Vágner Anikó – Juhász István:

Adatbázis-adminisztráció

Debrecen, 2011

Lektorálta: Veréb Krisztián

Debrecen, 2011. március 30.

A tananyag a TÁMOP-4.1.2-08/1/A-2009-0046 számú Kelet-magyarországi Informatika Tananyag Tárház projekt keretében készült. A tananyagfejlesztés az Európai Unió támogatásával és az Európai Szociális Alap társfinanszírozásával valósult meg.

Előszó

Jelen jegyzet egy informatikai alapképzés egy elméleti kurzusának az anyagát tartalmazza. Áttekintést ad az adatbázis-adminisztrátori feladatokról, eszközökről. A jegyzet elsősorban relációs adatbázis-kezelő rendszerekkel foglalkozik, azonban néhány ismeretet más adatbázis-kezelő rendszerek esetén is jól lehet használni. A jegyzet haladó ismereteket tárgyal, ezek elsajátításához az alábbi előismeretek szükségesek:

- absztrakt adatszerkezetek
- állományszervezés
- operációs rendszer alapismeretek
- adatmodellezés
- a relációs adatmodell
- relációs adatbázis-kezelő rendszer alapismeretek
- a szabvány SQL nyelv ismerete

A jegyzet 15 fejezetből áll. A fejezetek között igen erős az áthivatkozás. A fejezetek sorrendje egyfajta logikát követ, de ettől eltérő sorrendben is feldolgozhatók. Minden fejezet végén kérdések segítik az elsajátított tudás ellenőrzését.

A jegyzet átfogó, általános ismereteket közvetít, amelyek a relációs adatbázis-kezelő rendszerekben közösen megvannak. Azonban a gyakorlatban az egyes konkrét rendszerek specialitásait, megközelítési módját, eszközeit kell ismerni, és valaki csak hosszú évek alatt megszerzett gyakorlat alapján válhat jó adatbázis-adminisztrátorrá.

1. fejezet – Az adatbázis-adminisztrátor

A fejezetben tisztázzuk, hogy ki az az adatbázis-adminisztrátor, és milyen feladatai vannak. A feladatok egy részét itt csak röviden említjük meg, mert a teljes jegyzet az adatbázis-adminisztrátor feladatait részletezi. Más feladatokra pedig csak utalunk, mert ismertnek tételezzük fel őket. Ezeken kívül még néhány alapfogalommal ismerkedünk meg, vagy csak felelevenítjük azokat.

