Előadás #01 09-09-08

2009. szeptember 8.
8:00

Tárgy honlapja: https://it.inf.unideb.hu/honlap/dba

Database Administrator (DBA)
Feladatai:
- Cég adatbázisaiért felelős
- Működés, hatékonyság, ill. az adatbázisokat használó alkalmazások működése
- Az adatbázis tervezése és karbantartása

Adatbázis	<------->	adatbáziskezelő-rendszer (DBMS)

A DBA minden feladatát meg kell tervezni, és ezt kell végrehajtani.
Reaktív megközelítés: Ha megtörtént valamilyen hiba, akkor azt meg kell csinálni.
Proaktív megközelítés: Gondolkozzak előre. Meg lehet előzni a problémákat.

DBA nem maradhat ki az alkalmazásfejlesztés életciklusának egyetlen részéből se.

1. Követelmények feltárása: Vannak-e a cégnél újrafelhasználható adatok?
2. Elemzés/Tervezés: az adatmodelleket koncepcionális és logikai adatmodellt kell készíteni.
3. Fejlesztés: Előtte fizikaiba konvertálja a modellt.
4. Tesztelés: DBA biztosít tesztadatokat.
5. Működés: biztosítja az elérhetőséget, teljesítményt figyel, biztonsági mentést készít, jogosultságokat ad.
6. Karbantartás: Ha az üzlet változik, akkor lehet, hogy az adatbázisnak is változni kell.

A DBA felelős a teljes adatbázis-környezetért, az adatbázis-kezelő telepítése, IT infrastruktúra meghatározása, az alkalmazások elérhessék az adatbázist, Ad-hoc lekérdezések támogatása.

Adatadminisztrátor:
- Az üzleti felhasználóknak szükséges adatok azonosítása és katalogizálása.
- Koncepcionális és logikai adatmodell létrehozása, az üzleti folyamatokhoz tartozó adatelemek közötti kapcsolatok pontos leírá sa.
- Egy vállalati adatmodell létrehozása, amely magában foglalja a cég minden üzleti folyamatának minden adatát
- A cég adatpolitikájának meghatározása.
- Az adattulajdonosok és gondnokok azonosítása.
- Az adatok kontrollálásához és használatához a szabványok meghatározása.

Adatbázis adminisztrátor:
- Létrehozza a fizikai adatmodellt.

Rendszeradminisztrátor:
- Az adatbázis-kezelő rendszer megvalósítása.
- IT infrastruktúra megvalósítása.
- A DBMS együtt tudjon dolgozni más eszközökkel.

DBA feladatok:
- Biztosítsa, hogy a cég adatai hasznosak, használhatóak, elérhetőek, korrektek legyenek.
  - Adatbázis terv (relációs) létrehozása, megvalósítása.
  - Koncepcionális ill. logikai adatmodell létrehozása.
  - Logikai adatmodell megvalósítása fizikai adatbázisból.
- Teljesítmény monitorozás és hangolás:
  - Terhelés: Igények (online tranzakciók, batch feladatok, ad-hoc lekérdezések, adattárház kérdések, analitikai kérdések) definiálják.
  - Áteresztőképesség: A teljes HW és SW kapacitását definiálja. I/O sebesség, processzor sebesség, a gép párhuzamos kapacitása, OS és rendszerszoftver hatékonyságát, stb...
  - Erőforrás: SW és HW eszközök a rendszerben.
  - Optimalizáció: Adatbázis kérésekhez lehet rendelni hatékony elérési útvonalat. Ezt általában a DBMS teszi meg relációs lekérdezéseknél. Adatbázis elemeket, SQL formulákat is lehet optimalizálni.
  - Versengés: Ha egy erőforrásért magas az igény, akkor versengés következik be. Ahogy a versengés nő, úgy csökken az áteresztőképesség.
- Az adatbázis teljesítmény úgy definiálható, mint az erőforrás használat optimalizálása, az áteresztőképesség növelése, a versengés csökkentése, a lehetséges legnagyobb munkaterhelés mellett.
- Elérhetőség: Gyakran összefügg a teljesítménnyel. Leállási idők minimalizálása, ehhez kell a proaktív szemlélet.
- Adatbázis biztonság és feljogosítás: Jogosultságkezelés. Az is feladat, hogy ne törjék fel az adatbázist. Nem csak grant és r evoke van, lehet nézeteket is használni!
- Mentés és helyreállítás: Fel kell készülni a problémákra.
  - A legutolsó konzisztens állapotra állítsuk vissza az adatbázist.
  - Alkalmazói hiba esetén egy időpillanatra kell helyreállítani.
- Adatintegritás: Nem szabad hagyni, hogy az adatok megsérüljenek.

    - Fizikai
    - Szemantikai
    - Belső: index helyreállítás, mutató konzisztencia, mentési konzisztencia (ne készüljenek helytelen mentési másolatok.
- DBMS verzió migráció:
- Ezermester

Előadás #02 09-09-15

2009. szeptember 15.
8:00

Hány DBA szükséges?

Sok tényezőtől függ:

- Adatbázisok száma
- Adatbázisok mérete
- Felhasználók száma (felhasználók minősége ;])
- Alkalmazások száma
- Szolgáltatás szintű megállapodás (Service-level agreement (SLA)): végfelhasználó és DBA+ fejlesztők közötti megállapodás: követelmények, leállási idők meghatározása.
- Elérhetőségi követelmények: mennyi leállási idő engedélyezett.
- Leállás hatása: pénzügyi értelemben kell érteni.
- Teljesítmény követelmények
- Alkalmazások típusai
  - Küldetéskritikus alkalmazások: nem állhat le.
  - OLTP: Online Transaction Processing: Klasszikus banki rendszer. (Kell a 100%-os konzisztencia)
  - OLAP:Online Analitical Processing: Adattárházakba használják: a cégnél lévő OLTP rendszerek adatait gyűjtik össze és elemzésre használják.
- Illékonyság
- DBA csapat tapasztalata
- Programozói személyzet tapasztalata: a programozót tudnak-e SQL-ül, vagy PL/SQL-ül
- Végfelhasználók tapasztalata: Ad-hoc SQL lekérdezések.
- DBMS rendszerek változatossága: Pl. van egy Ora meg egy MSSQL.
- Adatbázis adminisztrációs eszközök: Minél több olyan eszköz van, ami automatizál, annál kevesebb a DBA feladata
- Teszt- és termelési rendszer: A két rendszernek nem kell ugyanolyannak lennie. DBA feladata a tesztadatok megadása.
  - Tesztrendszer: Alkalmazásfejlesztőknek sandbox. Nem biztos hogy csak 1 van, lehet több is, a párhuzamos alkalmazásfejlesztés miatt. DBA feladata az alkalmazások termelési környezetbeli viselkedésének, teljesítményének megbecslése.
  - Termelési rendszerben már nincs fejlesztés, 100%-os konzisztencia kell.
- Egyéb rendszerek:
  - Minőségbiztosítás

Adatbázis kapcsolódások esetén: (MVC)

- Megjelenítési logika: Ügyfélszolg-os kislány kattintgat rajta
- Üzleti logika: A szoftver valódi működése
- Adatbázis réteg

Adatbázis környezet létrehozása:

1. DBMS kiválasztása, telepítése: Ehhez valamilyen stratégiát szoktak használni
   Milyen DBMS-t válasszunk?
   Lehetőleg egyfélét DBMS-t használjunk.
   1) Oracle
   2) MSSQL
   3) DB2
   4) Postgres
   5) Mysql

   i. Operációsrendszer támogatás
   ii. Cég típusa
   iii. Benchmark-ok: Teljesítményre vonatkoznak leginkább.
   iv. Skálázhatóság: Milyen felhasználó számot és milyen db méretet támogat a DBMS
   v. Támogató szoftvereszközök elérhetősége: Lekérdezés elemző, adattárház támogató, adatbázis adminisztrációs, mentés/helyreállítás, teljesítmény monitorozás, adatbázis segédprogramok
   vi. Szakemberek: van-e elegendő képzett szakemberem.
   vii. Tulajdonjog költsége: Licence, támogató szoftverek, szakemberek költsége, DBMS-hez szükséges erőforrások költsége.
   viii. Kiadások ütemezése: Lehet anyagi szempont, illetve ha jön egy új kiadás. a változások kezelése.
   ix. Ügyfél referencia: Megkérdezed a szomszédot. :)

Előadás #03 09-09-22

2009. szeptember 22.
8:00

Az USB-s nagyon jó! xD

DBMS architektúrák:

- Enterprise (vállalati): Nagy teljesítményhez, skálázhatósághoz van tervezve, sok felhasználó, többféle alkalmazás, általában nagy gépen fut, több processzort támogat, párhuzamos lekérdezéseket támogat.
- Departmental (ágazati): Kis vagy közepes munkacsoport, kevesebb a funkcionalitása.
- Personal (személyi): Egy felhasználót támogat, általában PC-n fut, sokkal olcsóbb, mint egy vállalati, nem is alkalmas több felhasználós alkalmazásokhoz.
- Mobil: Vállalati vagy ágazati különleges változata. Kézi eszközre telepítik, és egy helyi adatbázis elérést biztosít, és bizonyos időközönként szinkronizálni kell az adatokat egy központi szerverrel.

Klaszterezés: több független számítógép együtt végzi a munkáját, és ez kívülről úgy néz ki mintha 1 gép dolgozna, skálázhatóság és elérhetőség érdekébe teszik ezt. 99.99%-os elérhetőség.
Architektúrák:

- Shared-disk: Ugyanazon a lemezeszközön osztoznak, minden processzorhoz külön memória tartozik. Általában nagygépes környezetben van, 2-3 nagygépet kapcsolnak össze. Ha egy gép kiesik, akkor a többi átveszi a feladatát.
- Shared-nothing: Minden gépnek saját erőforrása van. Van egy hálózat, amin kommunikálnak egymással. Kliens kérés az erőforrásgazdához irányítódik. Ha egy gép kiesik, akkor a többi átveszi a feladatát. Sok kis PC-t kapcsolnak össze.

Hardver + OS nagy hatással van az adatbázis környezetre.

DBMS telepítése:

- Elolvassuk a telepítési útmutatót! :)
  - OS követelmények
  - Hardver követelmények
- Tárolási követelmények: az adatbázis hatékony működéséhez ezeket jól kell elhelyezni.
  - Rendszerkatalógus vagy adatszótár: A DBMS működéséhez szükséges dolgok vannak az adatszótárban.
  - Naplóállomány (aktív és archív naplók, rollback szegmensek, és más naplók): a módosítások kerülnek bele, elég nagy mennyiségű adat.
  - Indító és vezérlő állományok: nem nagy mennyiség.
  - Munkaállományok - ezek állományok
  - Ideiglenes adatbázis struktúrák - ezek meg DB struktúrák, pl. a lekérdezési fákhoz van szükség ilyenekre.
  - Alapértelmezett adatbázisok a rendszerstruktúrákhoz: Ebbe vannak az adatok, nagy mennyiség.
  - Rendszer dump és hiba feldolgozó állományok: Hiba mindig van az adatbázisrendszerbe, kell egy olyan hely, ahova ezeket írja.
  - DBA adatbázisok az adminisztráláshoz, monitorozáshoz és hangoláshoz: DBMS-ek nem mindig támogatnak olyan eszközöket, amelyekkel ezek megoldhatók. Pl.: Mentési stratégia, helyreállítási stratégia.
- Memóriakövetelmények: DBMS célja, hogy az adatok elérése gyors legyen, így próbálja az I/O-t elkerülni. Így először a pufferbe ír illetve onnan olvas. SQL utasításokat is tárolhat pufferben. Pl.: Oracle épít a lekérdezésekhez egy lekérdezési fát, és azt is berakhatja a pufferbe. Zárolási kérések, rendezés, folyamatok optimalizálása, SQL feldolgozás.
  A DBMS mindig a pufferhez megy, ha ott nem találja, akkor megy a vinyóra.
- DBMS konfigurálása:
  DBMS rendszerparamétereinek konfigurálása szabályozza azt a módot, amelyben a DBMS funkciók és erőforrások elérhetővé válnak a DBMS számára.
  Telepítéskor meg kell adni egy részét, és nem lehet változtatni. A módosítható paraméterekhez valamilyen eszközt biztosít a DBMS.
- Támogató infrastruktúra szoftverek: Az adatbázis használatához szükséges egyéb szoftverek. Tranzakció feldolgozás, programozási nyelv, web-szerverek, alkalmazásszerverek.
- Telepítés felülvizsgálata: Teszteket kell futtatni, hogy megfelelően működik-e a DBMS a telepítés után.
- DBMS környezet:
  - Tesztelési környezet
  - Minőségbiztosítás
  - Integráció
  - Termelési környezet

DBMS verziók és kiadások frissítése
Főkiadások kiadási ciklusa 12-18 hónap.
Előnyök:

- Új funkciók
- Új alkalmazáshoz kellhet új verzió
- Javul a teljesítmény + elérhetőség
- Jobb a támogatás, a szállító mindig az új kiadásokat támogatja

Hátrányok:
- Általában az üzleti műveletek megszakítását jelenti: Esetleg le kell állítani az adatbázisrendszert.
- Más megszakítás is történhet: Pl.:Adatbázis struktúrát kell konvertálni.
- Költséges: 10-25%-kal is nőhet. Maga a DBMS + a telepítés, tesztelés, esetleg a hardver fejlesztése.
- Rosszabb teljesítményű SQL elérést generálhat, mint előtte: új funkciókat is bele kell venni a fa generálóba, és a régi select-ünk nem működik olyan jól.
- Az előnyök kihasználása érdekében, a DBA-nak lehet hogy néhány módosítást kell végeznie.
- A támogató szoftverek hiányosak lehetnek az új DBMS kiadásokhoz, amikor azt kiadják.

