követi a folyamatot.
  (Beérkezett SQL utasításokat tárolja, átalakítja).
  **A végrehajtási terv itt tárolódik.**
- **Data dictionary cache** - Adatszótár, leggyakrabban kiolvasott adatokat tárolja.
- **Server result cache** - A lekérdezés eredményeit tárolja.

## PGA

- private SQL area - A folyamatoknak saját területük van, ahol a végrehajtási terv tárolódik.
- Session memory - A folyamatoknak saját területük van, ahol a végrehajtási terv tárolódik.
  Az adott session-höz tartozó ideiglenes adatok találhatóak meg.

## Adatbázis transzakciók

Összetartozó DML (módosító) utasítások sorozata

Van egy eleje és egy vége. Feltételezzók, hogy a tranzakció kezdetekor az adatok megfelelnek az elvárásainknak és a végén is van egy konzisztens állapot.
Közte lehet egy nem konzisztens állapot.

Egy egységként értelmezzük, konzisztens állapot között mozgunk.

ACID:

- Atomicity - Atomosság - A tranzakció egységesen hajtódik végre, vagy teljesen, vagy semmiképpen.
- Consistency - Konzisztencia - A tranzakció végén a konzisztens állapotban kell lennie.
- Izolation - Izoláció - Azt várja el, hogy amikor pl 2 tranzakció lefutott, akkor utána úgy nézzen ki az adatbázis állapota, mintha külön-külön futott volna.
  (Tud várakoztatni, eldönti mely tranzakciókat fésülhet össze.)
- Durability - Tartóság - Lefutott egy tranzakció, sikeres volt, ...

# Gyakorlat

## OLTP

Szervezet mindennapos működéséhez tartozó folyamatok (pl boltban való fizetés = módosítási kérések), ezekből sok van, gyakran használt műveletek, általában kevés adatot érintenek.

## DSS/DWW

- Egy napban csak egyszer töltünk be adatokat, de akkor nagyon sokat. Több sort, táblát, nagyon sok adatokat érintünk ilyenkor.
- Lekérdezések: Nem teszünk fel sok lekérdezést.
- Olvasás és adatok kombinálása jelenti a fő feladatot.

Aggregáció(s művelet): Több sort érint, több művelet, több adatból állítjuk elő a választ.