Adatbázis – adatbázis-kezelő rendszer – adatbázis-adminisztrátor

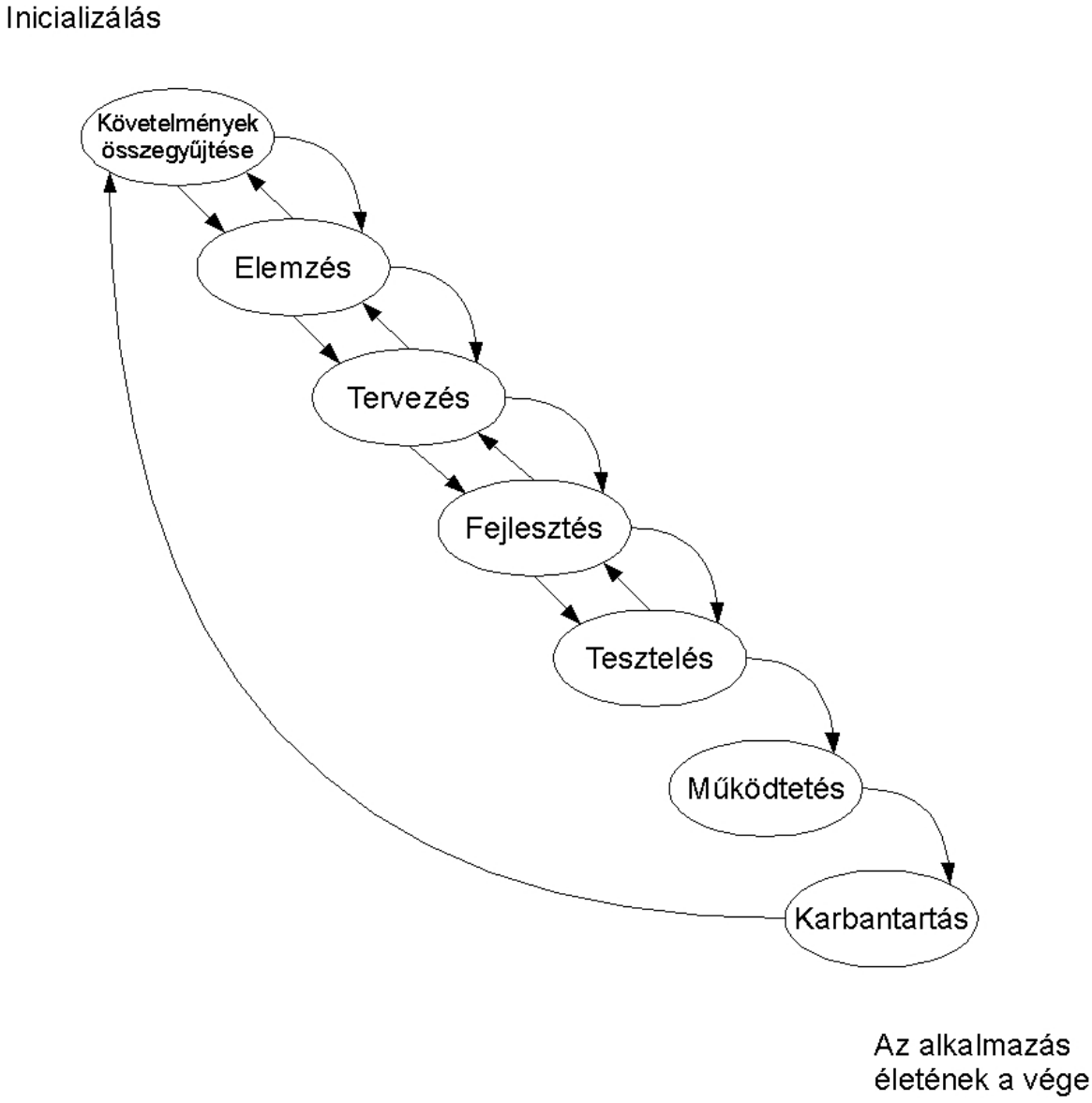
Ahhoz, hogy az adatbázis-adminisztrációról beszélni tudjunk, tisztában kell lennünk azzal, hogy egyáltalán mi is az az adatbázis, amivel az adminisztrátor dolgozik. Az *adatbázis* egy szervezett adattár, amelyben az adatok adatelemekként (például mezőkként, rekordokként, és állományokként) érhetőek el.
Az adatokkal természetesen dolgozni is szeretnénk, amihez szükség van egy szoftverre. Az *adatbázis-kezelő rendszer* (Database Management System, DBMS) egy szoftver, amely a végfelhasználóknak vagy az alkalmazásprogramozóknak lehetővé teszi, hogy megosszák és kezeljék az adatokat. Segítségével az adatbázisban adatokat tudunk létrehozni, módosítani, visszanyerni és tárolni, továbbá az adatok tulajdonságait és a köztük lévő kapcsolatokat leírni. Egy adatbázis-kezelő rendszer általában felelős az adatintegritásért, az adatbiztonságért, az adatelérés vezérléséért és optimalizálásáért, az automatikus visszagörgetésért, az újraindításért és a visszaállításért.
Az egyes cégek alkalmaznak egy vagy több olyan képzett szakembert, aki csak a cég adatbázisaival és a kezelő szoftvereivel foglalkozik. Az *adatbázis-adminisztrátor* (Database Administrator, DBA) felelős egy cég adatbázisainak tervezéséért és karbantartásáért, azaz a cég adatbázisainak folyamatban levő működésének biztosításáért és hatékonyságáért, és azért, hogy az alkalmazások elérjék az adatbázisokat.