Metaadat kezelés:
Metaadat: Adatokról való adat. Adatok struktúrája, korlátozása, típusa, hosszúsága, szöveges leírása.
Metaadat stratégia: van egy szervezet, cég, akkor fontos, hogy egy metaadat stratégiát kifejlesszen.
- Politika, ami szerint a metaadatokat használják.
- Eljárások az adatok tulajdonosainak és gondnokságának definiálására és azonosítására
- Az összegyűjtendő metaadattípusok azonosítása
- Az egyes metaadattípusok céljának leírása
- A metaadatok tárolására és gyűjtésére szolgáló módszerek
- Módszerek a metaadat elérésére
- Elvek, amelyek végrehajtják az adatgondnoksági eljárásokat és a biztonságot a metaadat eléréséhez: Jogosultságok
- Azonosítani a metaadat forrásokat.
- Mértékek a minőség méréséhez és a metaadat használhatóságához.

DBA-nak ott kell lenni amikor a metaadatstratégiát fejlesztik.

Metaadattípusok:
- Technológiai
- Üzleti

DBA-k rendszeresen használják a rendszerkatalógust:
- Minden adatbázis, tábla, oszlop, index, nézet, kapcsolat, tárolt eljárás, trigger stb. neve
- Elsődleges kulcsok, külső kulcsok
- Melyik tábla melyik nézetben van
- Oszlopok adattípusai, hosszúságai, megszorítások
- Fizikai állományok nevei, és információk róla
- Jogosultságok és biztonsági információk
- Az utolsó DB definíciós változás dátuma és ideje, és a felhasználó ID-ja, aki megvalósította azt
- Adatbázis szervezési információk

A rendszerkatalógus nagyon jó forrása a metaadatoknak, mert aktív, integrált és védett.
Aktív: a metaadatok automatikusan felépülnek és karbantartódnak.
Integráció: együtt jár az aktivitással, mert a technológiai adatokat a DBMS tartja karban (pontosan és naprakészen).
Védettség: a normál DBMS működéssel férhetünk hozzá a rendszerkatalógushoz, és máshogy nem.

Nem található meg a rendszerkatalógusban:
- Metaadat a nem-adatbázis állományokról.
- Módosítási információk: Pl.: csak az utolsó van tárolva, de korábbira is szükségem van
- Adatbázisprogramról információk
- Információk a kötegelt feladatokról és tranzakciókról
- Adatmodell metaadat leírás: pl. a logikai adatbázisterv nincs pontosan leírva.
- Adattárház és ETL (Extract, Transform, Load) metaadatok.
- Az adatok tulajdonlását és gondnokságát leíró metaadatok.

Ehhez találták ki a repository-kat.

Előadás #04 09-09-29

2009. szeptember 29.
8:00

Repository-k: ebbe szoktuk tárolni a metaadatokat, a szervezet adatvagyonát.
Általában tud:
- Információt tárolni az adatokról, a folyamatokról és a környezetről
- Támogatja, hogy ugyanazt az adatot többféleképp tekintsük
- Nagyon alapos doksi
- Támogatja a modell létrehozást és adminisztrációt
- Támogatja a verziókezelést és változás felügyeletet
- Végrehajtja a név konvenciókat: Segítség, hogy hogyan nevezem el a dolgokat.
- Feloldja és kivonatolja a több forrásból származó metaadatokat: Egy oszlopot két táblába máshogy neveznek.
- Jegyzetet generál az adatelem definíciókból

Mit várunk el a repository-tól?
- A repository által használt adattárakat a DBMS-ünkben táblákban tárolhatjuk
- A repositorynak képesnek kell lennie a rendszerkatalógust közvetlenül olvasni: eddig mi van az adatbázisban, és ahhoz képest hozza létre a változásokat.
- Ha mégsem olvassa közvetlenül a rendszerkatalógust, egy interfészt kell biztosítani, amely leegyszerűsíti azt, hogy a repository a rendszerkatalógus információkat olvassa
- A repository biztosít egy interfészt minden modellező és tervező eszközhöz, amely az adatbázis objektumok generálásához használható.

Előnyei:
- Többféle rendszer metaadata egy helyen: Különböző rendszereknél az összefüggéseket könnyű meghatározni.
- Minták felfedezése, használata
- Konzisztencia biztosítása: az adatelemek és az üzleti szabályok között.
- Örökölt rendszer kihasználása: van egy rendszerem, jó doksival, de kell egy másik rendszer, akkor az eredeti doksiból mindent újra tudok használni.
- Gyorsan változó környezet támogatása: Gyorsabban tudok metaadatot keresni a repositoryban, mint az adatbázisban
- Újrafelhasználhatóság

Feladatok:
- Naprakésze tartás: Ha valami változik az adatbázisban, akkor a repositoryt is naprakészen kell tartani.

Honnan kerülnek a repositoryba az adatok?
- Programfejlesztési eszközökből, kódkönyvtárakból
- Üzleti felhasználóktól
- Adatmodellező eszközökből
- Rendszerkatalógusból adatbázis metaadatok
- Adattárház eszközökből, ETL metaadatok
- Automatikus műveletekből és feladat ütemező eszközökből, működési metaadatok
- Adathasználati metaadatok a lekérdező eszközökből

Lehet hogy sok szervezetnél nincs ilyen repository, vagy elavult, akkor sokra nem megyünk vele.

Adat- és tároláskezelés

Az adatoknak táblateret kell biztosítani, egy adatbázisba több táblatér lehet, egy táblatérben több tábla, az se biztos, hogy egy tábla csak egy táblatérben szerepel.

Célok a tárolási rendszer kialakításához:
- A legfontosabb az adatvesztés megelőzése
- A megfelelő kapacitás elérése és a tárolási megoldás skálázhatósága, ha a tárolási igény nő
- Az adatok gyors elérése a szolgáltatás minimális megszakításával, vagy megszakítás nélkül
- Hibatűrő és gyorsan javítható legyen, ha hiba történik
- Leállás nélkül lehessen lemezeket cserélni és bővíteni
- Költséghatékonyság

Állományok és adathalmazok
- Állományok elhelyezése a lemezre:
  - Táblák
  - Indexek: a keresést gyorsítja, el szokták különíteni az adatoktól külön tárhelyre
  - Tranzakciós napló: ezt is el kell különíteni.
- Raw partíció és állományrendszer: Raw partícióba az oprendszer nem szemetel bele, csak a DBMS kezeli. Az OS nem tudja nyomon követni.
- Ideiglenes adatbázis állományok: Egy tranzakció hatásköréig vannak meg ezek az állományok. DBA rendel hozzá lemezeszközt és tárterületet.
- Tömörítés

Miért használunk több lemezeszközt?
- Adatok elkülönítése
- Párhuzamos adatfeldolgozás

Helykezelés

Mit kell követni a DBA-nak?
- Eszköztöredezettség
- Töredezettség használati információ
- Elérhető szabad helyek
- Szegmens és partícióméret
- Szegmensenként foglalt táblák és indexek
- Jelenleg fenntartott területek összege
- Objektumok, amelyek helyhiánnyal küzdenek

Adatlap szerkezet:
- Lapfejléc: Fizikai gazdálkodási információk.
  Tartalmazhat:
  - Lapazonosítót
  - mutatót a következő illetve előző lapra
  - valamilyen azonosítót, ami megmutatja, hogy melyik lap melyik táblához tartozik
  - a szabad tárhoz mutatókat tartalmazhat
  - a táblához minimális sorhosszúságokat.
- Adatsorok: A tábla aktuális sorai. Egy sor általában ráfér egy lapra, kivétel képek, nagy adattípusok.
- Offset tábla: Mutatók, az adatlap egyes sorait címzik. (Nem minden DBMS és nem minden tábla esetén létezik)

Allokációs lapok: A helyhasználat felügyeletére vannak kitalálva. A fizikai lapokat felügyelik. Ezek a lemezen nem egymás mellett helyezkednek el, és az ~ mondják meg, hogy ezek hol helyezkednek el a lemezen.
Adatrekord szerkezetek: Egy sor egy adatlapon kell hogy elhelyezkedjen. A sor kiegészül plusz infóval a tárolás szempontjából, így lesz rekord.
- Sor fejléc: A sor adattartalmának szerkezete és elrendezése.
- Soradat: Az aktuális adattartalmat tartalmazza. Általában a definíció sorrendjében vannak jelen.
- Offset tábla: Mutatók, a soron belüli változó hosszú mezőket kezeli és felügyeli.

A táblaméret kiszámítása: Az adatlapra kerülnek a sorok… Típusokat, hosszakat össze kell adni, változó hossznál átlagos értéket kell venni.

Indexlap szerkezetek: ugyanúgy tároljuk, mint a sorinformációkat, csak kiegészül.
- Fejléc információk
- Sorhossz
- Indexkulcs értékek: A kulcsokhoz tartozó aktuális adatértékek a definíció sorrendjében. (adattartalom)
- Lapmutató: Az adatlap fizikai helyét tartalmazzák.
- Offset és adjust táblák: Változó hosszúságú mezők kezelése/felügyelése.

Az indexméret számítása: Függ az index felépítésétől.

Tranzakció naplók: Folyamatosan nő a mérete, lehet, hogy nagyobb, mint maga az adatbázis.
Mentésnél és helyreállításnál kell, hogy konzisztens állapotba hozhassuk az adatbázist.

