KORSZERU ADATBAZISOK

EA ZH / ELOVIZSGA

KERDESEK

Tartalom

Relációs adatbázisok, optimalizálás

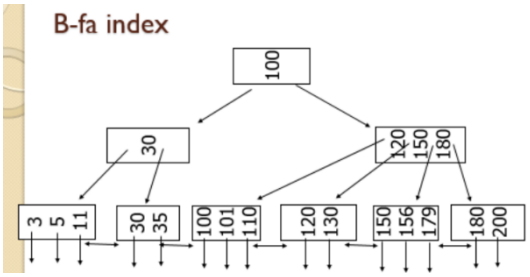
1. B-fa, bitmap indexek szerkezetének és használatának ismerete

Indexek használata:

- keresést gyorsító segédstruktúra,
- több mezőre is lehet indexet készíteni,
- az index tárolása növeli a tárméretet,
- nem csak a főfájlt, hanem az indexet is karban kell tartani, ami plusz költséget jelent,
- ha a keresési kulcs egyik indexmezővel sem esik egybe, akkor kupac szervezést jelent.

B-fa index:

A hagyományos index B-fa szerkezetű, a fa leveleiben vannak a bejegyzések, és mellettük a sorazonosító (ROWID), amit mutatóval is jelölhetünk. Az index lehet egy vagy többoszlopos.



Ha egy érték többször szerepel a táblában, akkor az indexben is többször fog szerepelni, minden sorazonosítóval külön-külön. A levélblokkok mindkét irányban láncolva vannak, így növekvő és csökkenő keresésre is használható az index.
(pl. WHERE o > x vagy WHERE o < y)

Bitmap index:

Hasonló a B-fa indexhez, de a levelekben a kulcsérték mellett nem a ROWID-k tárolódnak, hanem egy bittérkép. (Az első és utolsó érintett ROWID valamint a köztük lévő sorokra vonatkozó bittérkép.) Minden sornak egy bit felel meg, ami azokra a sorokra lesz 1-es, amelyek az adott értéket tartalmazzák.

Bitmap (bittérkép) index

Empno	Status	Region	Gender	Info
101	single	east	male	bracket_1
102	married	central	female	bracket_4
103	married	west	female	bracket_2
104	divorced	west	male	bracket_4
105	single	central	female	bracket_2
106	married	central	female	bracket_3

REGION='east'	REGION='central'	REGION='west'
1	0	0
0	1	0
0	0	1
0	0	1
0	1	0
0	1	0

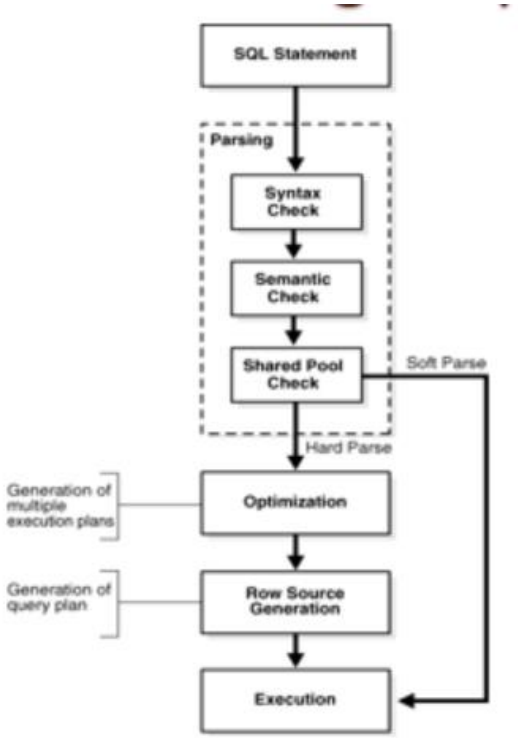
2. Adatszótár (data dictionary): mit tárol, kinek a sémájában van, adatszótár nézetek és prefixei

- Oracle adatbázis-kezelő esetén a metaadatok az adatszótárban (data dictionary) tárolódnak *(Data dictionary : központi, csak olvasható referencia táblák és nézetek)*
- Minden base table és user-accessible view a SYS sémájában van
- adatszótár nézetek:

Data Dictionary Views

	Who Can Query	Contents	Subset of	Notes
DBA_	DBA	Everything	N/A	May have additional columns meant for DBA use only
ALL_	Everyone	Everything that the user has privileges to see	DBA_ views	Includes user's own objects and other objects the user has been granted privileges to see
USER_	Everyone	Everything that the user owns	ALL_ views	Is usually the same as ALL_ except for the missing OWNER column (Some views have abbreviated names as PUBLIC synonyms.)

3. **SQL lekérdezés végrehajtásának folyamatát leíró ábra ismerete, fel kell tudni rajzolni, lépéseit érteni kell**
4. **Következő fogalmak ismerete és magyarázata 1-2 mondatban: shared pool check, access path, CBO, költség, row source, row source tree**



Syntax Check:

- a lekérdezés szintaktikailag hibátlan-e

szintaktikai hiba pl. a hiányzó vessző, elírt kulcsszó, elfelejtett idézőjel, stb.

Semantic Check:

- a lekérdezés „értelmes-e”

szemantikai hiba pl. nem létező táblára vagy változóra hivatkozás, adattípusok nem stimmelnek

Shared Pool Check

- Végrehajtottuk-e már ezt korábban?
  - Hash érték generálás (SQL ID)
  - Összehasonlítás a korábban futtatott utasítások hash értékeivel
  - Ha van találat:
    - szemantikai ellenőrzés
    - környezet ellenőrzés

Optimization

- Optimális végrehajtás megtervezése

Execution plan (végrehajtási terv):

- hogyan hajtjuk végre az utasítást
  - CBO (Cost-Based Optimization)
    - Cost (költség) = **adott környezetben** a művelet becsült erőforrásigényét reprezentáló számérték
    - több lehetséges végrehajtási terv közül a legkisebb költségűt választja ki
    - statisztikák alapján becsül

Row Source Generation

- Az optimális végrehajtási terv "lefordítása" az SQL motor által végrehajtható lépések sorozatára.
- Row source: rekordok halmaza
  - pl. táblák, nézetek; JOIN vagy GROUPING műveletek eredménye
  - az egyes végrehajtási lépések bemenetei és kimenetei ilyen rekord halmazok

Row Source Tree

- A Row Source Generator által előállított fastruktúra, amely megmutatja:
  - a táblák sorrendjét
  - a hozzáférés módját az egyes táblákhoz
  - a táblák összekapcsolásának módját (ha van a lekérdezésben join)
  - egyéb műveleteket (szűrés, rendezés, csoportosítás)

Access Path

- A hozzáférés módja: hogyan olvasunk ki egy row source-t
- Unáris művelet
- Bemenete és kimenete is egyetlen row source
  - Ellentétben pl. a Join művelettel, amelynek a bemenete két row source
- Különféle adatstruktúrákra különböző hozzáférési módok

5. Access path típusok ismerete

Heap-organized tables

- Full Table Scans, Table Access by Rowid, Sample Table Scans

B-Tree indexes

- Index Unique Scans, Index Range Scans, Index Full Scans,Index Fast Full Scans, Index Skip Scans, Index Join Scans

Bitmap indexes

- Bitmap Index Single Value, Bitmap Index Range Scans, Bitmap Merge

Table clusters

- Cluster Scans, Hash scans

6. Join végrehajtásának típusai (működésük és választásuk)

Meg kell határozni:

- Access path
  - Minden row source-ra külön-külön
- Join módszer
  - Row source páronként
- Join típus
  - Inner, outer, equijoin, antijoin, semijoin
    - Join sorrend
  - Kettőnél több tábla esetén

Cél: minél kevesebb rekorddal kelljen dolgozni

- A CBO arra törekszik a módszer és a sorrend megválasztásánál, hogy minél korábbi lépésben lecsökkenjen a sorok száma.