Az adatbázis-adminisztrátor szemlélete

Az adatbázis-adminisztrációs feladatokhoz kétféleképp lehet hozzáállni: proaktív és reaktív módon.
A reaktív adatbázis-adminisztrátor leginkább tűzoltás jelleggel végzi el a feladatait. Az adatbázis-adminisztrátor a probléma bekövetkezte után próbálja orvosolni azokat. A reaktív adatbázis-adminisztrátor az előtte álló problémák közül mindig a legsúlyosabbnak a megoldására fókuszál.
Ezzel ellentétben a proaktív adatbázis-adminisztrátor a lehetséges problémák megelőzésén dolgozik. Tehát még a problémák bekövetkezte előtt látja, hogy mi okozhat majd problémát, és olyan megelőző lépéseket tesz, amelyek segítségével elkerüli a problémát.
Egy adatbázis-adminisztrátor életében nem lehet minden lehetséges problémát proaktív módon kezelni. Még egy tapasztalt adatbázis-adminisztrátor sem tud minden lehetséges problémára felkészülni, lesznek nála is tűzoltás jellegű feladatok. Főleg azok a feladatok, amelyek a programozói és a felhasználói kérések, problémák megoldására szolgálnak. Ide tartozik a futási idejű kivételek kezelése és a váratlan leállás utáni újraindítás is. Előfordulhat, hogy ezek a kérések elárasztják az adatbázis-adminisztrációs csoportot, és a csoport nem győzi azokat megoldani. Ennek viszont a proaktív feladatok látják kárát. Sok oka lehet annak, hogy az adatbázis-adminisztrációs csoport nem tudja megfelelően ellátni a feladatait: munkaerőhiány, túlvállalás az új alkalmazásfejlesztési projektek támogatásában, nem rutin feladatok megjelenése, vagy költségvetési hiány.

Az adatbázis-adminisztrátor szerepe az alkalmazásfejlesztésben

Ahhoz, hogy az adatbázis-adminisztrátor a problémákat proaktív módon kezelni tudja, a cégen belüli adatbázisok kezelésére és fejlesztésére stratégiai tervet fejleszt, amit természetesen meg is valósít. Ehhez tudnunk kell, hogy a cégen belül az adatok kezelése egy fejlődési folyamaton megy keresztül. A tervnek figyelembe kell vennie az alkalmazásfejlesztés életciklusának minden fázisát. Tekintsük át ezeket a fázisokat (1-1. ábra) az adatbázis-adminisztrátor szemszögéből nézve!