Előadás #05 09-10-06

2009. október 6.
8:00

Adatok mozgatása és elosztása

- Adatbetöltés és adatkimentés
  - A LOAD segédprogram
    - Tekintetbe kell venni:
      - Újraindíthatóság: Akkor kérdés, ha hibázik a LOAD.
      - Klaszterek: Valamilyen szempont szerint rendezzük. Egy tábla oszlopában többször el őfordul ugyanaz az adat, és eszerint rendezve akarjuk berakni az adatokat. Lehet hogy érdemes az adatbázison kívül klaszterezni a dolgot.
      - Triggerek
      - Megszorítások
      - Jogok
  - Input állomány leírása:
    - Egyes oszlopok adattípusa
    - Adat kezdő és befejező pozíciója
    - Tagolt állományok(;, stb.)
    - NULL értékek kezelése
    - Adat alapértéke
    - Konvertálás
  - Hatékony betöltés
    - Indexek: Hatékony betöltés ellen van. Betöltés előtt vegyük le az indexet a tábláról, majd ha befejeződött a betöltés, akkor generáljuk újra.
    - Párhuzamos betöltés: Külön tárolók esetében van értelme.
    - Naplózás kikapcsolása: Egyértelműen gyorsabb lesz, de ha helyre kell állítani, akkor a betöltésről semmi nincs naplózva.
    - Kezdő adatok beszúrás: LOAD hatékonyabban tölt kezdőadatokkal, mintha insert-tel töltjük fel.
    - Tömeges törlés: Itt is gyorsabb a LOAD, mintha delete-tel törlünk.
  - Más alkalmazások futtatása a LOAD alatt: Minden már az adatbázisban lévő adat hozzáférhető a LOAD alatt.
  - Az UNLOAD segédprogram: Az UNLOAD célja, hogy adatokat olvasson ki az adatbázisból és azokat egy output állományba írja.
    - Konkurencia: Olvasni mindenképp lehet UNLOAD mellett, a módosításokat meg kell fontolni (inkább ne engedjük).
    - Adat kimentés egy képmásolati mentésből: Azért jó, mert így nem fogjuk vissza az adatbázis teljesítményét, hiszen ezekkel az állományokkal nem dolgozik senki. Ez nem biztos, hogy a legfrissebb, de ezt naplóból lehet javítani.
    - LOAD paraméterek generálása: Valószínű, hogy UNLOAD után máshol ezt az adatbázist LOAD-olni szeretném.
    - Adatkódolási séma: Az adatbázisban egyféle kódolás van, de nekem a kimenetbe másféle kell.
    - Lebegőpontos adat: Ezzel mindig szopás van.
    - Adatkimentés korlátozása: Általában nem egy teljes táblát akarok kimenteni,csak részhalmazát. Pl.: tesztadatnak, vagy csak bizonyos sorokat akarok vizsgálni.
      - LIMIT: Megadhatom, hogy hány sort mentsen ki.
      - SAMPLE: Mintát vegyünk ki. Százalékos értéket kell megadni, hogy az összes adat hány százalékára vagyok kíváncsi.
      - WHEN: Valamilyen feltételnek megfelelő adatokat ment ki.
    - Adatkimentés nézetből
  - Alkalmazás tesztágyak kezelése
- Export és Import: Ugyanaz, mint a LOAD és UNLOAD.
- Nagy mennyiségű adat mozgatása
  - ETL Szoftver: Transzformáció lehet érték transzformáció vagy kódolási, stb.
  - Replikáció és propagáció:
    - Replikáció: Fogom az adatbázist és átmásolom egy másik helyre. Működhet automatizálva.
    - Propagáció: Csak a változásokat fogja migrálni. Ezt megteheti tranzakció napló használatával.
  - Üzenetküldő szoftverek: Más néven üzenet sorbaállító szoftverek. Egy rendszer belerakja egy sorba az üzeneteket, egy másik pedig szedegeti ki. Egyszerű, heterogén, sokrétű kapcsolatot hozhatunk létre. Nem muszáj azonnal fogadni az üzeneteket.
  - Más módszerek

Elosztott adatbázisok

Az elosztott adatbázis teszi lehetővé, hogy az adatok fizikailag teljesen különböző helyeken, különböző adatbázisokban legyenek tárolva úgy, hogy bármely helyen lévő adat bárhonnan elérhető, és módosítható. Jogosultság kezelés van benne.

- Autonómia: A különböző helyeken lévő elosztott adatbázis implementációk milyen mértékben működhetnek egymástól függetlenül.
- Izoláció: Egy elosztott rendszeren belül definiált helye tudnak-e a többi létezéséről.
- Átlátszóság: Arra utal, hogy az adatok helye mennyire védett. Ha átlátszó, akkor a programozónak nem kell tudni, hogy az adot t adat melyik helyen van.

Lehet úgy, hogy egy adatbázist hozok létre és az adatokat több helyre pakolom, vagy több adatbázist kapcsolok össze. Ha több adatbázisom van, akkor szövetkezhetnek, ilyenkor a vezérlést a DBA határozza meg. Ha nincs szövetség, akkor olyan, mintha önálló adatbázisokról beszélnénk.

Elosztott rendszer felállítása

Sok kis gépen futó adatbázis adminisztrációja bonyolultabb, mint ha kevés szerveren fut.
A DBA-nak meg kell mondani a fejlesztőknek, hogy melyik adat hol érhető el, az elérésnek milyen teljesítmény befolyásoló tényezőkkel kell számolni.
Célja:

- Hálózati forgalom minimalizálása
- Blokkos adatküldés soros helyett.

- Távoli adatelérés helyett a helyi adatokat támogatjuk.

Adatelosztási szabványok:

- DRDA (Distributed Relational Database Architecture): Elosztott Relációs Adatbázis Architektúra. Nem kell tudnunk, hogy hol va nnak az adatok. A végfelhasználó úgy látja, mintha az egész egy logikai egységet alkotna. Szabványt biztosít, de API -t nem.
- RDA (Remote Database Access): Távoli Adatbázis Elérés. Távoli kapcsolatot hoz létre a szerver és a kliens között. A kliens eg y olyan folyamattal érintkezik, amely kontrollálja az adatátvitel az adatbázishoz ill. az adatbázisból. Célja: heterogén adatbázisokk al és környezetekkel a kapcsolódást biztosítja.

Elosztott adatok elérése:

- Távoli kérés (remote request): egyetlen helyről egyetlen kérést tartalmaz, amely egyetlen munkaegységben van.
- Távoli munkaegység (remote unit of work): több helyen lév ő adathoz férünk hozzá, de nem egy munkaegységen belül. Ehhez a programozónak tudnia kell, hogy hol vannak az adatok. Minden kérés egyetlen pontot kell hogy érintsen. Minden munkaegység tartalmazhat több SQL parancsot, viszont minden munkaegység egyetlen pontról kell hogy elérjen adatot.
- Elosztott munkaegység (distributed unit of work): Egy munkaegységen belül több DBMS elérhet ő.
- Elosztott kérés (distributed request): Egyetlen SQL utasítás több helyen lév ő adathoz egyszerre hozzá tud férni.

Előadás #06 09-10-27

2009. október 27.
8:00

Kétfázisú COMMIT: elosztott rendszereknél az adatok integritásának a megőrzése a cél.
- Az összes adatbázison alkalmazni kell a COMMIT-ot.
- Az előkészítő szakaszban minden résztvevő felkészül egy COMMIT-ra.
- Minden résztvevő informálja a koordinátort, hogy a naplórekordok sikeresen kiíródtak, és jelzi, hogy kész a COMMIT végrehajtására.
- Ha minden résztvevő készen áll, akkor elkezdődik a COMMIT végrehajtása. Ebben a fázisban sorozatosan kommunikál egymással a koordinátor és az alárendelt résztvevők. Ha egy résztvevő azt mondja, hogy sikertelen a COMMIT, akkor a koordinátornak az összes helyen vissza kell vonatni a COMMIT-ot.

Teljesítmény: Az adatbázis teljesítménye annak az erőforrásnak optimalizálása, amit az átbocsátó képesség növelésére használunk és a verseny minimalizálására, megengedve ezzel a legnagyobb lehetséges munkafolyamat végrehajtását.
Elosztott rendszerek teljesítmény problémái: Az adatforgalom teljesítménye okozza a legnagyobb problémát.
Mit kell megvizsgálnia a DBA-nak
- A számítógép hardvere
- A helyi OS
- A hálózati rendszer
- A kérés küldő helyi adatbázisrendszer
- A hálózati hardver
- Bármilyen köztes termék, vagy tranzakció feldolgozó rendszert, amit a kérés küldő, vagy a szerver használ.
- A lemezre mentés valamint a mentést kezelő szoftver

Adatbázis biztonság:
Ahhoz hogy valamilyen műveletet elvégezzünk egy adatbázisban, kell hogy explicit jogot kapjunk hozzá, vagy mindenkinek legyen hozzá joga. Azt hogy kinek milyen joga van repository-ban érdemes tárolni. Biztonsági politika kell egy cégnél!
A jelszót bizonyos időközönként meg kell változtatni, ezt ki lehet kényszeríteni.
Milyen eszközökkel lehet korlátozni egy login-t:
- Hibás bejelentkezési próbálkozások száma
- Napok száma, amíg a jelszó érvényes
- Napok száma, amíg az account zárolva marad, ha a jelszó lejárt
- Jelszó újrahasználhatósága
- Minimális hosszúság, ill. alfanumerikus követelmények

Jogok adása és visszavonása (DCL):
GRANT: Milyen jogot és kinek adom. WITH GRANT OPTION-nel tovább lehet adni a jogot.
- Nem központi adminisztráció: Tovább lehet adni a jogokat, ezt sokkal nehezebb felügyelni.
- Központi adminisztráció: A DBA adja a jogokat, sok feladat hárul rá.

REVOKE

- A jogok típusai:
  - Táblára vonatkozó jogok:
    - GRANT SELECT ON tábla[(oszlop)] TO felhasználó
    - INSERT
    - DELETE
    - UPDATE
    - ALL
  - Adatbázis objektumra vonatkozó jogok: Ki hozhat létre és dobhat el. Ezeket a DBA hozza létre ezeket a jogokat.
    - TABLE
    - VIEW
    - INDEX
    - PROCEDURE: tárolt eljárás
  - Rendszerre vonatkozó jogok: Melyik felhasználó mit futtathat a DBMS-ben
    - Archív adatbázis naplózás
    - Adatbázis szerver leállítása, újraindítása
    - Nyomkövetés: GRANT TRACE TO felhasználó
    - Tárolás kezelése
    - Adatbázis gyorstárak kezelése
  - Programra vonatkozó jogok
    - GRANT EXECUTE ON eljárás TO felhasználó
  - Tárolt eljárásra vonatkozó jogok: ki futtathat
- Jogok adása PUBLIC-nak: Mindenkinek odaadom a jogot. Nagy rendszernél nem érdemes, mert nehéz a biztonság adminisztrálása.
- Jogok visszavonása: REVOKE jog FROM felhasználó
  - Kaszkádolt visszavonás: Ha WITH GRANT OPTION-nel adtam jogot valakinek és visszavonom tőle, akkor azoktól is visszavonom, akiknek ő adott.
- Biztonsági riportok: Monitorozni és riportolni kell az adatbázis jogait.

Szerepkörök és csoportok feljogosítása
- Szerepkör: Jogokat tudok adni a szerepkörnek, és ezt tudom odaadni a felhasználóknak. Ha utólag a felhasználónak még kell jogot adni, akkor elég a szerepkörnek odaadni.
  CREATE ROLE szk ;
  GRANT CREATE SEQUENCE TO szk;
  GRANT szk TO felhasználó;
- Csoportok:
  - Rendszer adminisztrátor: SA vagy SYSADM.

        - Adatbázisparancsokat futtathat
        - Minden objektumot elér
        - Telepítésért felelős
        - Rendszerkatalógusok és táblák tulajdonosa
    - Adatbázis adminisztrátor: DBA vagy DBADM.
        - Elvileg minden joggal rendelkezik
        - De letilthatják a táblában lévő adatok módosításáról.
        - Általában 1-1 adatbázishoz tartozik
    - Adatbázis karbantartó: DBMAINT
        - Segédprogramok futtatása, parancsok kiadása
        - Általában 1-1 adatbázishoz tartozik
    - Biztonsági adminisztrátor: SSO
        - Jogok adásához, visszavonásához szükséges jogok
        - Mindenféle biztonsághoz kapcsolódó tevékenységhez jogot kap, pl.: auditálás.
    - Műveletek (üzemeltetés) felügyelője: OPER vagy SYSOPER
        - Mentési, helyreállítási jog

Más adatbázis biztonsági mechanizmusok

- Nézetek használata a biztonsághoz: Nem a táblára adok olvasási jogot, hanem létrehozok hozzá egy nézetet.
    - CREATE VIEW d1 AS
      SELECT e_name, mgr
      FROM DOLGOZO;
    - CREATE VIEW d2 AS
      SELECT *
      FROM DOLGOZO
      WHERE dept_no=2;
    - A d1-et vertikális, míg a d2-t horizontális megszorításnak hívjuk.
    - WITH CHECK OPTION: csak a feltételnek megfelelő sorokat engedi letárolni.
- Tárolt eljárások használata a biztonsághoz: A mögöttes táblákat a tárolt eljárás fogja módosítani, a felhasználónak a mögötte s táblák módosítására nincs joga, de a tárolt eljárásra van futtatási joga, így a felhasználó csak azokat a módosításokat tudja elvége zni, amire a tárolt eljárásnak joga van.