◦ Pl. Ha vannak a táblák között olyanok, amelyeket összekapcsolva egyetlen sort kapunk, akkor ezt végzi el először.

Join módszerek:

- Nested loop join
- Sort-merge join
- Hash join
- (Cartesian join)

7. Statisztika szerepe és tartalma, hisztogramok, index clustering factor

Statisztikák:

- Táblákról (pl. sorok száma, blokkok száma, stb.)
- Oszlopokról (pl. különböző értékek száma (NDV) egy oszlopban, NULL-ok száma, adatok eloszlása, stb.)
- Indexekről (levél blokkok száma, famagasság, stb.)
- Rendszerről (I/O, CPU teljesítmény és kihasználtság)
- Hisztogramok:
- A CBO alapértelmezetten egyenletes eloszlásúnak feltételezi az adatokat
  - azaz az előforduló különféle értékekből kb ugyanannyi van egy oszlopban
    - Ha ez nem így van, akkor a join és szűrő kifejezések becslése nagyon pontatlan lehet.
    - Kb. melyik érték milyen gyakori egy oszlopban.
    - Segít a pontos kardinalitás-becslésben
  - kardinalitás=hány sorból fog állni az adott művelet eredményeként kapott row source

Index clustering factor:
Egy B-fa index esetén, a Index Clustering Factor méri a sorok fizikai csoportosulását az idex értékéhez mérten. Segít az optimalizálónak eldönteni, hogy egy index scan vagy egy full table scan lenne e hatékonyabb az adott lekérdezésnél. Alacsony clustering factor index scan hatékonyságára utal.

8. Hintek szerepe, általános szintaktikája, 5 darabot tudni a jelentésével együtt

Hintek:

- „Tippek” az optimalizálónak
- Ezek segítségével kényszeríthetünk ki
  - access path választást
  - join módszer választást
  - join sorrend választást
  - stb.
- Nem ajánlott ész nélkül használni!
  - A hatékonysága nagyon függ a környezettől.
  - Inkább csak tesztelésre használjuk.

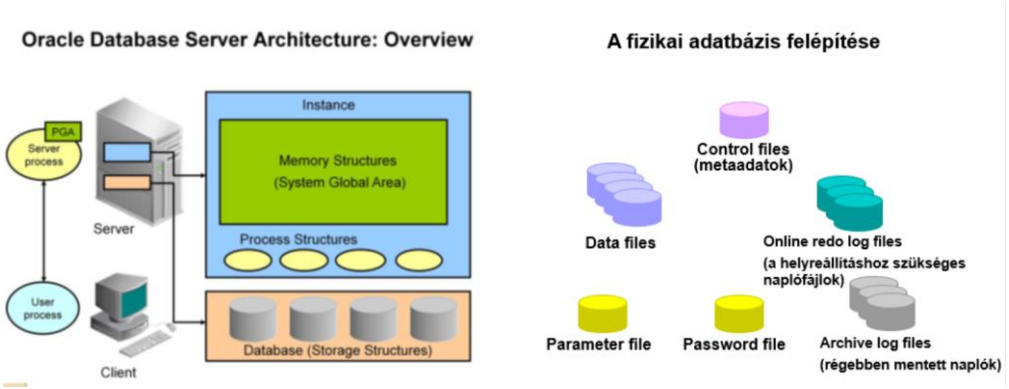
- Szintaktikailag: speciális komment
  - SELECT /*+ hint_text */ ...
- A hintet a SELECT, UPDATE, INSERT, MERGE, vagy DELETE után írjuk.
- A + közvetlenül a komment jel után következzen, nem szabad szóközt tenni közé.
- Csak egyetlen hintet tartalmazó komment lehet egy kifejezésben, de az tartalmazhat több, szóközzel elválasztott hintet.
- A szintaktikai hibás hintet figyelmen kívül hagyja a rendszer.

- Leggyakoribb hintek:
  - **INDEX**
    Index scan a megadott tábla megadott indexén INDEX(tábla indexnév)
  - **INDEX_DESC**
    csökkenő sorrendű index scan
  - **INDEX_FFS**
    index fast full scan
  - **INDEX_SS**
    index skip scan
  - **INDEX_JOIN**
    index join scan a megadott indexek összekapcsolásával
    INDEX_JOIN(tábla indexnév1 indexnév2)
  - **FULL**
    full table scan a megadott táblára

Oracle database server architecture

9. A következő fogalmakat ismerni és ismertetni kell tudni:
Oracle AB 2 fő része: adatbázis (vagyis a fizikai struktúrák rendszere), példány (instance), vezérlő fájl (control file), helyrehozó napló fájlok (redo log files), adatfájlok, Paraméterfájl, Jelszófájl, System Global Area (SGA) terület, szerverfolyamatok, 69. dia ábra, PGA

69. dia ábra ->



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 a rendszergazdák (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.

A vezérlő fájlok (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.

Táblaterek (tablespaces) és adatfájlok:

- Minden adatbázis logikailag egy vagy több táblatérre van felosztva.
- A táblaterek egy vagy több fájlból állnak.
- Az adatfájlok mindig csak egy táblatérhez tartoznak.
- A táblatér mérete szerint kétféle lehet:
  - nagy fájlból álló táblatér (bigfile tablespace): ez egyetlen fájl, de 4G blokkot tartalmazhat
  - kis fájlokból álló táblatér (small file tablespace): több kisebb fájlból áll

System Global Area (SGA):

- Egy Oracle példányhoz tartozó Oracle memóriaszerkezet a következő részekből áll:
  - System Global Area (SGA): Az összes szerverfolyamat és háttérfolyamat osztozik rajta
  - Program Global Area (PGA): Minden szerverfolyamatnak és háttérfolyamatnak saját memóriaterülete (PGA-ja) is van.
- Az SGA a példányhoz tartozó adatokat és vezérlő információkat is tartalmazhat.
- Az SGA a következő adatszerkezetből áll:
  - Database buffer cache: A beolvasott adatblokkok pufferterülete
  - Redo log buffer: A helyreállításhoz szükséges redo napló pufferterülete, innen íródik ki a napló a lemezen tárolt redo naplófájlokba
  - Shared pool: A felhasználók által használható közös puffer
  - Large pool: A nagyon nagy Input/Output igények kielégítéséhez használható puffer
  - Java pool: A Java Virtuális Gép (JVM) Java kódjaihoz és adataihoz használható puffer
  - Streams pool: Az Oracle Stream-ek pufferterülete
- Az SGA dinamikus, azaz a pufferek mérete szükség esetén a példány leállítása nélkül változtatható.

Program Global Area (PGA):

- A Program Global Area (PGA) a szerverfolyamatok számára van fenntartva.
- A PGA mérete és tartalma attól függ, hogy a példány osztott módra (shared server mode) konfiguráltuk-e.
- A PGA általában a következőkből áll:
  - Private SQL area: Futási időben szükséges memóriaszerkezeteket, adatokat, hozzárendelési információkat tartalmaz. Minden olyan munkaszakasz (session), amely kiad egy SQL utasítást, lefoglal egy saját SQL területet.
  - Session memory: A munkaszakaszhoz (session) tartozó információk, változók tárolásához szükséges terület**.**

Az adatbázishoz nem tartozó fájlok:

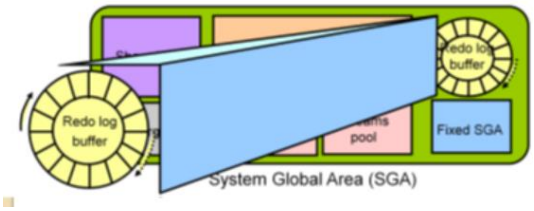
- A **paraméterfájl** az Oracle példány jellemzőit határozza meg, például az SGA memóriarészeinek méretét.
- A **jelszófájl**ból derül ki, hogy melyek azok a felhasználók, akik elindíthatnak vagy lekapcsolhatnak egy Oracle példányt.
- Az archivált REDO naplófájlok a naplófájl másolatai, amelyeket lemezhiba után lehet helyreállításhoz használni

10. Oracle memória szerkezete: SGA (ismerni: database buffer cache, redo log buffer, shared pool), PGA

<u>database buffer cache: adatbázis puffer</u>

- Az utoljára beolvasott blokkokat tárolja.
- A mérete a DB_BLOCK_SIZE inicializáló paramétertől függ.
- A pufferek számát a DB_BLOCK_BUFFERS adja meg.
- Amikor egy lekérdezést kell végrehajtani, a szerverfolyamat először megnézi, hogy a keresett blokk nincs-e itt. Ha nem találja a pufferben, csak akkor olvassa be a blokk másolatát az adatfájlból.
- Ha már nincs hely a pufferben, akkor a legrégebben használt adatblokk helyére olvassa be az újat.