1-1. ábra Az alkalmazásfejlesztés életciklusa

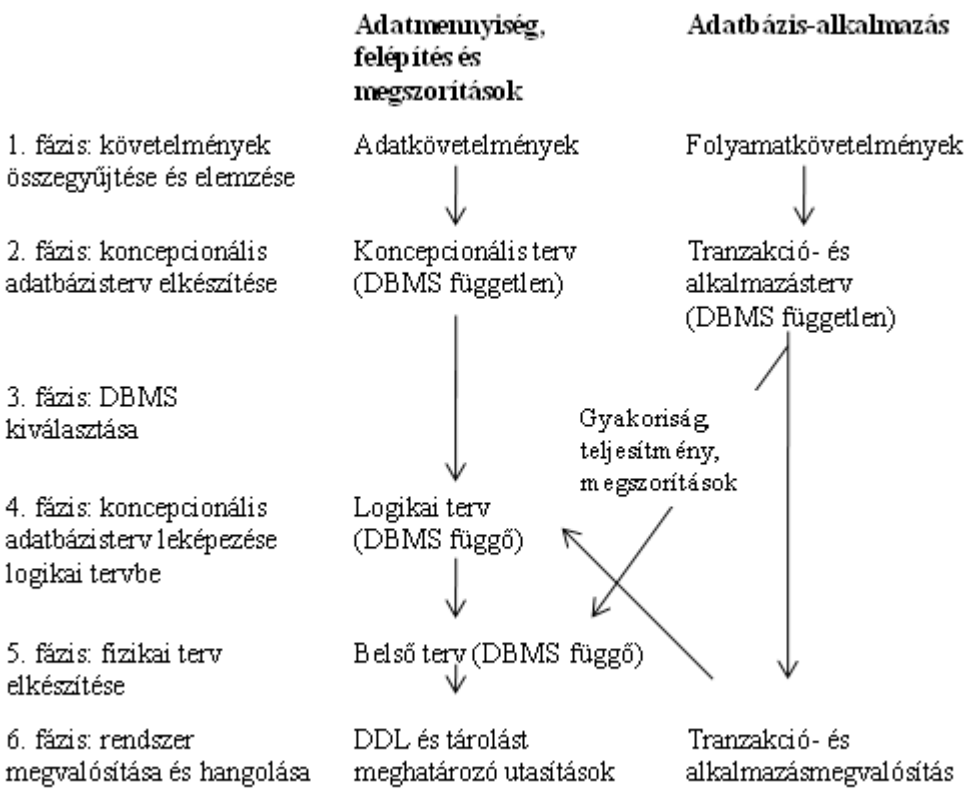
Egy adatspecialistának, általában az adatbázis-adminisztrátornak, jelen kell lennie a teljes cikluson keresztül. Az inicializálási fázisban és a követelmények összegyűjtésének a fázisában az adatbázis-adminisztrátor azonosítja a projekt adatelemeit, segít meghatározni, hogy az új fejlesztéshez szükséges adatok újak-e vagy már léteznek valahol a cégen belül, és dokumentálja a felhasználók kéréseit kielégítő adatkövetelményeket.

Az elemzési fázis alatt az alapvető adatkövetelmények alapján az adatbázis-adminisztrátor egy koncepcionális adatmodellt készít. A koncepcionális modell a felhasználói adatkövetelmények részletezése mellett már részletes leírást ad az egyedtípusokról, a kapcsolatokról, a megszorításokról. A koncepcionális modell nem tartalmaz információkat a megvalósításról, ezért általában könnyen érthető, és fel lehet használni a felhasználókkal való kommunikációban. Segítségével az adatbázis-adminisztrátor ellenőrizheti, hogy minden felhasználói adatkövetelményt figyelembe vett-e, illetve felfedezheti az esetlegesen konfliktusban álló adatkövetelményeket. A modell azt is lehetővé teszi, hogy az elemzési fázis alatt az adatbázis-adminisztrátor csak az adatok tulajdonságainak a meghatározására figyeljen, miközben a megvalósítási és a tárolási részletekkel egyáltalán nem kell foglalkoznia, sőt még azt sem kell tudnia, hogy milyen adatbázis-kezelő rendszert fog használni.

A tervezési fázis alatt az adatbázis-adminisztrátor a koncepcionális adatmodellt egy logikai adatmodellbe transzformálja. A logikai adatmodell elkészítéséhez már konkrétan tudni kell, hogy milyen típusú adatbázis-kezelő rendszert fog a cég használni az alkalmazásához, lehet ez például relációs vagy objektum-relációs. Ne feledjük el, hogy ez a jegyzet elsősorban relációs adatbázisokkal foglalkozik. A logikai modell létrehozása abból áll, hogy a koncepcionális adatmodellt az adatbázis-adminisztrátor a cég adatbázis-kezelő rendszere által használt modellbe transzformálja. Ez a leképezés egy adatbázis-tervező eszköz segítségével automatizálható.