Előadás #07 09-11-03

2009. november 3.
8:00

Auditálás: Követhetővé teszi az adatbázis erőforrásainak és jogainak használatát. Ha engedélyezve van, akkor audit sort fog termelni a DBMS.
Audit sorban szerepel: milyen objektumra volt hatással a művelet, ki teljesítette a műveletet, és mikor tette ezt.
Különböző szinteken lehet kérni, adatbázis szint, adatbázis-objektum szinten, felhasználói szint.
Ez teljesítmény csökkenéshez vezet.

- Képességek DBMS-ben:
  - A be és kijelentkezési próbálkozások
  - Adatbázis szerver újraindítása
  - A rendszer adminisztrátori jogosultsággal rendelkező felhasználók által kiadott parancsokat
  - Integritás megsértésének próbája (SQL Inject-et is ide soroljuk)
  - SELECT, INSERT, UPDATE, DELETE műveleteket
  - Tárolt eljárás futtatások
  - Sikertelen próbálkozások egy adatbázis vagy tábla elérése (jogosultság megsértése)
  - Rendszerkatalógus táblák változása
  - Sor szintű műveletek
- Tanácsok, ha auditálunk
  - Rendszererőforrást nagyon fogyasztja
  - Háttértár kell hozzá, pl. ha nincs hely, akkor megbénulhat az adatbázis
  - Érdemes külön eszközre helyezni, hogy csökkenjen a versengés

Külső biztonság
A DBA feladata ez is. OS szinten is meg kell oldania

- Rendszerkatalógus adatállományok
- Aktív és archív napló állományok
- Felhasználói adathalmazok a táblaterekhez
- Felhasználói adathalmazok az indexekhez
- Auditálási adatállományok
- Teljesítmény követési állományok
- Program és szkript állományok (forrás és futtatható kód)

Titkosítani lehet ezeket az állományokat, illetve kell egy OS szintű védelem az adatbázishoz.

Feladatütemezés és biztonság: Ha egy külső cég terméke intézi az ütemezést, akkor ehhez kell egy felhasználó, aminek nem szabad túl nagy jogkört adni.

Nem DBMS DBA biztonság: OS szintű biztonság, a DBA nem feltétlenül kapja meg a root jelszót.

Adatbázis mentés és helyreállítás

Az adatbázis hibákat 3 kategóriába sorolhatjuk:

- Példány hiba: Valamilyen OS hiba vagy szoftver hiba vagy belső kivétel eredménye. Kevés esetben jelent adathibát. Le kell állítani az adatbázist, ki kell javítani a hibát és újra kell indítani.
- Alkalmazás (vagy tranzakció) hiba: Szkript rossz időben fut, rossz inputtal, vagy rossz sorrendben. Ez hibás adatot eredményez. Minél hamarabb azonosítani kell a hibát.
- Média hiba: Lemeztárolási eszköz hiba, állományrendszer hiba, szalag károsodás, törölt adatállományok, stb… Ezek is károsítják az adatainkat. Ilyen hiba esetén olyan állapotba kerülhet az adatbázis, hogy az érvényes adatok olvashatatlanok lesznek, vagy megsérül a hivatkozási integritás, vagy érvénytelen adatokat tudunk kiolvasni. Erre a legjobb megoldások: RAID, tükrözött rendszer, UPS.

Képmásolati mentés: Az adatok egy konzisztens másolatát kell létrehozni. Ezt megtehetjük COPY, BACKUP, DUMP, EXPORT segédprogrammal, vagy az OS szintű állományt mentjük le. A DBA-nak meg kell győződnie arról, hogy a mentés pontos és érvényes. Azt is meg kell határoznia, hogy milyen gyakran készítsen képmásolati mentést, és hány mentési generációt kell megtartani. Ezekhez hozzá tartozik, hogy a megfelelő napló rekordok elérhetőek legyenek, vagy mentve legyenek helyreállítási célra. A naplónak a mentés elejétől meg kell lennie, ebből tudom meghatározni, hogy milyen konzisztens állapotom volt az adatbázisban. A mentés gyakoriságának meghatározásakor figyelembe kell venni, hogy mennyi idő kell az egyes objektumok helyreállításához.

Az, hogy maga a helyreállítás mennyi ideig tart, a következő tényezőktől függ:

- Naplórekordok száma, melyet a helyreállításkor fel kell dolgozni
- A napló tömörített-e
- A szükséges szalagok csatolásához és lecsatolásához szükséges idő
- Mennyi ideig tart a helyreállításhoz szükséges naplórész elolvasása
- Mennyi ideig tart a változott oldalak újra felolvasása
- DBMS felépítése, van olyan, ami az összes naplót olvassa és van olyan, amelyik csak a szükséges naplót olvassa a helyreállításhoz

Ha mentési pontig állítunk helyre, akkor kevesebb ideig tart.
Minél gyakrabban készítettünk képmásolatot, annál rövidebb ideig tart a helyreállítás.

Vezérelvek a helyreállítási környezet biztosításához:

- Legalább 2 teljes mentési generációt meg kell tartani.
- Kell egy helyi mentési stratégia, amelynek összhangban kell lennie a katasztrófa helyreállítási stratégiával.
- A mentés gyorsításhoz először lemezre készítsük el a mentést, és azt írjuk ki szalagra.
- Tömöríthetjük a mentéseket
- A mentési terv tartalmazza a rendszerkatalógus adatbázis objektumokat is.
- Biztosítsuk, hogy a mentési folyamat újraindítható legyen.

- A mentés után bizonyosodjunk meg róla, hogy helyes a mentés.
- A nem adatbázisban tárolt adatokat is ugyanabban az időben menteni kell.

Ökölszabályok:
- A napi üzletet ne akadályozza, de a helyreállításhoz szükséges idő is megfelelő legyen.
- Ha átszervezem a rendszert, érdemes teljes mentést végezni.
- Ha napló nélkül töltök adatot az adatbázisba, akkor is érdemes mentést végezni.
- Ha point-in-time helyreállítás van, akkor is érdemes mentést végezni.

Képmásolati mentés:
- Teljes vagy inkrementális mentés: A teljes mentés sokáig tarthat, az inkrementális kevesebb, de ehhez is kell a napló. A helyreállítás lassabb ha inkrementális mentést végzek. Akkor javasolt az inkrementális, ha az adatbázisban kevés változás történt (az adatblokkok 30-40%-ánál kevesebb változik). Kis adatbázisok esetén a teljes mentés a kedvelt. UNIX alapon ez maximum 100 GB. Naplózás nélküli változásoknál a teljes mentés javasolt. Van olyan eszköz amely egy teljes és egy inkrementális eszközből csinál egy új teljes mentést.
- Adatbázis objektumok és mentések: Az indexet vagy mentem, vagy helyreállításkor újraépítem. Ezek úgyis külön táblatérben vannak. Mitől függ, hogy mentem-e? Időtől (helyreállításnál) és tárigénytől (mentésnél). Nagyobb táblák és többszörös indexek esetén érdemes menteni. Feleljenek meg az adatoknak mentésnél, egyébként érvénytelen vagy hibás eredményeket kaphatunk vissza.
- Párhuzamos elérés: A mentés mellett elérhetőek-e az adatok. Ha elérhetőek az adatok, akkor a naplóra is figyelni kell. Lehet olyan átmenet, hogy az adatok csak olvashatóak, így nem történik változás és nem kell napló.
  Egy megfelelő stratégiát kell kidolgozni a DBA-nak, figyelembe véve a mentési eszköz működését, illetve a következőket:
  - Párhuzamos elérések és módosítások szükségessége a mentési folyamat alatt
  - A mentési folyamatra szánható időmennyiséget
  - A párhuzamos elérések hatását az adatok mentésének gyorsaságára
  - A helyreállítási segédprogram sebességét
  - Az adatbázisnaplók elérésének szükségességét

  Forró mentés: az adatbázis online marad.
  Hideg mentés: le kell állítani bizonyos állományokat.
  A forró mentés a problémásabb, mert nehezebb megvalósítani, nagyobb a CPU, I/O igénye, archiválni kell a naplót, a DBA-tól szkriptek írását igényelheti, illetve alaposabb tesztelést igényel.
- Mentési konzisztencia: Amikor mentési tervet készítünk, akkor a helyreállításkor mindig konzisztens állapotot kell létrehozni. Ehhez az kell, hogy minden adatbázis objektumot lementsünk. Ebbe bele tartoznak a hivatkozási megszorítások, triggerek, kapcsolatok.
  Van olyan segédprogram, amely az adatbázisnaplóba ment egy konzisztencia pontot. Mikor kell ezt megtenni:
  - Aktív napló archiválása előtt.
  - Kapcsolódó objektumok másolása előtt.
  - Képmásolati mentés elkészülése után.
  - Súlyos adatbázis módosítás végrehajtása előtt.
  - Csöndes időszakban.
- Napló archiválás és mentés: Ha az aktív napló betelik, akkor archiválni kell. Ilyenkor az aktív napló tartalma törlődik. Az archiválás automatikus, de a DBA tud kézzel archiválni.
- A mentési ütemezés meghatározása: Elég gyakori képmásolatot készítsünk, de mindez ne zavarja a napi üzletet.
  Kérdések:
  - Mennyire használják az adatot egy nap
  - Milyen gyakran változik az adat
  - Mennyire kritikus az adat
  - Könnyen újra létre lehet-e hozni az adatot
  - Milyen költsége van annak, ha nem érhető el az adat
  - Mennyi a pénzértéke, ha a rendszer egy percig áll
  - Milyen típusú elérés szükséges az adatokhoz (24/7 vagy le lehet állítani)

  A kritikus adatokat gyakrabban kell, az illékony adatokat szintén
- DBMS példány mentése: DBMS állományok, rendszerkatalógus, könyvtárobjektumok, konfigurációs állományok, rendszerkönyvtárak, szalagkezelő könyvtárak, különböző programokhoz forrás- és futtatható állományok. Akkor lehet kritikus ennek a helyreállítására, ha eszközhiba történik, frissítjük a DBMS-t vagy emberi hiba történik.
- A DBMS környezet tervezése a helyreállításra: Ha van többszörös adatbázis napló, akkor azt használjuk ki.
- Az adatbázis mentés más megközelítései: Lehet UNLOAD-ot használni (vagy EXPORT-ot). Valamilyen logikai mentést készítünk.
  Akkor érdemes használni:
  - Ha objektumot, vagy sort akarunk visszaállítani.
  - Ha verziót frissítünk, akkor is érdemes logikai mentést csinálni, és az adatokat vissza tölteni az új verzióba.
  - Heterogén adatbázis migráció: Különböző platformok közötti adatbázis mozgatás. Ok pl: különböző az OS struktúra.
  - Adatmozgatás

  Lehet tárolás kezelő szoftvert használni: a DBMS vezérlésén kívül történik a mentés. Lockolni kell az objektumot, mert ha nem tesszük, akkor nem lesznek az adatok konzisztensek.
- Mentési stratégia dokumentálása
- Adatbázis objektum definíciók mentése: Az ezekhez tartozó SQL kódokat is mentsük el, mert helyreállításnál szükséges lehet.

Előadás #08 09-11-10

2009. november 10.
8:05

Helyreállítás

A múltban volt egy időpillanat, amikorra az adatbázis állapotát vissza kell hogy állítsam.