<u>redo log buffer:</u>



- *is a circular buffer in the SGA*
- *holds information about changes made to the database*
- Ha már nincs hely a pufferben, akkor a legrégebben használt adatblokk helyére olvassa be az újat.
- *contains redo entries that have the information on redo changes made by operations such as DML and DLL*

<u>shared pool: közös SQL puffer</u>

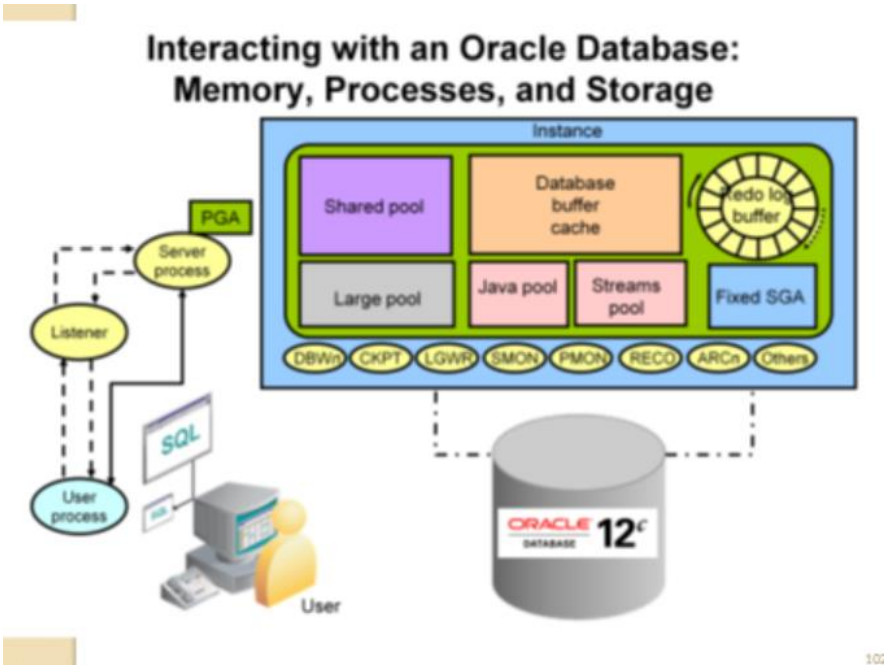
- A könyvtár (library) cache az SQL utasítás szövegét, elemzett (parsed) kódját, és végrehajtási tervét tartalmazza.
- Az adatszótár (data dictionary) cache a táblák, oszlopok és egyéb objektumok definícióját, jogosultságait tartalmazza.
- A méretét a SHARED_POOL_SIZE inicializáló paraméter állítja be.

<u>PGA (Program Global Area): kizárólagos memóriafelületek</u>

- Nincs megosztva
- Csak a szerverfolyamat írhatja
- A következőket tartalmazza:
  - Rendezési terület (sort area)
  - Információ a munkaszakaszról (session):
    - jogosultságok, statisztikák
  - A kurzorok állapota (cursor state)
  - A munkaszakaszhoz tartozó változók (Stack space)
- A szerver folyamat indulásakor foglalja le ezt a területet, befejezéskor pedig felszabadítja

11. Háttérfolyamatok (DBWn, LGWR ismerete), 102. dia ábra

- Amikor egy alkalmazás vagy Oracle eszköz (mint például az Enterprise Manager) elindul, akkor az Oracle szerver elindít egy szerverfolyamatot, amely lehetővé teszi az alkalmazás utasításainak végrehajtását.
- Az Oracle egy példány indításakor háttérfolyamatokat is elindít, amelyek kommunikálnak egymással és az operációs rendszerrel.
- A háttérfolyamatok kezelik a memóriát, puffereket, végrehajtják az írási, olvasási műveleteket a lemezen, karbantartásokat végeznek.
- A legfontosabb háttérfolyamatok a következők:
  - Database writer (**DBW*n***): Az adatpufferből kiírja lemezre, egy fájlba a módosított blokkokat
  - Log writer (**LGWR**): A REDO napló bejegyzéseit írja ki a lemezre
  - System monitor (SMON): Katasztrófa utáni indításkor elvégzi a helyreállítást
  - Process monitor (PMON): Ha egy felhasználói folyamat megszakad, akkor elvégzi a szükséges takarítást, karbantartást
  - Checkpoint (CKPT): Ellenőrzési pontok esetén elindítja a DBW*n* folyamatokat és frissíti az adatbázis összes adatállományát és vezérlő állámányát
  - Archiver (ARC*n*): A REDO napló állomány másolatait a mentésre kijelölt lemezekre írja ki, mikor egy naplófájl betelik vagy a következő online redo naplóba folytatódik az írás (log switch)

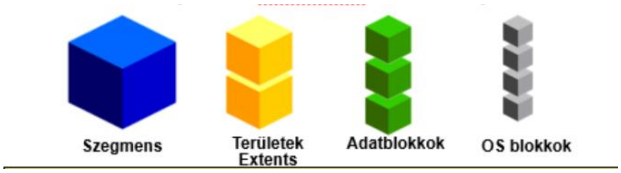


12. Táblatér, szegmens, extents, adat block

Táblatér:

- Minden adatbázis logikailag egy vagy több táblatérre van felosztva.
- A táblaterek egy vagy több fájlból állnak.
- Az adatfájlok mindig csak egy táblatérhez tartoznak.
- A táblatér mérete szerint kétféle lehet:
  - nagy fájlból álló táblatér (big file tablespace): ez egyetlen fájl, de 4G blokkot tartalmazhat
  - kis fájlokból álló táblatér (small file tablespace): több kisebb fájlból áll

Szegmensek, területek (extents), és blokkok:



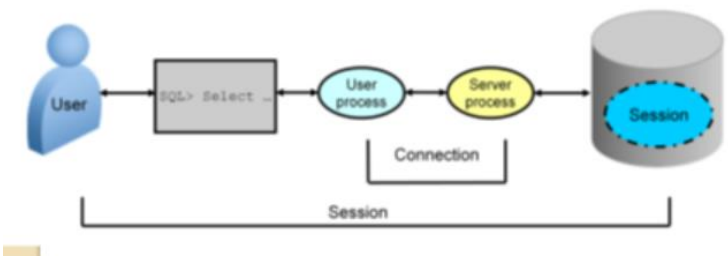
- Egy táblatér több szegmensből is állhat.
- Az adatbázis objektumokat, táblákat, indexeket a szegmensekben tároljuk.
- A szegmensek területekből (extents) állnak.
- A területek folytonos adatblokkok halmazai.
- Az adatblokkok az adatbázis legkisebb írható/olvasható egységei.
- Az adatblokkok operációsrendszerbeli blokkokra képezhetők le.
- Az adatblokk mérete alapértelmezésben 8K.
- Statikus adatbázis (adattárház) esetén nagyobb méretet érdemes használni, dinamikus adatbázis (tranzakciós adatbázis) esetén kisebbet.