A következő lépésben a fejlesztés megkezdése előtt a logikai adatmodellből fizikai adatbázistervet készít az adatbázis-adminisztrátor. Ehhez már konkrétan tudni kell, hogy melyik adatbázis-kezelő rendszert fogja a cég használni. Ez lehet pl. Oracle vagy DB2 vagy valamelyik nyílt forráskódú rendszer. A fizikai adatbázisterv meghatározza a tárolási struktúrákat, az állományokat, az indexeket, az elérési utakat és az adatbázis-állományokhoz kapcsolódó paramétereket és a szállításspecifikus információkat.

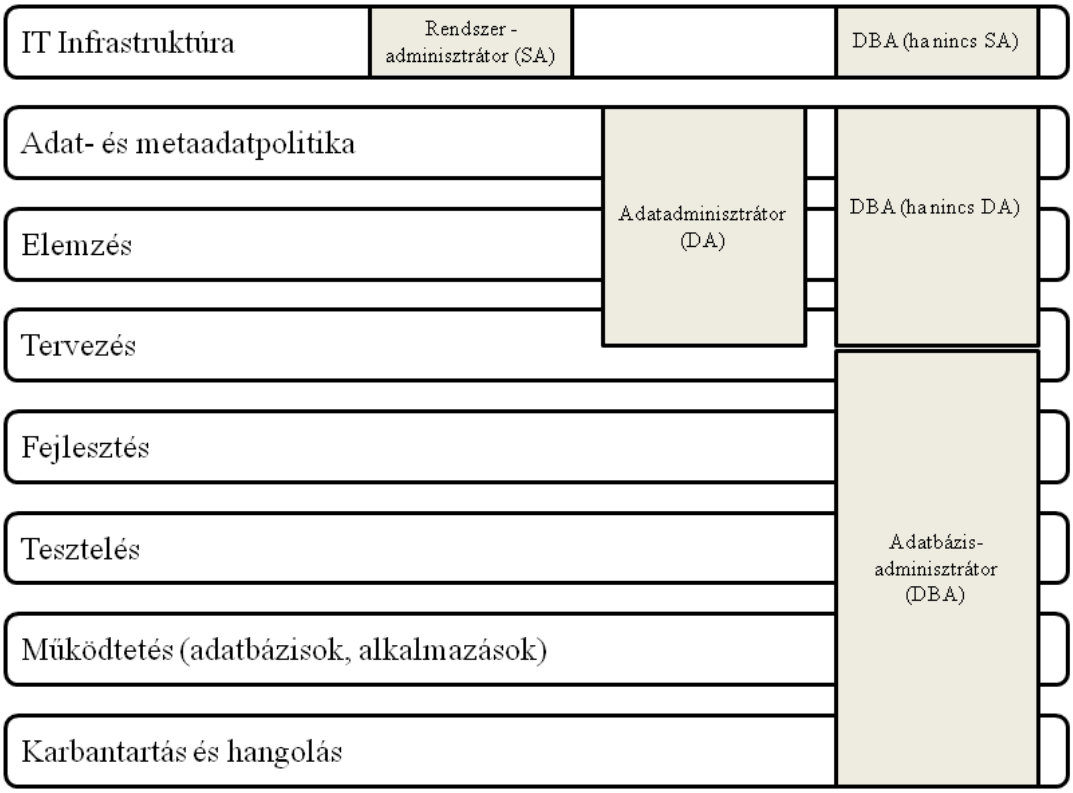
Az elemzési, tervezési és fejlesztési fázisokat az adatbázis-tervezés szemszögéből nyomon követhetjük az 1-2-es ábrán.