A helyreállítási opciók meghatározása:
- Fel kell ismerni, hogy hiba történt.
- Milyen helyreállítás a legjobb, amit elvégezhetünk, ehhez tudni kell a hiba okát és terjedelmét

Kérdések:
- Milyen hiba történt?
- Mi a hiba oka?
- Hogyan állt le az adatbázis, normal, abort, crash?
- Történt-e OS hiba?
- A szerver újraindult-e?
- Vannak-e hibák, az alert.log-ban?
- Milyen dump készült?
- Generálódott-e nyomkövetési állomány?
- Mennyire kritikusak az elveszett adatok?
- Próbáltak-e már helyreállítani, és ha igen, akkor milyen hatása volt?
- Milyen típusú mentés létezik, teljes, inkrementális, mind2?
- Mit kell helyreállítani, a teljes adatbázist, egy táblateret, egy táblát?
- A mentési stratégiánk támogatja-e a szükséges helyreállítás típusát?
- Ha hideg mentésünk van, hogy állt le az adatbázis, amikor a mentés készült?
- Minden archivált adatbázis napló elérhető-e a helyreállításhoz?
- Van-e logikai mentésünk?
- Milyen párhuzamos tevékenységek voltak, amikor az adatbázis összeomlott?
- A DBMS példányt újra lehet-e indítani?
- Elérjük az adatbázis objektumot?
- Milyen a rendszer elérhetőségi követelménye?
- Mennyi dolgot kell helyreállítani?
- RAW állományokat használunk-e?

Általános lépések az adatbázis objektum helyreállításához:
1. Azonosítsuk a hibát: nem válaszol, hibaüzenetet ad
2. Elemezzük a helyzetet: mi a hiba oka, típusa, hatásköre, illetve milyen helyreállítási módot kell használni.
3. Mit kell visszaállítani: mely adatbázis objektum hibás, és el kell készíteni egy helyreállítási scriptet.
4. Azonosítani kell a függőségeket a visszaállítható adatbázis objektumok között. A függő objektumokat is vissza kell-e állítani.
5. A szükséges képmásolati mentések elhelyezése.
6. Képmásolati mentésekből a helyreállítás.
7. Előregörgetés az adatbázis napló szerint.

Helyreállítás típusai:
- Helyreállítás az utolsó időpontig: Katasztrófa utáni helyreállítás. Természeti katasztrófa, média hiba, ami elpusztítja az adatközpontot. A katasztrófa előtt még volt adatbázis, oda akarok visszaállítani. Ehhez kell egy teljes képmásolati, ha volt, akkor inkrementális mentés, és a naplóból helyreállítok. Minél több naplót kell előregörgetni, annál több időt vesz igénybe a helyr eállítás.
- Point in time helyreállítás: Egy adott időpillanatra fog helyreállítani. Alkalmazás szintű problémák kezelésére szolgál. Részleges helyreállításnak is nevezhetjük. Egy adott időpont óta történt tranzakciók hatását eltávolítja. Valamilyen konzisztens állapo tot kell elérnünk.
  - Teljes képmásolati mentésből helyreállítok, majd a napló alapján előregörgetek.
  - A jelenlegi állapotból a napló alapján visszagörgetek.

  Ez attól függ, hogy melyik kevesebb idő.
- Tranzakció helyreállítás: Egy tranzakció véglegesített hatásait kell eltávolítanom. Ha nem futott másik tranzakció az adataimon, addig a hibás tranzakció hatását el lehet távolítani.
  - Point in time helyreállítást végzünk, és utána az érvényes munkákat újrafuttatjuk. Ez általában egy adott adatbázis objektumra vonatkozik.
  - Undo helyreállítás: Csak a rossz tranzakció hatásait távolítjuk el. A naplóból generálok egy anti SQL-t, és az ezekből készített scriptet lefuttatom.
  - Redo helyreállítás: Minden tranzakciót el kell távolítani, ami egy adott id őpont után történt, és csak a jó tranzakciókat újra alkalmazni.
- OS szintű helyreállítás: Ha tároláskezelő szoftvert használtunk a mentéshez, akkor a visszaállításhoz is azt kell használnunk.
- Offside katasztrófa helyreállítás: Ha valami baleset vagy természeti katasztrófa tönkreteszi az els ődleges adatfeldolgozó központot, akkor átköltözök valahova.

Optimális helyreállítási stratégia választás:
Legtöbbször alkalmazás vagy emberi hiba van, leállást is inkább a szoftverhiba okozza, mint a hardver hiba.
Kérdések:
- Tranzakció azonosítás: a problémás tranzakciókat lehet-e azonosítani, lehet-e a munkát újra végrehajtani.
- Adatintegritás: módosította-e a sorokat más, mióta a probléma történt, elérhető-e minden szükséges adat, volt-e azóta újraszervezés vagy betöltés, ha helyreállítok, akkor a másolat elveszik-e.
- Sebesség: melyik a leggyorsabb technika, mennyi adatbázis naplót kell a helyreállításhoz használni.
- Elérhetőség: milyen hamar érhető el ismét az alkalmazás, elkerülhető-e az offline mód.
- Támadóképesség: hiba mennyire hat az adatbázisra, születtek-e rossz döntések a hibás adatok alapján.

Mi rövidítheti meg az időtartamot?

- Minél kisebbek a komponensek, annál rövidebb ideig tart a helyreállítás.
- Ha vannak objektum partíciók, és ezeknek külön mentése, akkor lehet hogy a hiba csak egy -egy partícióra korlátozható, és a helyreállítás is gyorsabb lesz.
- Ha lemezen van a mentés illetve a napló, akkor a helyreállítás gyorsabb.
- Teszteljük a képmásolati mentéseket, hogy érvényesek-e, mert ha érvénytelennel próbálok meg helyreállítani, akkor lelassul a helyreállítás.
- Ha automatikus a mentés és a helyreállítás, akkor a manuális hibákat kizárjuk, így csökken a leállási id ő.
- Minél kevesebb legyen az adatbázis tervben a függőség, mert az önálló objektumokat könnyebb helyreállítani.

A hibák típusához megfelelő helyreállítás használata:
- Média hiba esetén az utolsó pontig
- Tranzakció hiba esetén point in time vagy tranzakció helyreállítás
- Adatbázis példány vagy alrendszer hiba esetén a hiba el őtti utolsó pontig

Index helyreállítás:
- Újraépítjük az indexet.
- Van mentés és csak helyre kell állítani: itt figyelni kell, hogy ne legyenek érvénytelen indexek. Sok adatnál ez a jobb.

Helyreállítási terv tesztelése: Dokumentációt kell írni minden hiba esetén történ ő helyreállításhoz. Benne kell hogy legyen minden adatbázis objektumhoz tartozó mentési és helyreállítási script. Személyek nevét, telefonszámát kell hogy tartalmazza, akiket a helyreállításhoz lehet hívni. Naprakészen tartás fontos, mert ha adatbázis objektumot módosítunk, akkor lehet, hogy a tervne k változnia kell. Minden féle hibára el kell készíteni és le kell tesztelni a tervet.

Egy eldobott adatbázis objektum helyreállítása: Ha logikai mentésünk van, akkor létrehozzuk az adatbázis objektumot és logika i mentésből visszatöltjük. Nem csak az objektumot, hanem a jogosultságokat, hivatkozási - illetve ellenőrző megszorításokat is vissza kell állítanom.

Teszt adatbázis létrehozásához az egyik legjobb mód, ha a termelésit egy másik gépen helyreállítom.

Előadás #09 09-11-17

2009. november 17.
8:04

Alternatívák a mentésre és a helyreállításra

Tartalék (Standby) adatbázisok: Az online-nal azonos másolat, majdnem naprakészek. Azért nem naprakész, mert a frissítések hatása késleltetve jelenik meg. Hiba esetén a vezérlés átkerül ide, és itt folytatódik a normál működés. A normál mentést nem helyettesíti, főleg azért nem, mert a hibák is átkerülnek. Ez főleg katasztrófa esetén hasznos.

Másolat (Replication): Lehet az eredeti adatbázis egy részhalmaza. Megvalósítható egyszerű tábla másolással is. A DBMS-ek nagy része biztosít automatikus replikációs eszközt.
Kétféle technika létezik:

- Szimmetrikus másolat: Az adatokat naprakészen tartja, mert ha adatot módosítok, akkor a tranzakció akkor lesz véglegesítve, h a a másolaton is megtörtént a módosítás. Megoldható, hogy a frissítés később történjen meg, de mindenképp automatikus.
- Pillanatfelvétel (snap shot) másolat: Lekérdezésen alapszik, a lekérdezés eredménye kerül egy táblába. Cél lenne, hogy naprak ész információt tartalmazzon, de ha valamilyen adatok frissülnek, akkor ennek is frissülnie kell. Ha egy adatról több ilyen van, akkor érdemes ezeket egyszerre frissíteni.

Mindkettő teljesítményproblémát fog okozni, mert amikor készülnek a másolatok, azok erőforrás igényesek. Ezek készülhetnek ugyanabban az adatbázisban, így redundáns adataink vannak. Mentést nem helyettesít.

Lemeztükrözés: Nem adatbázisszinten történik, hanem lemez (OS vagy HW) szinten. Ha az elsődleges lemezeszköz sérül, akkor a másodlagost lehet használni. Médiahibát lehet vele kiküszöbölni, és a mentést ez sem helyettesíti.

Terv katasztrófára

Katasztrófa: egy nem tervezett kiterjedt veszteség a kritikus üzleti alkalmazások területén, mely a számítógép feldolgozó-kapacitásának hiánya miatt történik, és több, mint 48 órára fennáll.

Katasztrófa terv: Előkészületi folyamat, ha katasztrófa következik be, akkor hogyan fogom visszaállítani az adatbázist. A lehetséges katasztrófákra fel kell készülni. Az adatbázis helyreállítás csak egy része a teljes üzleti helyreállításnak. Az alkalmazottakat értesíteni kell, az ügyfelek értesítését is meg kell oldani.

Kockázat és helyreállítás: A katasztrófa esetén előforduló költségek minimalizálása a célja a tervezésnek.
Meg kell határozni, hogy az adatveszteség milyen kockázattal jár.
A DBA-nak az egyes adatbázis objektumokhoz becslést kell végrehajtania, hogy ezek mennyire kritikusak.
Kockázat szempontjából három kategória van:

- Pénzügyi veszteség
- Üzleti szolgáltatás szünetelés
- Jogos kötelezettség

Másik csoportosítás:

- Nagyon kritikus: Ezt kell először helyreállítani. Pl.: telefonos cégnél a szolgáltatás ilyen, míg a számlázás csak üzleti kritikus.
- Üzleti kritikus: Elég 1-2 nap múlva helyreállítani.
- Kritikus: Elég 1-2 hét múlva is helyreállítani.
- Szükséges
- Nem kritikus

Általános katasztrófa helyreállítási irányelvek:

- Távoli oldal: Itt fogjuk a cég műveleteit, adatait tárolni. Elég messze kell lennie az elsődleges helytől. Ha a cég elég nagy, és van két adatközpontja, akkor lehet az, hogy az egyik központ mentéseit a másik helyre mentem és vica-versa.
- Írott terv: Minden kulcs embernek meg kell kapnia, a helyreállítási oldalon is kell belőle egy példány. Ezt naprakészen és relevánsan kell tartani. Ha változik a terv, akkor a régit megsemmisítjük és lecseréljük az újra.
  Bizonyos feltételeknek eleget kell tennie:
  - Explicit műveleteket kell hogy tartalmazzon. Mit kell tenni, ha bekövetkezik a katasztrófa.
  - Megfelelő sorrendben kövessék egymást a műveletek.
  - Határozzuk meg az eszközt, amivel dolgozni kell, és a szükséges pontos mentési információt.
  - Dokumentálja, hogy a szükséges információk hol vannak tárolva, és a helyreállítás helyéről hogyan lehet elérni őket.
  - Ha aki ismeri a tervet, nem érhető el, akkor más is eligazodjon a terven.