13. SQL utasítás végrehajtásának folyamata: példány, felhasználói folyamat, szerver folyamat

- Egy példányhoz kapcsolódáshoz (Connect) szükséges:
  - Felhasználói folyamat
  - Szerverfolyamat
- Az SQL utasítás típusától függ, hogy az Oracle szerver milyen komponenseire lesz szükség:
  - A lekérdezések olyan folyamatokat indítanak, amelyek ahhoz kellenek, hogy megkapjuk a kívánt sorokat
  - Az adatmódosító (DML) utasítások naplózó folyamatokat is indítanak, hogy elmentsék a változásokat
  - A véglegesítés (Commit) biztosítja a tranzakciók helyreállíthatóságát
- Az Oracle szerver nem minden komponense vesz részt egy SQL utasítás végrehajtásában.

- Háromféleképp lehet egy Oracle szerverhez kapcsolódni:
  - ***OPERÁCIÓS RENDSZEREN KERESZTÜL:***
    A felhasználó belép abba az operációs rendszerbe, ahol az Oracle példány fut és elindít egy alkalmazást, amely eléri az adatbázist ezen a rendszeren. Ekkor a kommunikáció útvonalat az operációs rendszer belső kommunikációs folyamatai hozzák létre.
  - ***KLIENS-SZERVER KAPCSOLATON KERESZTÜL:***
    A felhasználó a helyi gépén elindít egy alkalmazást, amely a hálózaton keresztül kapcsolódik ahhoz a géphez, amelyen az Oracle példány fut. Ekkor a hálózati szoftver kommunikál a felhasználó és az Oracle szerver között.
  - ***HÁROMRÉTEGŰ (three-tiered) KAPCSOLATON KERESZTÜL:***
    A felhasználó gépe a hálózaton keresztül kommunikál egy alkalmazással vagy egy hálózati szerverrel, amely szintén a hálózaton keresztül össze van kötve egy olyan géppel, amelyen az Oracle példány fut. Például a felhasználó egy böngésző segítségével elér egy szerveren futó alkalmazást, amely egy távoli UNIX rendszeren futó Oracle adatbázisból gyűjti ki az adatokat.

- Mielőtt egy felhasználó küld egy SQL utasítást az ORACLE szervernek, előtte kapcsolódnia kell egy példányhoz.
- A felhasználói alkalmazások, vagy például az *i*SQL*Plus, felhasználó folyamatokat (*user process*) indítanak el.
- Amikor a felhasználó az Oracle szerverre kapcsolódik, akkor készül egy folyamat, amit szerverfolyamatnak hívunk. Ez a folyamat kommunikál az Oracle példánnyal a kliensen futó felhasználó folyamat nevében. A szerver folyamat hajtja végre a felhasználó SQL utasításait.
- Kapcsolódáskor egy munkaszakasz (session) kezdődik. A munkaszakasz a felhasználó érvényesítése (validálása) esetén kezdődik és a kilépéséig vagy egy abnormális megszakításig tart.
- Egy felhasználó több munkaszakaszt is nyithat. Ehhez szükséges, hogy az Oracle szerver elérhető, használható legyen. (Néhány adminisztrációs eszközhöz még ez sem szükséges).
- Megjegyzés: Ha fentieknek megfelelően egyértelmű a megfeleltetés a felhasználó és a szerverfolyamat között, akkor dedikált szerverkapcsolatról beszélünk.
- A kapcsolat egy kommunikációs útvonal a felhasználó folyamat és az Oracle szerver között.



Naplózás

14. Relációs ABKR-ek esetén mi a tranzakció fogalma

Egy tranzakció mindig egy konzisztens adatbázis állapotból indul ki és a módosítások egy olyan sorozatát tartalmazza, melyek végén ismét egy konzisztens adatbázis tartalom áll elő.

15. ACID tulajdonságok

- Atomikusság (**Atomicity**): A tranzakcióba bevont DML utasításokat egy egységként kell kezelnie az adatbáziskezelőnek
- Konzisztencia (**Consistency**): A tranzakció befejezése után az adatbázisnak konzisztens állapotba kell kerülnie.
- Elszigeteltség (**Isolation**): A párhuzamosan futó tranzakcióknak egymástól függetlenül kell működniük
- Állandóság (**Durability**): A lezárt tranzakciók eredménye nem veszhet el

16. Rendszerhiba, helyreállítás, naplózás, archiválás

- A tranzakció közben végrehajtott változásokra vonatkozó adatokat a tranzakció log(ok)ba írják ki. A log fájlt célszerű egy másik háttértárolóra tenni, mint az adatbázis fájlokat.
- Ezután módosítja a tranzakció a memóriába betöltött adatbázis rekordokat.
- Az adatbázis adminisztrátor rendszeres időnként mentéseket (backup) készít az adatbázisfájlok tartalmáról.

MPP (massively parallel processing) database

17. Classic Database Architecture are based on SMP (Symmetric Multi Processor) Systems. Vázolja fel ezen rendszerek 3 fő jellemzőjét!

- többprocesszoros szimmetrikus számítógép architektúra
- minden processzor saját cache-el rendelkezik
- memória, bus és a I/O rendszer megosztott

18. Massive Parallel Databases are based on a „shared-nothing“ Architecture. Sorolja fel ezen rendszerek 3 fő jellemzőjét!

- több processzor párhuzamosan futtat egy folyamatot, Mindegyik processzor az operációs rendszer egy saját példányát futtatja
- egy „Fő” node, a metaadatok és az adaszótár számára,
  N data Node tárolja a felhasználó adatokat,
  nagy teljesítményű hálózat a node-ok összekötésére
- minden node-nak saját processzor + cache, saját memória, saját háttértár

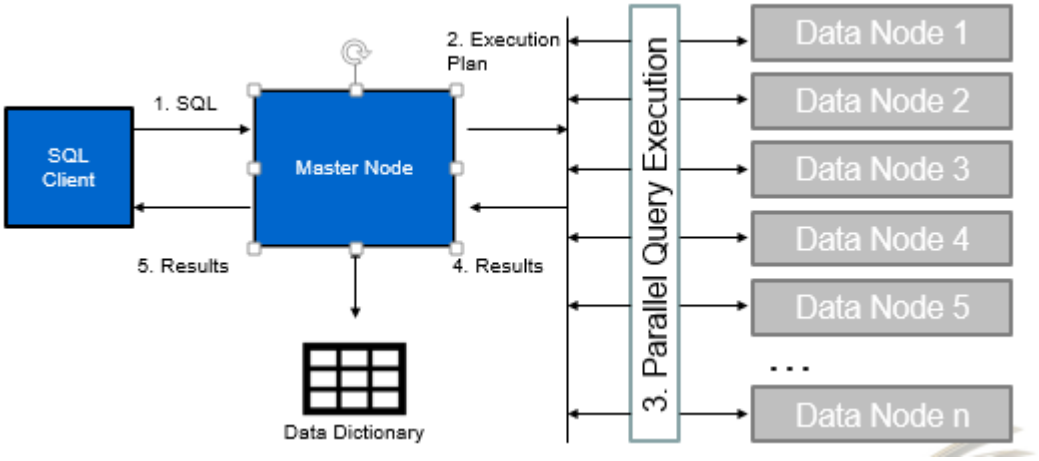
19. Milyen felhasználási területe van az MPP adatbázis rendszereknek?

appliance-ekként, azaz olyan komplett rendszerekként értékesítik, melyekben adatbázis-kezelésre optimalizált hardver és szoftver (operációs rendszer, adatbázis szerver) elemek egyaránt megtalálhatók.

20. Milyen data distribution stratégiákat ismer MPP rendszerek esetén? Ismertesse ezek jellemzőit!

- **Hash algoritmus:** A rekordokat hash eloszlás függvény alapján osztja el a lemezekre. Egy megadott oszlop alapján áll elő a függvény, minden sort hozzárendel egy és csak egy elosztáshoz. Azonos adattípusú, azonos oszlop értékek ugyanazok lesznek és ugyanarra az elosztásra kerülnek.
- **Round Robin:** i-edik rekordot az i mod n lemezre írja, ezzel biztosítja a rekordok egyenletes eloszlását a rendelkezésre álló lemezekre
- **Range partition:** tartomány partícionálás, a rekordokat bizonyos tartományok alapján osztja szét a lemezekre.
  Pl. nevek, A-H-ig 1.lemezre, I-Z-ig 2. lemezre stb.
- **Split on all:** Minden node-ra rákerül minden rekord, így minden node tárhelyén rendelkezik a teljes táblával.