1-2. ábra Adatbázis-tervezés fázisai

Az adatbázis-adminisztrátornak az alkalmazás teszteléséhez példaadatokat kell a fizikai adatbázisba felvennie. A programozók és a saját munkájának a segítésére valamilyen automatikus megoldást kell találnia, hogy a tesztadatokat egyszerűen lehessen az adatbázisba újra és újra betölteni.
Ha az alkalmazás fejlesztése a működési státuszba kerül, az adatbázis-adminisztrátor előkészíti az adatbázis-kezelő rendszert a munkaterhelésre. Ez az előkészítés magában foglalja a megfelelő biztonsági üzenetek megvalósítását, az új alkalmazás tár- és memóriakövetelményeinek felmérését és módosítását, és annak előrejelzését, hogy az új munkaterhelés milyen hatással lesz a már meglévő adatbázisokra és alkalmazásokra. Az adatbázis-adminisztrátor felelős az adatbázisnak a teszt környezetből a termelési környezetbe való migrálásáért.
Amíg az alkalmazás működik, az adatbázis-adminisztrátor rengeteg feladatot teljesít: biztosítja az elérhetőséget, figyeli a teljesítményt, hangol, biztonsági mentéseket és helyreállításokat végez, és jogosultságokat ad.
Egyetlen alkalmazás és adatbázis sem marad statikus sokáig. Mivel az üzlet folyamatosan változik, az üzletet támogató IT rendszereknek is változniuk kell. Ha karbantartásra kerül sor, az felfogható egy új fejlesztésként, így az adatbázis-adminisztrátornak ebben az esetben is a teljes folyamatban részt kell vennie, a követelmények összegyűjtésétől a megvalósításig.
Végül, ha az alkalmazás eléri életének a végét, az adatbázis-adminisztrátornak kell segítenie meghatározni az alkalmazás azon adatait, amelyeknek a cég érdekei miatt túl kell élnie az alkalmazást.
Az adatbázis-adminisztrátor felelős a teljes adatbázis-környezet kezeléséért. Gyakran ez magában foglalja az adatbázis-kezelő rendszer telepítését és az IT infrastruktúra meghatározását, mindezt úgy, hogy az alkalmazások elérhessék az adatbázist. Ezeket a feladatokat be kell fejezni, mielőtt az alkalmazói programok elkészülnének.
Sok cégnél az ad hoc adatbázis-elérés alapvető követelmény. Az adatbázis-adminisztrátor feladata

az ad hoc lekérdező környezet kialakítása, amely magában foglalja a lekérdező és riportoló eszközöknek a kiválasztását vagy megvalósítását, a vezérelvek kialakítását a hatékony ad hoc lekérdezések biztosításához, és az ad hoc SQL utasítások hangolását és monitorozását.
Az 1-3. ábrán láthatjuk, hogy a különböző adatbázishoz kapcsolódó feladatokért melyik szerepkörhöz rendelt adminisztrátor a felelős.



1-3. ábra Adminisztrátori feladatok

Adatbázis-, adat- és rendszer-adminisztráció

Néhány cég különböző szerepköröket definiál az adatok üzleti és technikai szempontjai alapján. Az adatok üzleti szempontjai tartoznak az adatadminisztrátorhoz, a technikai oldalt az adatbázis-adminisztrátor kezeli. Nem minden cégnek van külön adatadminisztrációs feladatköre. Sok cég az adatadminisztrációt az adatbázis-adminisztrációs szerepkörhöz rendeli.
A cégek sokszor az adatkezelés technikai oldalát is felosztják: az adatbázis-adminisztrátor felelős az adatbázis-kezelő rendszer használatáért és egy rendszer-adminisztrátor vagy rendszerprogramozó felelős az adatbázis-kezelő rendszer telepítéséért és frissítéséért.