  A tervnek a következőket kell tartalmaznia:
  - Helyszínek: Távoli helyszínek címe, telefonszám, faxszám, a közeli hotel címe
  - Személyek: Nevek, elérhetőségek. DBA-k, rendszeradminisztrátorok, végfelhasználók, főnök.
  - Hitelesítés: A biztonsági hitelesítések listája és a személyek, akikhez rendelve van.
  - Helyreállítási eljárások és scriptek: Minden rendszerszoftverhez, alkalmazáshoz és adathoz. Teljes step-by-step eljárások. Megfelelő sorrend, telepítési CD-k, vagy, hogy honnan lehet beszerezni.
  - Riportot mentési szalagokról, azok tartalmáról, mikor küldték az elsődleges helyszínről, mikor érkezett meg.
- A katasztrófaterv tesztelése: Érdemes legalább évente egyszer tesztelni, illetve ha a következők történnek:
  - Jelentős napi művelet változás
  - Rendszer HW konfiguráció változás
  - DBMS frissítés
  - Helyreállításért felelős személy megváltozása
  - Az elsődleges adatközpont más helyre költözése
  - A napi mentési eljárásokban változás
  - Új alkalmazások, vagy meglévő kritikus alkalmazások frissítése esetén
  - Jelentős adatmennyiségbeli vagy napi tranzakcióbeli növekedés

  A tesztelést arra használjuk, hogy megtaláljuk a hibáit és gyengeségeit és javítsuk is ki őket. A tesztelés legyen életszerű, tehát a főnöknél kell kezdeni.

Az adatbázis mentés a katasztrófa helyreállításhoz

- Képmásolati mentés: Képmásolati mentést a helyreállítási oldalra minél hamarabb oda kell küldeni. Az indexeket nem muszáj men teni. Minden mentési állományról készüljön napi riport. Ebből tartsunk egyet a helyi és a távoli oldalon is. A riportnak minden szükséges információt részleteznie kell a helyreállításhoz, beleértve az állománynevet, a mentés típusát, hogyan készült a képmásolat, a mentés napja és ideje, az átküldés napja és ideje és az érkezés napja és ideje. Mentés mellé át kell küldeni a naplót is, mert anélk ül a konzisztencia sem biztos, illetve az utolsó naplóbejegyzésig tudunk helyreállítani. Helyreállítás után a DBA -nak az adatintegritást ellenőriznie kell. Ezekből az utolsó hármat tartsuk meg.
- Mentés az adatbázis leállításával:
  - Leállítjuk az adatbázist.
  - Lementünk minden adatbázis objektumot
  - Ha sikerült, akkor újraindítjuk az adatbázist
  - Kiírjuk a mentést, és elküldjük a távoli oldalra
- Más megközelítések: Pl.:Távoli tükrözés
- Néhány irányelv:
  - A helyreállítás sorrendje
  - OS és szoftver verzió megfelelő legyen, és a megfelelő karbantartási szinten legyenek telepítve.
  - Adatlappangás: Milyen régi az adat? Éjszakai mentés esetén 24 órás lehet, és ilyenkor naplóból tudunk helyreállítani.
  - Emlékezés más létfontosságú adatokra: alkalmazásokhoz kapcsolódó állományok:
    - DDL könyvtárak az adatbázis-objektumokhoz, a mentéshez
    - Alkalmazás program források és futtatható állományok
    - Tárolt eljárásokhoz források
    - Felhasználó által definiált függvényekhez források
    - Futtatható állományok
    - Könyvtárak és jelszavak más szállítótól származó DBA eszközökhöz
    - Alkalmazás által használt kapcsolódó adatok
  - Tömörítés: a programunk ugyanaz legyen, mint amivel betömörítettük.
  - Utómentési képmásolat: Ha befejeztük a katasztrófa után a helyreállítást, akkor készítsünk egy képmásolati mentést.
- Katasztrófa megelőzés:
  - Áramszünet esetén UPS
  - Végfelhasználók tanítása, dokumentáció készítésével az emberi hiba megelőzése.

Adatok elérhetősége

Ha nem elérhetők az adatok, akkor a programok nem futnak, az üzlet veszteséget szenved. A DBA a felelős az adatbázis elérhetőségéért, azért, hogy az adatbázis online legyen.
24/7-es elérhetőség: ez általában a weben elérhető adatbázisokra vonatkozik.

Az elérhetőség meghatározása:

- Az erőforrás elérhető az ügyfelek számára. A felhasználók elérik az adatokat.
- Az időtartam százaléka, mely során a rendszer produktív munkára használható.

Az adatbázis teljesítmény nem egyenlő az elérhetőséggel. Ha a teljesítmény gyenge, akkor a felhasználó ugyanúgy nem tudja elérni az adatbázist, mintha az adatbázis nem elérhető.
Ha az adatbázisunk áll, akkor sem elérhető, de ilyenkor teljesítményről nem is beszélhetünk.
Négy részből tevődik össze az elérhetőség:

- Kezelhetőség: A hatékony környezet létrehozásának és fenntartásának képessége, amely szolgáltatásokat nyújt a felhasználók számára.
- Visszaállíthatóság: A szolgáltatás helyreállításának a képessége.
- Megbízhatóság: A szolgáltatás nyújtásának képessége meghatározott szinten egy adott id őtartamra vonatkozóan.
- Szervizelhetőség: A fennálló problémák meghatározásának képessége, azok okainak megállapítása, és a problémák helyreállítása.

Előadás #10 09-11-24

2009. november 24.
8:00

Megnövekedett elérhetőségi követelmények: A rendszerek egyre inkább megkövetelik a 24/7-es elérhetőséget.
- Csökkenő karbantartási idő: Adatbázisok töredezettség mentesítése, újraszervezése.
- Döntéstámogatás
- Adattárházak: Teljesítményre hatással van, mert adatokat kell kivenni meglévő rendszerekből.
- Állandó elérhetőség: A 24/7-es elérhetőséget fel lehet cserélni 24/24-esre, ez akkor van, ha több időzónában dolgozik a rendszer.

Az állásidő költsége
Amikor ezt becsüljük, figyelembe kell venni a következőket:
- Leállás során elveszett üzlet
- Annak a költsége, amíg a rendszerben helyreáll a rend miután a rendszerek ismét elérhetőek
- Perek jogi költsége
- Csökkent részvényérték hatása
- Sajtó negatív ill. pozitív hatásai

Általában a vezetőség nem tudja, hogy a leállás mivel jár.

Mennyi elérhetőség az elegendő?
- 99%, kevés, egy évbe 87 órát állhat
- MTBE (Mean Time Between Failure - Meghibásodások közötti átlagos időtartam)
- 100%-os nincsen, de használnak olyat, hogy 99.999%, ez 5 perc évente. Ennek eléréséhez jól felépített adatbázis rendszerre van szükség. Nagy tervezés szükséges a megvalósításhoz.

Elérhetőségi problémák:
- Az adatközpont elvesztése: Erre van a katasztrófa helyreállítás. Helyreállításkor nem biztos, hogy naprakész lesz az adatbázis.
- Hálózati problémák: Nem a DBA feladata a megoldása.
- A szerver hardver elvesztése: Lehet klaszterezést vagy tartalékrendszert használni ellene.
- Lemezzel összefüggő leállások: Fizikai meghajtó mechanizmus meghibásodik, vezérlő meghibásodik, egy vezeték megsérül. RAID, SAN, NAS használata lehet megoldás.
- OS meghibásodás: Valamilyen bug, vagy verziófrissítés.
- DBMS szoftverhiba: Valamilyen bug, verziófrissítés, nem érhetőkel az indítási paraméterek, napló állomány sérült, vagy hiányzik.
- Alkalmazási problémák: könyvtárak elvesztése, bugok, stb.
- Biztonsági és engedélyezési problémák: Ezek általában emberi hibák. Pl.: ha a felhasználó nem kap jogot, akkor nem tudja elérni az adatbázist.
- Adatok sérülése: A tartalom lesz hibás, ebből hibás üzleti döntések következhetnek. Alkalmazási programok hibái, DBMS szoftver hibái, rossz adatbázis-tervezés vagy felhasználói hiba. A hibás adatokat offline módba tesszük, és megpróbáljuk kijavítani a hibát.
- Adatbázis objektumok elvesztése: Ezek is általában emberi hibák, valamilyen helyreállítást kell végezni, ha az objektum elveszett, akkor a jogosultságokat is vissza kell állítani. Ha az index veszik el, akkor a teljesítmény romlik, de ezt helyre lehet állítani. Nézettábla esetén az alkalmazások nem tudják elérni, ha tudjuk mi volt benne, akkor nem nehéz helyreállítani, ha nem, akkor egy félnapos munka. :)
- Adatvesztés: A táblából néhány sor kitöröltek, vagy fölülírtak. Ez lehet alkalmazási program hiba, vagy emberi hiba. Ilyenkor is helyreállítást kell használni.
- Adatreplikálási és propagálási hibák: Különböző adatbázisok közötti szinkronizációról is lehet szó, meghibásodás esetén az adatok nem lesznek naprakészek. A publikáló oldalon nem lesz hiba, az előfizetői oldalon lesznek hibák. Ha nem megfelelő időpontban indul el, akkor teljesítményproblémákat okozhat.
- Súlyos teljesítménybeli problémák: Nem csak az adatbázis leállása esetén vannak elérhetőségi problémák. Sérült, vagy nem megfelelően definiált index, új felhasználók, adatnövekedés, nem megfelelő adatbázis statisztika, zárolási probléma okozhat ilyen problémát.
- Helyreállítási kérdések: Mentési/helyreállítási stratégiák vannak hatással az elérhetőségre. Mennyi idő alatt történik a helyreállítás, illetve milyen hatással van a mentés az adatbázis teljesítményére.
- DBA hibák: Hibás mentés, helyreállítás, verziófrissítés.
- Tervezett és nem tervezett leállások: A karbantartó feladatokhoz kell tervezett leállás. Tervezett leállás 70%, nem tervezett 30%, de a nem tervezettek fele a tervezett alatt keletkező problémák miatt történik.

Elérhetőség biztosítása:
- Rutinkarbantartás végzése, mialatt a rendszerek működnek: Ilyen termékekkel a karbantartási idő csökkenni fog. Vannak olyan segédprogramok, amelyek nem szakítják meg az adatbázis működését.
  - Adatbázis újraszervezése
  - Adatbázis mentése
  - Adatbázis helyreállítási megoldások, melyek a helyreállított adatokat használják leállás nélkül
  - Adatmentési és betöltési folyamatok (leginkább döntéstámogatási rendszerek és adattárházak számára szükséges)
  - Statisztika gyűjtő segédprogramok
  - Integritást ellenőrző segédprogramok

  Ezek a teljesítményre mindenképp hatással vannak, ha nem is áll le az adatbázis.
- DBA feladatok automatizálása: Az automatizált feladat kevésbé fog tévedni, mintha egy ember csinálja ugyanazt. Minél összetettebb, annál hasznosabb. Kevesebb idő szükséges hozzá. Biztonsági mentésekhez a legfontosabb.
- A magas elérhetőségű eszközök kihasználása: Klaszterezés vagy párhuzamos technológia. A modern hardver és OS eszközök kihasználása. A régi rendszereknél a rendszerparaméterek átállításához le kellett állítani az adatbázist.
- Klasztertechnológia hasznosítása: A terhelést sokkal könnyebben szét lehetett osztani. OS támogatás legtöbbször megvan.

<u>Teljesítménykezelés</u>
Monitorozni és hangolni kell a teljesítményt. Proaktívan kell hozzáállni. Legtöbbször ez nem csak a DBA feladata, hanem a programozóé illetve a többi rendszeradminisztrátoré.

A teljesítmény definíciója: A felhasználó információt kér az adatbázistól, a DBMS kielégíti ezt a kérést.
Egyszerű definíció: Azt a sebességet, amellyel a DBMS kielégíti az információkérést nevezzük adatbázis teljesítménynek.
Összetett definíció:

Öt tényező befolyásolja az adatbázis teljesítményét:

- Munkaterhelés:
  - Online tranzakció
  - Batch feladat
  - ad-hoc lekérdezés
  - Adattárház elemzés
  - Rendszerparancsok kombinációja, amely egy időben a rendszeren fut
  - Hónap végi zárás
- Áteresztőképesség: A számítógép teljes képessége az adat feldolgozásra.
  - I/O sebesség
  - Processzor sebesség
  - A gép párhuzamos képességei
  - OS és a rendszerszoftver hatékonysága
- Erőforrás: Rendelkezésre álló hardver- és szoftvereszközök.
- Optimalizáció: A rendszerek minden típusa optimalizálható. A relációs adatbázis az egyetlen, amelyben a lekérdezés optimalizáció valósul meg.
- Versengés: Ha az erőforrásra nagy az igény, akkor versengés léphet föl. A munkaterhelés két vagy több komponense egy egyedi erőforrást próbál használni, mégpedig konfliktusos módon. Pl.: Ugyanazt a sor akarom 2 helyről update-elni. Ahogy a versengés nő, az áteresztőképesség csökken.