21. Milyen rendszert ábrázol az alábbi ábra! Ismertesse az ábrán látható adat elérési folyamatot!



- MPP RDBMS adat elérési folyamata
- SQL kliensen keresztül kiadott parancsból a Master Node végrehajtási tervet készít, és Data node-okból lekéri az adatokat, vissza kapja az eredményt és az adat szótár használatával az SQL kliens számára feldolgozható formában adja vissza

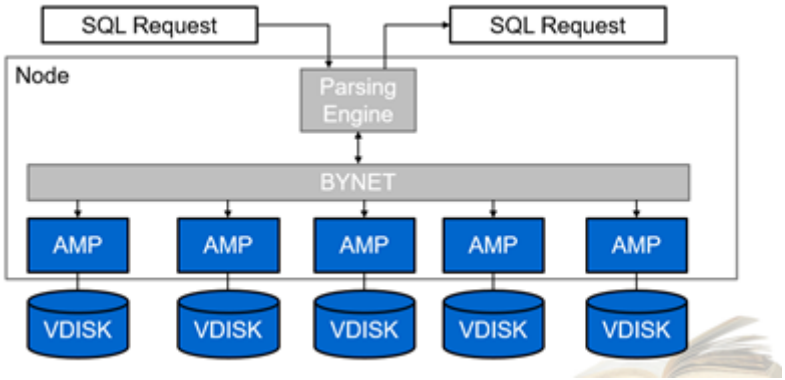
22. Soroljon fel 3 relációs MPP adatbázist!

- Teradata
- Vertica
- Netezza
- Greenplum

23. Írjon 8 kiemelten jellemző tényt a Teradata DBMS-ről!

- MPP architektúrán alapszik
- Shared nothing
- lineárisan skálázható, akár 2048 node-ig
- magas fokú összekapcsolhatóság
- érett optimalizáló, kezdetektől paralel végrehajtásra tervezték
- támogatja az SQL-t
- robosztus segédprogramok az adatok importálásához/exportálásához (FastLoad,MultiLoad,FastExport,TPT)
- automatikusan osztja el az adatokat a tárhelyeken

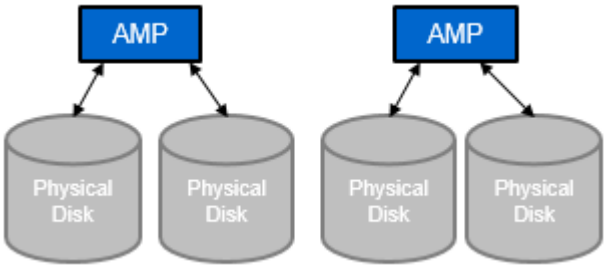
24. Ismerje az alábbi ábrán felvázolt Teradata rendszer-komponenseket, ismertesse ezek szerepét és jellemzőit.



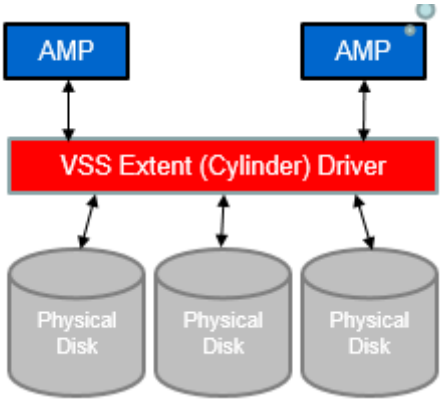
- **Node:** alapegysége a TeraData-nak, saját Oprendszer, Cpu, memória, és saját példányú TeraData RBDMS szoftver és lemezterület
- **Parsing Engine (PE):** virtuális processzor,
  elemei: Session Controller, Parser, Optimalizáló, Diszpécser.
  Kliens és az RDBMS közti kommunikációt kezeli.
- **BYNET:** Nagy sebességű hálózat, összekapcsolja és kommunikál az összes AMP-vel és PE-vel a rendszerben. Nagy teljesítményű, Hibatűrő, kiegyensúlyozott terhelésű, skálázható
- **AMP (hozzáférési modul processzor):** Virtuális processzor, minden AMP-n Adatbáziskezelő alrendszer található, saját virtuális lemeze van.
- **Lemezek:** adatokat tárolja, saját AMP-hez kapcsolódnak

25. Miben különbözik az alábbi 2 architektúrájú Teradata DBMS? Írja le a fő jellemzőiket és a különbséget!

1. rendszer:



2. rendszer:



Az 1. rendszer: pre-TVS: AMP-knek saját fizikai disk-jük van

A 2. rendszer: TVS-sel, VSS-nek van saját lemeze, az AMP-k tudnak a fizikai helyéről a lemezeknek

26. TVS típusú Teradata esetén mit jellemez a „Data Temperature” fogalma? Milyen 3 kategóriát ismer, és mit jelentenek ezek a fogalmak? Miért különböztetik meg ezt a 3 kategóriát az adatok tekintetében?

- Data Acces Frequency-t jellemzi
- Kategóriái:
  - HOT: nagyon sűrűn használt Data
  - WARM: gyakran használt Data
  - COLD: ritkán használt Data
- Megkülönböztetik őket, hogy a frequency-hez megfelelő teljesítményű tároló eszközt tegyenek alá:
  - HOT -> gyors, drága
  - WARM -> középgyors, olcsóbb
  - COLD -> lassú, olcsó

27. Teradata data distribution folyamata.

- A Terradata Hash Algoritmust használ hogy véletlenszerűen elossza a Table-Rowkat az AMP-k között
- A Primary Index kiválasztása meghatározza hogy egy Tábla sora egyenlően vagy egyenlőtlenül lesz elosztva az AMP-k között
- Egyenlően elosztott Table Row-k, egyenlően elosztott Workload-okat eredményeznek
- Terradata automatizálja a fizikai Data Location-ök elhelyezését és karbantartását

28. Teradata esetén mit jelentenek a következő fogalmak: primary key, primary index, partitioned primary index, secondary index.

- Primary key: a relációs adatmodellezésben az a kifejezés, amely meghatározza az entitás egyedi azonosító attribútumait.

- Primary index: az elsődleges indexoszlop értékét használják a Teradatában az AMP-eken való tárolás meghatározásához hash algoritmus segítségével.

- Partitioned primary index: a sorrend, amelyben a sorok tárolva vannak: PPI használatával a sorokat először a partíció majd a row-hash tárolja. Javíthatja a nagytáblák teljesítményét a range-lekérdezésekben ún. partition elimination technikával. Ez azt jelenti, hogy az optimalizáló felismeri, hogy a partíciók, amelyek nem tartalmazhatják a kért adatokat, ki lesznek zárva az adat visszakeresés során.

- Secondary index: egy alternatív út az adatokhoz. A teljesítmény javítására használják, elkerülve ezzel a full table scan-t. Olyan, mint egy primary index, mivel lehetővé teszi a rendszer számára a táblázat egy sorának keresését. Nincs hatással a sorok elosztására az AMP-eken. A másodlagos indexkeresés általában olcsóbb, mint egy full table scan. Plusz költség a lemezterületen és a karbantartásban, de szükség esetén hozzáadhatók vagy eldobhatók.

Relational DBs

29. Emeljen ki 6 olyan tulajdonságot, ami az SQLite adatbázis-kezelőre jellemző! Adjon meg 3 nagyon gyakori alkalmazási területet SQLite választásához! Milyen hiányosságai vannak az SQLite-nak (3 db-ot írjon)!