Adatadminisztráció

Az adatadminisztráció az adat, mint erőforrás kezelésének üzleti oldalát elválasztja az adatokat kezelő technológiától. Így sokkal közelebb kerül az adat az őt használó üzleti felhasználóhoz.
Az alkalmazásfejlesztés életciklusát tekintve az adatadminisztráció inkább az adatok gyűjtését, elemzését, és a tervezést foglalja magában, míg az adatbázis-adminisztrátor a tervezésben, a

fejlesztésben, és a működési fázisban dolgozik.
Az adatadminisztrátor (Data Administrator, DA) felelős az üzleti koncepciók megértéséért, ő azonosítja és katalogizálja az üzleti felhasználók számára szükséges adatokat. Ezek alapján hozza létre a koncepcionális és a logikai adatmodellt, amelyben az üzleti folyamatokhoz tartozó adatelemek közötti kapcsolatokat is pontosan leírja.
Az adatadminisztrátori csoportnak át kell látnia a cégnél lévő minden üzleti folyamat összes adatát. A jó adatadminisztrációs csoport munkájának eredményeként meghatározza a cég adatpolitikáját, azonosítja az adattulajdonosokat és gondnokokat. A cég adatainak a felügyeletéhez és használatához szabványokat alakít ki, megtervezi és érvényesíti a használathoz szükséges követelményeket, illetve segít szabványosítani az adatkezelési folyamatokat. Az adatadminisztrátori szerepkör magában foglalja az adatok dokumentációját, megosztását és létrehozását céges szinten.
Az adatadminisztrátor felelőssége biztosítani, hogy az adatok megfelelően legyenek dokumentálva. Ez a dokumentáció általában egy repository-ban van tárolva. Egy fontos különbség az adatadminisztrátor és az adatbázis-adminisztrátor között, hogy az adatadminisztrátor a repository tartalmára figyel, míg az adatbázis-adminisztrátor a fizikai adatbázisra és az adatbázis-kezelő rendszerre.
A metaadatok ugyancsak a repository-ban vannak tárolva. A metaadatok biztosítják a környezetet, amelyben az adatokat megérthetjük és számunkra információvá válnak. Az adatadminisztrátor felelős a cég metaadat-stratégiájáért. A metaadat-stratégia abban segíti a céget, hogy az információvagyont a felügyelete alatt tartsa, megértse és felmérje az értéküket. A Metaadatok című fejezetben további részleteket találhatunk.
A következő különbség az adatadminisztrátor és az adatbázis-adminisztrátor között, hogy az adatadminisztrátor a metaadatokkal, míg az adatbázis-adminisztrátor az adatokkal foglalkozik.
Az adatadminisztráció egyik legnagyobb feladata az adatmodellek létrehozása. A koncepcionális adatmodell az adatkövetelményeket nagyon magas szinten vázolja fel, míg a logikai adatmodell az adattípusokat, hosszakat, kapcsolatokat és számosságokat mély részletekben írja le. Relációs és objektumrelációs adatbázisok esetén az adatadminisztrátor normalizációs technikákat használ, hogy megbízható adatmodelleket szállítson, amelyek pontosan leírják a cég adatkövetelményeit.
Sok adatbázis-adminisztrátor nem szereti az adatadminisztrátorokat, mint adatmodellezőket, akikre csak azért van szükség, hogy a végfelhasználókkal megbeszéljék az adatbázis követelményeit. Egy igazi adatadminisztrátor feladata több, mint egyszerű adatmodellezés. Az adatadminisztráció egy üzletorientált szakterület, amely a cég adatvagyonáért felelős.
Általában az adatbázis-adminisztrátorok nem képesek az adatadminisztrátor feladatait felelősségteljesen ellátni. Az okok a következőek:

- Az adatbázis-adminisztrátornak sok más technikai kötelessége van, melyek az idejének a nagy részét kitöltik.
- Az adatbázis-adminisztrátori csoport vezetőjének általában nincs akkora végrehajtó hatalma, amely megengedné, hogy politikát diktáljon.
- Az adatbázis-adminisztrátor általában nem rendelkezik olyan képességekkel, hogy hatékonyan tudjon kommunikálni az üzleti felhasználókkal, és lehet, hogy nem is ért az adott üzleti folyamathoz.
- A legtöbb adatbázis-adminisztrátor boldogabb, ha technikai problémákkal kell foglalkozni, üzleti vagy nem technikai problémák helyett.