Az adatbázis teljesítmény definiálható, mint az erőforrások optimalizálásának használata, hogy növeljük az áteresztőképességet, csökkentsük a versengést, lehetővé téve a lehető legnagyobb munkaterhelést. Hardver, OS és alkalmazások teljesítményéről is beszélnünk kell.

Előadás #11 09-12-01

2009. december 1.
7:57

Pareto-elv (80/20-as szabály): Az eredmény 80%-a az erőfeszítés 20%-ából származik.

Mi csökkentheti az adatbázis teljesítményét?
- Szegényes teljesítményű SQL (pl.: tábla átfuttatások, amikor a teljes táblát végig kell keresni)
- Megfelelő indexhiány, vagy az SQL nem a megfelelő indexet használja.
- Elavult adatbázis statisztika
- A táblák összekapcsolása nem optimális sorrendben
- Alkalmazás összekapcsolások a hatékonyabb SQL összekapcsolások helyett
- Nem megfelelő összekapcsolási metódusok
- Hatékony SQL beágyazva egy nem hatékony alkalmazáskódban
- Nem hatékony allekérdezési formulák
- Szükségtelen rendezés (GROUP BY, UNION)

SQL monitorokkal könnyen megtalálhatók ezek a hibák. Ebből meg tudja mondani, hogy mennyi erőforrást fogyasztanak az SQL-ek, illetve, hogy ki és milyen programból adta ki ezt.

Mit érdemes ellenőrizni?
- Példány és OS teljesítmény
  - Memória allokáció
  - Naplózási opciók
  - I/O hatékonyság
  - Munkaterhelést a szerveren, ahol a teljes alkalmazás illetve az adatbázis fut
  - Adatbázis séma definíciók

Monitorozás és kezelés
- Monitorozás: Megtalálom a problémát. Ez a környezet vizsgálatát jelenti, eszközök kimeneténekáttekintése, rendszer futásának általános …
- Elemzés: Összegyűjtjük az információkat.
- Javítás vagy optimalizáció: Bizonyos műveleteket lehet automatizálni. Pl.: egy esemény hatására valamilyen javító művelet el tud indulni.

Teljesítmény becslése a termelésbe kerülés előtt: A fejlesztés alatt lesz valamilyen teljesítmény, de a tesztszerveren nincs annyi adat és nem olyanok az erőforrások, mint a termelésinek, de a DBA-nak meg kell becsülnie a teljesítményt a termelési rendszeren. Az SQL-eket lehet egyedileg hangolni, de nem biztos hogy ez lesz az optimális.

Történeti trendek: Erőforrás használati trendek, és teljesítménystatisztikák. Ezekből a DBA akár hónapokkal előre meg tud jósolni bizonyos problémákat.

Szolgáltatás szintű kezelés (service-level management (SLM))
Szolgáltatás szintű egyezmény (servive-level agreement (SLA)): Megegyezés az adminisztrátorok és a felhasználók között.

Teljesítményhangolás típusai
- Rendszerhangolás
  - DBMS szoftver telepítésének a módja
  - A memória, CPU, lemez és más erőforrások konfigurációs opciói
  - OS, hálózati szoftver, tranzakció feldolgozó beállításai
- Adatbázis hangolás
- Alkalmazás hangolás: SQL hangolás

Ezeket kevés külön-külön hangolni, egymásra is hatással vannak.

<u>Teljesítményhangoló eszközök</u>
- Teljesítménykezelő kategóriában:
  - Teljesítmény monitorok:
    - Valós idejű
    - Történeti trendekre alapuló
    - Near time, valamilyen intervallumokra vonatkozhat
  - Hatékonyság becslő eszközök: Programokhoz, SQL utasításokhoz tudunk teljesítmény becslést adni. Elérési útra, működési környezetre, szabály-, vagy következtető motorra alapozva.
  - Kapacitástervező eszköz: Elemzi a jelenlegi környezetet és az adatbázis tervet, és ezek alapján a DBA tud kérdéseket feltenni
  - SQL elemző és hangoló eszköz: Grafikus vagy karakteres. A lekérdezés elérési útját adja meg (lekérdezési fa), lehet SQL utasítás és programok esetén is használni.
  - Advisory (tanácsadó) eszköz: Bővítjük az előzőt, valamilyen tudásalappal, ami tippeket ad, hogy hogyan optimalizálhatjuk az SQL utasításokat.
  - Rendszer elemző és hangoló eszköz: Az adatbázis és rendszer paramétereket lehet látni és módosítani vele.
- Teljesítmény optimalizáló kategóriában
  - Újraszervező eszköz
  - Tömörítő eszköz: Kevesebb I/O, viszont több CPU idő.
  - Rendező eszköz

Rendszerteljesítmény
- Hardver konfiguráció
  Kérdések:
  - Megfelelőek-e a DBMS környezet számára a hardver képességei?
  - A firmware-ek naprakészek-e?
  - Megfelelő mennyiségű memória van-e a gépben? (az összes telepített szoftverhez)

    - Van-e megfelelő mennyiségű lemezterület allokálva és konfigurálva a DBMS használatára?
    - Milyen típusú tárolási eszközt használunk?
    - A hálózati kábelek megfelelően működnek-e?
    - A fizikai kapcsolatok teljesen kapcsolódtak-e, és működnek-e?
    - Van-e UPS?
- Lemezes tároló és az I/O: Az I/O a legköltségesebb művelete a DBMS-nek. A lemezelérést érdemes lenne optimalizálni. Hatékony megoldás az SSD, a könyv előnyben részesíti a RAID-del és a SAN-nal szemben is.
- Kapcsolat az OS-sel
  Kérdések:
    - Elegendő mennyiségű memória lett-e allokálva az OS szintű feladatokhoz?
    - Van-e elegendő mennyiségű memória allokálva swap területhez?
    - Hogyan lettek az adatbázis állományok allokálva az adatbázis megvalósításakor? (raw partíció)
    - Megfelelő prioritást határoztunk-e meg az adatbázis feladatokhoz?
    - Az OS a kért verzión van-e telepítve?
    - Van-e javítás az OS-hez, és ezek fel vannak-e telepítve?
    - Az OS konfigurációs paraméterei módosultak-e amikor telepítettük a DBMS-t? És ha igen, akkor ez hogyan hatott más folyamatokra?
- Más szoftverek: Pl.: tranzakció feldolgozók, hálózati szoftverek, üzenet sorba állító szoftverek, web kapcsolódási és fejlesztő szoftverek, programozási nyelvek. Ezek is hatással lehetnek az adatbázis teljesítményére.
- A DBMS komponensei: Van egy csomó paraméterünk, amiket nagyon szépen lehet változtatni, ezek lehetnek valamilyen állományban, lehet hogy DBMS-beli utasítást kell hozzá kiadni, vagy valamilyen rendszereljárást kell futtatni. Ha nagy adatbázist akarunk létrehozni, akkor az alapértelmezett paraméterek nem lesznek jók.
- A konfigurációk típusai: Vannak dinamikusan és nem dinamikusan változtatható paraméterek.
- Memóriahasználat: A DBMS-ek imádják a memóriát, és minden féle gyorsítótárat (puffer) használnak, hogy az erőforrások minél tovább a memóriában maradjanak. Ezzel csökken az I/O és javul a teljesítmény.
    - Adatpuffer
    - Eljárásgyorsító: SQL és programokhoz kapcsolódó struktúrák vannak benne. Az adatmódosító és a lekérdező SQL utasítások előtt optimalizálni kell. Ez belső struktúrát hoz létre, amely egy elérési utat reprezentál, ami az adatok olvasására való. Az eljárásgyorsítóban ezeket az elérési utakat tudjuk tárolni. Ha még egyszer fut az SQL, akkor nem kell optimalizálni.
    - Rendezési gyorsító: Ideiglenes lemezterületek helyett használja a DBMS a közbenső rendezési eredmények memóriában való tárolására.
    - DBMS belső struktúra gyorsító: Ezt általában nem látjuk, de jó, ha van. Nem kell kezelnie a programozónak vagy DBA-nak, de jó ha tudjuk, hogy van, mert memóriában helyet foglal
    - Naplógyorsító: A naplórekordok memóriában tárolására szolgál. Két féle van, írási és olvasási.
        - Írási: Mindenféle DML műveletet naplózunk. Ezek először a memóriába kerülnek, aztán a lemezre.
        - Olvasási: A rollback és a recover műveletek használják ezt a naplógyorsítót.
- Memória további területei:
    - Felhasználói kapcsolatok: bárki/bármi ami kapcsolódik az adatbázishoz, ahhoz szükség van DBMS-beli memóriára a kapcsolat fenntartásához és kezeléséhez.
    - Különböző (adatbázis által használt) eszközöknek lehet szüksége rendszermemória fenntartására.
    - Nyitott adatbázisok: Minden egyes adatbázisnak memóriakövetelménye van, ezért meg lehet adni egy maximális számot (paraméter), hogy hány nyitott adatbázis tartozhat egy DBMS-hez.
    - Nyitott objektumok: Tábla, index, vagy bármilyen adatbázis objektum. Minden adatbázis objektum memóriát követel. Ehhez is van egy paraméter, ami megmondja, hogy maximum hány adatbázis objektum lehet egy időben nyitva.
    - Zárak: Ezekhez van egy zártábla, de ezekhez is kell memória. Ehhez is van egy paraméter, hogy egy időben egyszerre maximum hány konkurens zár tartható fenn.

<u>Az adatgyorsító részei:</u>
- Használatban lévő lapok: A DBMS jelenleg olvassa vagy módosítja. Más adatbázis folyamatok nem érhetik el ezeket a lapokat.
- Módosított lapok: Módosítva lettek, de nem kerültek a lemezre kiírásra. Más adatbázis folyamatok ezeket sem érhetik el.
- Elérhető lapok: Nem használt lapok, ezek érhetők el más folyamatok számára.

Ha túl sok a nem elérhető lap az adatgyorsítóban, akkor a DBMS-nek törekednie kell arra, hogy minél gyorsabban kiírja ezeket a lemezre, egyébként az adatbázis feldolgozásból eredő írási kéréseknek várniuk kell.

Adatgyorsító monitorozása és hangolása: Ha túl sok, akkor pocsékoljuk a memóriát, ha túl kicsi, akkor túl gyakran kell a lemezre írni. Van egy ún. olvasási hatékonysága, ami megmutatja, hogy milyen jól teljesíti a gyorsító az elsődleges feladatát, hogy elkerülje a fizikai I/O műveleteket.

Olvasási hatékonyság = (DB I/O kérések - fizikai I/O)/(DB I/O kérések)

Ha sok a szekvenciális feldolgozás, akkor a hatékonyság csökken. Ha ritkán kérik le újra az adatokat, akkor alig van fizikai I/O.
Ha ez 80% alatt van, akkor érdemes az adatgyorsítót növelni.

Eljárásgyorsító monitorozása és hangolása: A párhuzamosan futó SQL-ekhez kell hogy alkalmazkodjon, ennek is van olvasási hatékonysága, 60-80% között jó, de ez függ az alkalmazások típusától, illetve, hogy egyes programok vagy SQL-ek hányszor futnak.

Előadás #12 09-12-08

2009. december 8.
8:15

Nyílt adatbázis objektumok:

Ha adatbázis objektumot megnyitunk, akkor azt az a DBMS -nek kezelnie kell. A nyílt adatbázis objektumoknak memória igényük van. Ha lekorlátozom ezt, akkor nem fogja megenni a memóriát.