- Kis erőforrás igényű rendszer
- adott adatbázist több szál is használhatja
- számos fejlesztőkörnyezetből használható
- platform független
- nyílt forráskódú
- Kis méretű
- jólbeágyazható
  - Gyakori alkalmazása:Beágyazott rendszerek, mobiltelefon, böngészők süti tárolására(Mozilla firefox)
  - Hiányosságai: nincs hozzáférés szablyozás, Nagy Dataset, magas konkurencia

30. Fogalmazza meg a HIVE adatbázis-kezelőről legfontosabb tanultakat 5 mondatban.

- Facebook által lett fejlesztve, top level Apache project.
- A data warehousing infrastruktúra, Hadoop alapú.
- Azonnal adatot készít egy klaszteren a nem Java programozók számára SQL lekérdezéseken keresztül. HiveQL-re épült, SQL szerű lekérdezési nyelv.
- Értelmezi a HQL-t és létrehozza a MapReduce munkákat, amelyek a klaszteren futnak.
- Lehetővé teszi az adatok egyszerű összegzését, ad-hoc jelentését és lekérdezését és a nagy adatmennyiségek elemzését.

31. Hasonlítsa össze a MySQL és PostgreSQL adatbázis-kezelőket 12 db (előadáson vett) szempont szerint

1. Open source:
- ***MySQL***: a fejlesztési project megcsinálta a GNU General Public License általános szabályai szerint elérhető forráskódot, valamint számos saját tulajdonú licenc megállapodásokat. Most Oracle Corporation tulajdonában van, és számos fizetett kiadást kínál saját tulajdonú felhasználásra.
- ***PostgreSQL***: a PostreSQL Globan Development Group fejlesztette, amely egy sokszínű csoportja vállalatoknak és egyéni hozzájárulóknak. Igyenes és opensource szoftver. A PostgreSQL licenc egy liberális nyílt forráskodú licenc, hasonló a BSD vagy az MIT licencekhez.

2. ACID-kompatibilitás:
- ***MySQL***: csak az InnoDB és az NDB Cluster Storage motorok használatával ACID-kompatibilis
- ***PostgreSQL***: ACID-kompatibilis, biztosítja, hogy mindenki követelménye teljesüljön

3. SQL-kompatibilitás:
- ***MySQL***: részben megfelel néhány verzió (SQL: 2003)
- ***PostgreSQL***: nagymértékben SQL-kompatibilis (SQL: 2008)

4. Párhuzamos kezelés:
- ***MySQL***: MVCC támogatás csak az InnoDB-ben
- ***PostgreSQL***: hatékonyan összeegyezhethető az MVCC-vel a végrehajtás

5. Replikáció:
- ***MySQL***: master-master, master-slave
- ***PostgreSQL***: master-slave

6. Teljesítmény:
- ***MySQL***: széles körben használt webalapú, BI alapú projekteken, ahol a gyors olvasási sebesség nagyon praktikus. Kevesebb teljesítmény: nehéz terhelés, komplex lekérdezés.
- ***PostgreSQL***: olyan nagy rendszereknél használják, ahol az olvasási- és írási sebesség fontos. Kevesebb teljesítmény: nehéz frissítés, olvasási sebesség.

7. Biztonság:
- ***MySQL***: ACL használata a kapcsolatokhoz, lekérdezésekhez, valamint a felhasználói és egyéb műveletekhez. SSL támogatás a kliens és a szerver között.
- ***PostgreSQL***: SSL támogatás a kliens és a szerver között. Támogatja a sorszintű biztonságot, és további ellenőrzéséket a SE-PostgreSQL protokoll segítségével (SELinux alapán).

8. Felhőszolgáltatás:
- mindkettő technológia támogatja a jelentősebb felhőszolgáltatásokat (Amazon, Google, Microsoft, DO)

9. Materializált nézet:
- ***MySQL***: támogatja az ideiglenes táblákat, de nem támogatják a materializált nézeteket
- ***PostgreSQL***: támogatja a materializált nézeteket és az ideiglenes táblákat is

10. 1CTE / Analytical (window) funcitons:
- ***MySQL***: támogatott a CTE és a window funciton-ök a 8.0 verziótól
- ***PostgreSQL***: CTE és window functions támogatás a kezdetektől

11. Indexek:
- **MySQL**: a legtöbb index B-fa és hash alapú, és támogatja a clustered index technológiát
- ***PostgreSQL***: tartalmaz beépített támogatás a hagyományos B-fára és a hash indexekre, de elérhetőek expression indexek és partial indexek

12. JSON kiterjesztés:
- ***MySQL***: JSON adattípussal rendelkezik, de nincs más NoSQL funkció, nem támogatja a JSON indexelés
- ***PostgreSQL***: támogatja a JSON-t és más NoSQL funkciókat, mint a natív XML támogatást és a kulcs-érték párokat HSTORE-ral, támogatja a JSON indexelést

13. Slogans
- ***MySQL*:** the most popular relational database
- ***PostgreSQL*:** the most advanced(opensource) relational database

NoSQL

32. Jellemezze a 3Vs tulajdonságokat!

- **Volume:** A klasszikus relációs adatbázis-architektúrák csak a Scale-Up funkcióval rendelkezhetnek,drága hardver.
- **Variaty:** A tárolt adatokat előre meg kell határozni. A változtatásokhoz mindig drága és időigényes módosítások szükségesek a modellben.
- **Velocity:** a klasszikus RDBMS rendszerek a sok párhuzamos írás-olvasással nehezen boldogulnak

33. Milyen felhasználási területekről érkezett az első igény a nem tisztán relációs adatbázisok iránt?

elsősorban a közösségi alkalmazások/oldalak felől volt igény rá

34. Soroljon fel igényeket az új rendszerek felé?

- Nagy adatmenniység ellenére a nagy teljesítmény elérése,Trade-off NoSQL esetén a konzisztencia,2 szempontból:
- CAP tétel
- NoSQL adatbázisokban csak a Base-t kell végrehajtani

35. Mit tud a CAP tételről?

Az elosztott rendszerek mindig eleget tesz a következő követelmények közül kettőnek, de sose mindháromnak egyszerre

- Konzisztencia(C)
- Elérhetőség(A)
- Partíció tolerancia(P)

36. Mit jelent a BASE kifejezés? Jellemezze a 3 fogalmat 1-1 mondatban!

- optimista tárolási stratégiával írja le a property-ket
- Basic availability: a tároló az idő nagy részében működik
- Soft-state: a tárolók állandóan kölcsönösen konzisztensek
- Eventual consistency: a tárolók későbbi pontban mutatnak konzisztenciát

37. Milyen 2 fő replikáció típust ismer? Jellemezze őket 2-3 mondatban!

- Master-slave: egy node-ot alkot a hiteles másolatnak ami kezeli a slave szerverek szinkronizációját, és kezeli az olvashatóságot
- Peer-to-peer: a replikáció lehetővé teszi bármelyik node írását, a node kezeli a másolatainak a szinkrinozálását

38. 4 fő NoSQL rendszer kategória

- key-value store
- Columnar store
- Document store
- Graph store

39. Hasonlítsa össze a vertical scaling és a horizontal scaling fogalmakat!

- vertical scaling esetén több erőforrást(cpu,memoria) adunk a géphez,
- horizontal scaling esetén pedig több gépet rakunk a rendszerbe

Key Value store, Redis

40. Soroljon fel tipikus use case-eket Key Value store használatára!

- Caching
- Session Store
- Messaging Channels
- Creating secondary Indexes

41. Mondjon 3 Key vaue store ABKR-t!

- Redis
- Amazon DynamoDB
- Microsoft Azure CosmosDB
- Hazelcast

42. Milyen programozási nyelvekhez van API a Redis esetében (min. 5-öt!)

- java/javascript
- scala
- python
- ruby
- c/c++c#
- R

43. Miért in-memory típusú rendszer a Redis?

Mert az összes adatot, amit a Redis-ben használunk, a mi DB-server-ünk memóriájába tárolja.