Adatbázis naplók:
A DML utasításokat szokták naplózni. A naplót használja a rollback és a recover. A naplózást ki lehet kapcsolni. A napló mére te folyamatosan nő, ezt monitorozni kell.
A változásokat először naplóba írjuk, csak utána kerülnek az adatbázisba. Ha valami történik az adatommal, akkor először itt fogom keresni, ha itt nincs bejegyzést, akkor az elveszett.
Ellenőrzési pontot hozhatunk létre, hogy minden rekord és adatlap biztonságosan a lemezre íródjon.

Be lehet állítani, hogy milyen gyakran történjen ellen őrzési pont. Ez lehet előredefiniált intervallum, vagy a kiírt naplórekordok előredefiniált száma.

Mi történik, ha újraindul az adatbázis?
Lesznek olyan naplórekordok, amiket vissza kell görgetni

Helyreállításkor kell a napló, hogy visszaállítsuk az adatbázist. El őször a mentéseket rakjuk vissza, aztán a tranzakciós naplókat előregörgetjük, hogy az aktuális képet megkapjuk.

Az adatbázis napló konfigurációja:
- I/O puffereket kell definiálni a naplóhoz: Először a memóriába kerül a napló. Nem kell várni az adatbázisnak az I/O-ra.
  - Input puffer: a rollback és a recover műveleteket támogatja.
- Aktuális napló állományokat definiálni
  - Dual naplózás: redundáns naplóhasználatot eredményez. Két különböző állományba írom ugyanazokat az adatokat. Ha az egyikben történik valami hiba, akkor a másik megvan. Javasolt a két naplót külön lemezre helyezni.
- Napló kimentést definiálni
  - Archiválás: Az aktív naplót archív naplóba menti. Az aktív naplót csak úgy tudom kimenteni, ha nem használja senki, így át kell kapcsolnom a naplózást egy új aktív naplóba.

Ha mentést hajtunk végre, akkor a naplót is menteni kell vele.

Minden adatbázis művelet naplózva van?
- Ki lehet kapcsolni a naplózást, nagymennyiségű adatváltozásnál érdemes is, de utána teljes mentés kell
- Create indexet nem biztos, hogy naplózza, mert az indexet újra lehet generálni.

Zárolás és versengés: Van egy objektumom, és ezzel többen szeretnének olvasni.
Mi lesz hatással a teljesítményre?
- Zárolás felfüggesztés: az egyik folyamat zárat kért, olyat, amivel a másikat nem engedi hozzáférni, emiatt a másik folyamat felfüggesztésre kerül. A zárolás miatt nem tud tovább dolgozni.
- Timeout: Az alkalmazás megáll, mert tovább volt felfüggesztve, mint egy előre megadott időintervallum. Ez paraméterezhető.
- Holtpont: Az egyik folyamat kért egy objektumra egy zárat, egy másik folyamat egy másik objektumra kért egy zárat, és utána az egymás által zárolt objektumokat akarják használni. Paraméterben be lehet állítani a holtpont keresési kört.

Rendszerkatalógus: Egy elszeparált helyre illik elhelyezni.
Más konfigurációs opciók:
- Hány beágyazott triggert lehessen hívni. Az egymásba ágyazott hívások maximális számát lehet korlátozni. Ha eléri a maximális számot, akkor az egészet vissza kell görgetni.
- Biztonsági opciók: jogosultságokhoz, biztonsági funkcionalitáshoz.
- Elosztott adatbázisokhoz lehet megadni különböző paramétereket.

Rendszermonitorozás: A DBA feladata, hogy figyelje, hogy mi történik a rendszerben, nem csak az adatbázist kell monitorozni, hanem az OS -t, a hálózatot, adatbázisszervert, és minden kapcsolódó dolgot.

Adatbázis teljesítmény:
- Particionálás: Van egy csomó adatunk, az adatbázisban tábláink vannak, amiket állományokba kell elhelyezni, ezt úgy is megtehetem, hogy táblánként kerül egy állományba, vagy van több kis táblám, és az kerül egy állományba, vagy van egy nagy táblám, és az nem biztos, hogy belefér egy állományba, így többe szeretném elhelyezni. Ehhez használjuk a particionálást. (Igazából nem a táblákhoz, hanem a táblaterekhez rendelünk állományokat) Ha különböző lemezeszközökön vannak a particionált állományok, az a párhuzamos feldolgozást segíti.
- RAW partíció vagy állományrendszer:
  - RAW partíció: Nem az OS kezeli az állományokat, hanem a DBMS. Hatékonyabb lehet, mert az OS-beli puffer nem okoz késleltetést, illetve mert az OS-beli fejlécek nem kerülnek rá az adatokra.
- Indexelés: Az egyik legnagyobb eszköz a teljesítményhangolásra. A DBA-nak meg kell állapítani, hogy az új indexeknek milyen hatása lesz a teljesítményre. Ha sokat módosítok egy táblán, akkor hátrány is lehet, mivel az indexet is módosítani kell. Ha indexet teszek egy táblára, akkor meg kell vizsgálni a proci teljesítményt, az I/O-t, stb… Ha a táblánk túl kicsit, akkor felesleges rátenni az indexet (10 lap). Ha PK-t vagy unique megszorítást teszek a táblára, az automatikus indexgenerálást eredményez. Ha a teljes táblát keressük, akkor felesleges az index. Probléma lehet az is, ha változó hosszúságú oszlopokat indexelünk.
  - A sorokat oszlopérték(ek) szerint szeretnénk elhelyezni.
  - Kapcsolatok hatékonyságának növelése. (FK megszorítás)
  - Táblák közötti adatok összehasonlítása.
  - Adatok csoportosítása.
  - Adatok rendezése.

  SELECT empNo, lastName, salery FROM epmloyee
  WHERE salery > 15000;
  Ha erre olyan indexet hozunk létre, ahol a salery oszlop szerepel, akkor növeli a teljesítményt. Ha benne van az empNo és a lastName oszlop, akkor az indexet túlterheltük, és a teljesítményt tovább növeltük, mivel nem kell elmenni a tábláig, csak egyszerűen felolvasom a indexből. A túlterhelt indexekkel vigyázni kell, mert nem minden lekérdezésnél jó.
- Denormalizálás: A normalizáció ellentéte. Ha redundáns adataink vannak, akkor az adatkinyerés felgyorsulhat. Miket érdemes figyelembe venni:
  - Előre összekapcsolt táblákat készíthetünk: ezeket sokat fogjuk használni, és így gyorsabb használni
  - Riport tábla: Bizonyos riportokat sűrűn használunk

    - Tükör tábla: A táblára két típusú környezetnek párhuzamosan van szüksége.
    - Osztott táblák: Két különböző csoport egy tábla különböző részeit használja.
    - Kombinált táblák: Az 1-1 vagy 1-n kapcsolatot egyesítjük egy táblába.
    - Gyorsító táblák: A hierarchiák támogatására.
    - Fizikai denormalizáció
    - DBMS jellemzők kihasználása
    - Redundáns adatok tárolása, hogy csökkentsük a szükséges összekapcsolások számát.
    - Az ismétlődő csoportok egy sorba tárolását, hogy csökkentsük az I/O-t és a szükséges lemezterületet.
    - Származtatott adatok tárolása, a számítások, és a költséges algoritmusok elkerülésére.
- Klaszterezés: A sorokat fizikailag a lemezen egy megadott oszlop szerinti sorrendben fogja tárolni. Ehhez általában van egy klaszterező index és ez kényszeríti a táblába a sorokat, hogy az oszlop szerint sorrendbe legyenek tárolva. Meg lehet adni, hogy az indexben több oszlop is lehet, de egy táblához csak egy klaszterezési sor lehet. Ha új sort szúrunk be egy klaszterbe, akkor vagy oda kerül, ahova való, de ha nem talál elég helyet, akkor máshova szúrja be, és mutatókkal fogja megtalálni. Ha sokat módosítunk rajta, akkor újraszervezés szükséges. Viszonylag nagy tábláknál érdemes használni. Elsődleges kulcsot nem klaszterezünk mert definíció szerint egyedi.
  Laposztás: ha beszúrunk új adatot, akkor előfordulhat, hogy nincs hely. Ha nem talál olyan lapot, amelybe beférne, akkor új lapot kell létrehozni, ezt hívják laposztásnak.
    - Normál laposztás: Létrehoz egy új, üres lapot a megtelt lap és a következő lap között. A megtelt lap bejegyzéseinek felét átmozgatja az üres lapra, és ezután rakja helyre a mutatókat és szúrja be az új sort. Ha növekvő sorrendű sorokat szúrunk be, akkor sok hely mehet kárba, mert félig telt oldalakat hoz létre a DBMS.
    - Monoton laposztás: Létrehoz egy új, üres lapot a megtelt lap és a következő lap között. Az új értékeket az új lapra szúrja be. Ez akkor hasznos, ha a sorok szigorúan növekvő sorrendbe lesznek beszúrva. Ha a lapvégére kell beszúrni az új sort, akkor a monoton laposztást érdemes meghívni.

  Van olyan adatbázis, amelyik mindkettőt támogatja, illetve van olyan, amelyik csak az egyiket.
- Illesztett (interleaving) adatok: Ha két tábla adatait gyakran összekapcsoljuk, akkor fizikailag érdemes egy helyre rakni őket. (Mondjuk egy táblatérbe)
- Szabad hely: Új adatokat legyen hova beszúrni.
    - Csökkentheti az újraszervezés gyakoriságát.
    - Csökkenti a versengést.
    - Növeli a beszúrások hatékonyságát.
    - Ha új sort szúrunk be, akkor azt megfelelően lehet klaszterezni.
    - A változó hosszúságú soroknak és módosított soroknak van helyük nőni.
    - Ha egy lapon kevesebb sor van, az jobb konkurenciát eredményez, mert ha a lapot zároljuk, akkor kevesebb adatot zárolunk, így kevesebb adat nem lesz elérhető a többi felhasználó számára.

  Create utasításba szokott lenni egy "pct free" nevű paraméter, amely meghatározza, hogy az egyes adatlapokon hány %-nak kell elérhetőnek lennie a jövőbeli beszúrásokhoz. Másik paraméter a "free page", ez a lapok egy bizonyos számát adja meg, amely után egy teljesen üres lap érhető el.
  Hátrányok:
    - Lemeztárolási követelmények nagyobbak
    - A keresés tovább tart
    - Ha a lapon kevesebb sor van, akkor több I/O művelet szükséges, hogy megkapjuk a kért információt.

  Mire alapozzuk, hogy mennyi szabad helyet hagyunk?
    - Beszúrások és módosítások gyakorisága
    - Szekvenciális és véletlen elérések mennyisége
    - Nem klaszterezett adatok hatása
    - Feldolgozás típusa
    - Sorláncolás, sormigráció és laptörések valószínűsége
- Tömörítés: Már volt 100x... I/O, lemezterület csökken, CPU használat nő.
- Állomány elhelyezés és allokáció:
    - Indexeket és az adatokat külön rakjuk.
    - Ha van két vagy több táblám, amiket egyszerre érem el, akkor érdemes külön tárolóra rakni.
    - Adatbázis naplót is külön tároljuk az aktuális adatoktól.
    - Elosztott adatelhelyezés: Az adatnak azon a szerveren kell lennie, ahol a legvalószínűbb, hogy keresik, illetve ahol a leggyakrabban érik el. A cél a hálózati adatátvitel csökkentése.
    - Lemezallokáció: A lemezhez valamilyen logikai nevet rendelünk, utána a DBMS rendszerkatalógusába kerül, és onnan már el tudom érni.
- Lapméret: A lap vagy a blokk (lap by Ora) méretét lehet meghatározni DBMS szinten. Táblák sorait tároljuk lapokon. Hogy mekkora lesz a lap méret, azt a DBA-nak illik kiszámolnia. Ezt csak telepítésnél lehet beállítani.
- Újraszervezés: Ha a fizikai allokáció szétszórttá válik, akkor újraszervezésre van szükség. Lehet csak táblateret vagy indexet újraszervezni.