44. Soroljon fel 5 parancsot a Redis esetében! Magyarázza el a jelentésüket.

- Set key value : kulcs érték beállítása
- Keys * : kulcs keresés
- del key : kulcs törlés
- get key : a kulcshoz tartozó érték lekérése
- exsist key : kulcs létezés vizsgálata

45. Milyen adattípusokat ismer Redis esetén?

string, lists,sets, hashes, sorted sets, bitmaps and hyperlogs

HBase

46. HDFS és HBase rendszerek kapcsolata és különbsége

A HDFS nagy fájlok tárolására alkalmas megosztott fájl rendszer, az HBase a HDFS-re épített adatbázis.
Míg a HDFS nem támogat gyors egyéni rekord lekérdezést, az HBase nagyobb táblákra is igen.
A HDFS nagy késleltetésű kötegelt feldolgozást (=nincs kötegelt feldolgozás), míg az HBase alacsony késleltetéssel biztosít egyedülálló sorokat több milliárdnyi rekord közül (véletlenszerű elérés).
A HDFS csak szekvenciális adat elérést tud, míg az HBase belső hash táblákat használ, amiben az adatokat indexelt HDFS fájlokban tárolja, a gyorsabb lekérdezés érdekében.
Véletlen eléréssel dolgozik.

47. HBase is more a "Data Store" than "Data Base". Miért?

Azért, mert hiányoznak belőle a relációnális adatbázisokra (RDBMS) jellemző tulajdonságok, mint típusos oszlopok, másodlagos indexek, triggerek, fejlett lekérdező nyelvek, stb.

48. HBase logikai adatmodell (row key, column family, column, column qualifier, cell value, version/timestamp). Sparse table, dynamic columns.

Row Key: Minden sornak van egy egyedi kulcsa, ami azonosítja a sort.

Column family: A Keyspace-ben vannak, (olyanok, mint a táblák relációnális modellben) sorokat tartalmaznak, amin belül oszlopok találhatóak.

Column: Minden oszlop tartalmazza az oszlop nevét, értékét, és időbélyegét.

Column qualifier*:* Oszlop nevek, azaz, oszlop kulcsok.

Cell value: Adott sorban, azon belül, adott oszlop érték mezője.

Version / Timestamp: Az adatbeszúrás dátumát és idejét tartalmazza. Ezzel határozható meg a legfrissebb adat verzió.

Sparse table: Arra utal, hogy nagyon sok az üres cella.

Dynamic columns*:* HBase tulajdonság, melyre azért van szükség, mert az oszlopnevek a cellákon belül vannak kódolva, különböző celláknak pedig különböző oszlopjuk van.

49. HBase fizikai adatmodell(HFile tartalma, regions, table cell tartalma), region server.

HFile tartalma:Minden column family külön fájlokban vannak letárolva, ezeket hívjuk HFiles-nak. Ezekben a kulcs és verzió számok mindegyik column family-be többszörösen bekerülnek, az üres cellákat pedig nem tárolják.

Regions: Minden column family, minden táblában horizontálisan particionálásra kerülnek, region-ökbe, key range alapján. Ezek kerülnek aztán a data nodes-okba, "RegionServers"-ként. Ezek alapján képes az HBase horizontális sharding-ra.

Table cell: Az adatok Hbase tábla cellákban tárolódnak el. A teljes cella, a hozzáadott struktúrális információkkal, a Key Value. Azokra a cellákra, amelyekbe állítottunk értéket, teljes valójukban tárolásra kerülnek. (Row key, column family name, column name, timestamp, és az érték)

Region server: Region-ök csoportját a Region server szolgáltatja (körülbelül 1000 region-t) a kliensnek.

50. HBase architecture. Ismerni kell a következő komponensek/fogalmak jelentését, szerepét:NameNode, ZooKeeper, Hmaster, Region server, Meta table, WAL, BlockCache, Memstore, Hfile, Region Flush, Minor compaction, Major compaction, Data replication.

NameNode: Ez tartja fenn a meta adat információkat az összes olyan fizikai adatblokk számára, amelyek a fájlokat tartalmazzák.

ZooKeeper*:* elosztott koordinációs szolgáltatás a kiszolgáló szerverek állapotának fenntartásához a clusterben. A ZooKeeper tartja számon, hogy mely szerverek elérhetőek, és szerver hibát is jelez (heartbeats-en keresztül). Megegyezéses alapon biztosít közös, megosztott állapotot. (3, vagy 5 gép kell a megegyezéshez)

HMaster*:* A region-ök ddl végrehajtását (create, delete) az Hmaster végzi. Felelős a region szerverek koordinálásáért, és az admin funkciók ellátásáért.

Meta table: HBase tábla, ami az összes region-t számon tartja a rendszerben. (B fára hasonlít.)

WAL*: (WriteAheadLog)* Egy fájl a megosztott fájl rendszeren. Olyan új adatok tárolására szolgál, amiket még nem tároltunk véglegesen. Visszaállításra szolgál, hiba esetén.

BlockCache*:* Olvasó cache, gyakran olvasott adatokat tárol a memóriában. A legrégebben használt adat akkor kerül ki, amikor a cache megtelik.

MemStore*:* Író cache, ami a disk-re írás előtt rendezi, és tárolja az adatot. Minden column familyre region-önként egy MemStore jut.

HFile*:* Key Values által rendezett sorokat tárol a diszken.

Region Flush: Amikor kellően megtelik a MemStore, az adatok egy új HFile-ba kerülnek, amelyek azidő alatt jönnek létre, míg a KeyValue alapján a MemStore szortírozza az adatokat. Maga a Flush tehát az a folyamat, amely a MemStore-okkal a region-ben létrehozza a diszkre az új HFile-okat.

Minor compaction: Az a folyamat, amely során az HBase több kisméretű HFile-okat kevesebb számú, de nagyobb méretű HFile-okká írja át. (Merge sort segítségével hajtja végre)

Major compaction: Egy region-ben az összes HFile-t egy HFile-ba mergeli, column family-nként, és a folyamat közben törli a törölt, vagy lejárt cellákat. Ezzel, növeli az olvasási teljesítményt, viszont amíg a merge tart, megterheli a diszkeket, és a hálózat forgalmát is, ezért ezt hétvégékre, vagy estékre szokták időzíteni.

Data replication*:* Az összes kiolvasás, és írás a primary node-ról származik. A HDFS a WAL-t, és a HFile-okat replikálja, amely automatikusan történik. Amikor írás történik a HDFS-en, egy helyi másolat készül, aztán egy replikált a másodlaogs node-ra, és egy harmadik másolat a harmadlagos node-ra.

MongoDB

51. Milyen típusú adatokat tárol? (BSON)

Majdnem bármilyen típusú adatot képes tárolni (stringek, számok, dátumok stb.)

52. Mit jelent a BSON? Hogyan kapcsolódik ez a fogalom a Mongodb-hez?

- BSON egy számítógépes adatcsere-formátum.
- A MongoDB a JSON dokumentumokat bináris kódolású formátumban, úgynevezett BSON formátumban kezeli.

53. Soroljon fel 5 különböző adattípust Mongodb-ben!

- Sok adattípust kezel: string, integer, boolean, double, arrays, data, array, symbol, code

54. Soroljon fel 5 operátort!

- Update operators: $set, $unset, $max, $min, $rename, $inc
- Logical operators: $and, $or, $not, $nor
- other: $eq, $gt, $gte, $lt, $lte, $in, $ne, $nin

55. Sorolja fel az RDBMS szerinti database, table, row, column, index, join fogalmak megfelelőit MongoDB-ben!

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field

index -> index

join -> there is no support for complex joins in MongoDB

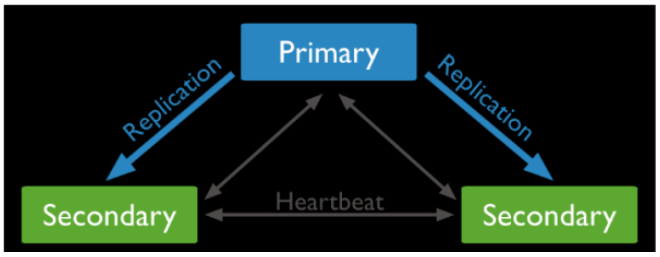
56. Jellemezze a MongoDB architektúra komponenseit!

Primary, Secondary, Arbiter, Hidden node, Router, Config server

Replica set, Shard

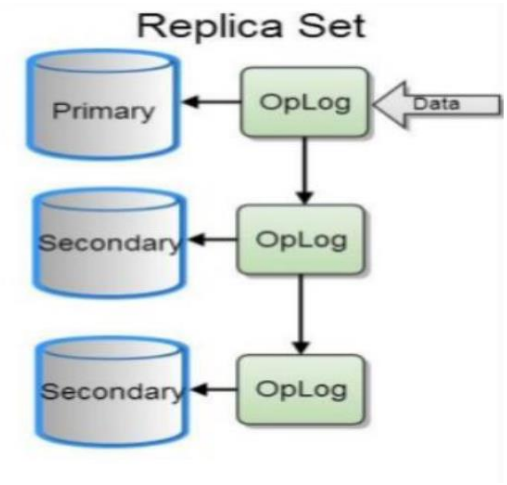
melyik node-ra írunk, melyik node-ról olvasunk

mikor nem elérhető a rendszer?



57. Legyen tisztában a következő fogalmakkal, tudja őket 1-2 mondatban jellemezni: oplog, szavazás, shard key, write concern, read concern

Oplog



- ***szavazás:*** Ha a primary node leáll, a secondary nodeok közül kell újat választani. A szavazatok többségét kapó lesz az új primary node.
- ***shard key:*** Mindegyik shard cluster tud másolni is az adatok tartóssága és rendelkezésre állása miatt. Az adatállományt a shard keyek alapján osztják meg.
- ***write concern:*** Többszinten garantálják az adatok elterjedését a másolatokon.
- ***read concern:*** Többszinten állítják be a biztonságos adatok olvasását a clusterből.

Graph DB, Neo4j

58. Soroljon fel 3 példát olyan alkalmazásra, ahol érdemes lehet gráf adatbázist alkalmazni. Milyen típusú adatok esetén célszerű alkalmazni?

- Útvonal tervezők
- Közösségi oldalak
- Hálózati topológia tervező
- komplex, hierarchikus szerkezetű adatok tárolására ,egyedek, kapcsolatok, tulajdonságok, címkék

59. Mondjon egy olyan példát, amit hatékonyabb gráf adatbázisban modellezni, mint relációsban.

Útvolnal tervezés

60. Native graph processing jelentése

Az adatbázis tulajdonságára utal. Egy adatbázis motor, ami az indexmentes szomszédságot használja, minden egyes csomópont közvetlen hivatkozásokat tart fenn a szomszédos csomópontokra. Minden csomópont mikroindexként működik a közeli csomópontok számára.

61. Index free adjacency jelentése

Az indexmentes szomszédság sokkal hatékonyabbá teszi a gráf bejárást, mint a globális indexstruktúra létrehozása nem natív gráf adatbázisban (gráf implementálása relációs adatbázisban). Különösen akkor, ha a kétirányú keresések fizikailag kapcsolódnak, a csomópontoknak meg vannak az előnyei a globális indexstruktúrával szemben.

62. Labeled property graph adatmodell jellemzése

Csomópontokat és kapcsolatokat tartalmaz. A csomópontok tartalmazzák a kulcs-érték párok tulajdonságait. A csomópontokat egy vagy több címkével lehet címezni. A kapcsolatokat elnevezzük és irányítjuk, és mindig van egy kezdő- és egy végcsomópont. A kapcsolatok is tartalmazhatnak tulajdonságokat.

63. RDF adatmodell jellemzése. RDF jelentése, RDF adatmodell, mint gráf, RDF lekérdező nyelve

Resource Description Framework

alany -> állítmány -> tárgy

Az alany mindig egy erőforrás, a tárgy is lehet egy erőforrás, vagy csak irodalmi.

Minden állítás két csompópontot hoz létre a gráfban. Néhány egyedileg azonosított egy URI-val (erőforrás), néhány csak egyszerű tulajdonság (irodalmi). Minden állítás élt hoz létre. A csomópontoknak és az éleknek nincs belső struktúrája. Lekérdező nyelve a SPARQL.

64. Soroljon fel gráf típusokat

- Irányított/Irányítatlan
- Ciklukus/Aciklukus
- Súlyozott/Súlyozatlan
- Ritka/Sűrű

65. Gráf traversing/ gráf bejárás jelentése. 2 alapvető gráf bejáró algoritmus

egy gráf egyik tulajdonsága, hogy összefüggő-e, a gráfbejárással ezt a tulajdonságot lehet ellenőrizni. akkor bejárható egy gráf, bármelyik csúcsából indulva bejárhatjuk úgy, hogy minden csúcsba elérünk.(csomópontok és élek módszeres meglátogatása)

- Mélységi bejárás
- Szélességi bejárás
- legrövidebb út(Dijkstra)

66. Gráf adatmodell tervezésének szempontjai

- csomópontok modellezése
- kapcsolatok modellezése
- modellezési idő

67. Relációs modell konvertálása labeled property graph modellbe

- entiti típus(tábla) -> csúcspont azonos cimkével
- tábla rekord-> 1 csúcspont entitás cimkéval
- Attribútumok (tábla oszlopok)-> csúcspont tulajdonságok
- PK-FK kapcsolat -> kapcsolatok
- M:N tábla attribútumok -> kapcsolat tulajdonságok

68. Mit tud a Neo4j native graph storage-ról?

- egyik legkedveltebb gráf adatbázis kezelő
- Cypher leíró nyelvet használ
- nincs benne sharding

69. Van-e Neo4j-ben sharding?

Nincs

70. Mit jelent a replication Neo4j-ben?

Replikációt használ, hogy az összes claster számára elérhető legyen a adott gráf

71. Cypher nyelv jellemzése (diák alapján): főbb parancsok (ismerjen néhányat fejből), megszorítások, indexek

- Neo4j leíró nyelve
- SQL nyelvhez hasonló
- case sensitive
- Főbb parancsok: MATCH RETURN (MATCH pattern RETURN result) ,CREATE RETURN,
- megszorítás példa: MATCH(x:Person)-[:knows]->(y.Person) RETURN x.Name, collect(y.Name)
- Index példa: Create index on :Person(Name)

72. Mit tud a Clustering architecture-ről Neo4j esetében?

Since Neo4J is dealing with highly connected Data, it offers no solution for Sharding like the Aggregate Stores we met before. Neo4J Clusters are using Replication to ensure, that a Graph is availible on every Machine of the Cluster.

The Roles of the Machines are definied as this in an Neo4J Cluster:
The Core Servers
Their main responsibility is to Safeguard the Data. The Core Servers replicate all Transactions between each other. The Raft Protocol ensures, that the Data is safly durable before confirming transaction commit to the end User Application
The Read Replicas
The irresponsibility is to Sacle out the Workload. They are Neo4J Databases capable off ullfilling read-only queries and procedures.

Synchronizing Read Replicas:
Read Replicas are asynchronously replicated from Core Servers. A Read Replica periodically (msRange) polls a Core Server for any new Transaction that is processed since the last poll. The Core Server then ships the Transactions to the RR.
Outage of Servers:
If the Core Server Cluster suffers enough failiures that it can no longer process writes, it will be comeread-only to preserve Safty.
Losing a RR dows not impact the Clusters availibility. It only means a loss in Query throughput performance.

Reasons for Clustering Neo4J:
1)High Availibility
Makes the Graph Application resilient, when it comes to Hardware or even Data Center Outages.
2)Scaleability
React on increasing Read-Requests from Applications/Users with increasing the numbe rof Read-Replicas. Especially analytic Application on the Graph (with complex, longrunning Queries) can be directed to dedicated Read Replicas to be divided from the OLTP RRs.