Ha az adatadminisztrátori és adatbázis-adminisztrátori feladatok egyidejűleg léteznek egy cégen belül, akkor a két csoportnak nagyon szorosan együtt kell dolgoznia. Fontos, hogy ugyanaz legyen a főnökük, ez megkönnyíti az együttműködést. Törvényszerű, hogy van néhány olyan ismeret, amelyre mindkét csoportnak szüksége van. De az adatbázis-adminisztrátor általában nem fogja megérteni az üzleti kérdéseket úgy, mint az adatadminisztrátor, és az adatadminisztrátor soha nem fogja úgy megérteni a fizikai adatbázist, mint az adatbázis-adminisztrátor. Ennek ellenére igaz, hogy az adatadminisztrátor és az adatbázis-adminisztrátor is hatékonyabban dolgozik, ha van

bizonyos ismerete a másik szakterületéről.

Adatbázis-adminisztráció

Röviden áttekintjük, hogy mi a feladata az adatbázis-adminisztrátornak, ha van adatadminisztrátor. Az első kötelessége, hogy megértse az adatmodellt, amelyet az adatadminisztrátor épített és megbeszélje a modellt az alkalmazásfejlesztőkkel és más technikai szakemberekkel. Az adatbázis-adminisztrátor a logikai adatmodellt használja a fizikai adatbázisok felépítéséhez. Az adatbázis-adminisztrátor transzformálja a logikai adatmodellt egy hatékony fizikai tervbe. Alapvető, hogy az adatbázis-adminisztrátornak a hatékony és megfelelő fizikai adatbázisterv elkészítéséhez az adatbázis-kezelő rendszerről szóló ismereteit használnia kell. Az adatbázis-adminisztrátor nem számíthat az adatadminisztrátorra a végső fizikai modellnél, mint ahogy az adatadminisztrátor sem számíthat az adatbázis-adminisztrátorra a koncepcionális és a logikai modellnél.
Az adatbázis-adminisztrátor a kommunikációs csatorna az adatadminisztrátori csoport és a technikai szakemberek, illetve az alkalmazásprogramozói csoport között. Természetesen az adatbázis-adminisztrátor munkájának zöme a fizikai modellből készült adatbázis és az adatbázis-alkalmazások kezelésének folyamatos támogatása.

Rendszer-adminisztráció

Néhány cégnek, általában a nagyobbaknak, van rendszer-adminisztrátor (System Administrator, SA) vagy rendszerprogramozó szerepköre. A rendszer-adminisztrátor befolyással van az adatbázis-kezelő rendszer megvalósítására. A rendszer-adminisztrátor felelős az adatbázis-kezelő rendszer telepítéséért. A rendszer-adminisztrátornak általában nincs felelőssége az adatbázistervért és támogatásáért. Az adatbázis-adminisztrátor az adatbázisért felelős, míg a rendszer-adminisztrátor az adatbázis-kezelő rendszer telepítését, módosítását és támogatását biztosítja. Továbbá a rendszer-adminisztrátor felel azért, hogy az IT infrastruktúra úgy legyen megvalósítva, hogy az adatbázis-kezelő rendszer együtt tudjon dolgozni más rendszerszoftverekkel. A rendszer-adminisztrátor más technikai szakemberekkel dolgozhat együtt, hogy a tranzakciófeldolgozó eszközöket, az üzenetsorba-állító eszközöket, a hálózati protokollt, és az operációs rendszer paramétereit konfigurálja az adatbázis-kezelő rendszer hatékonyan működéséhez. A rendszer-adminisztrátor biztosítja, hogy az IT infrastruktúra üzemeljen az adatbázis-fejlesztéshez. Ezt úgy teszi meg, hogy az adatbázis-kezelő rendszert megfelelően beállítja, az adatbázis-kezelő rendszer szállítójától kapott folyamatos karbantartásokat és frissítéseket alkalmazza és koordinálja a migrációt az új adatbázis-kezelő rendszer kiadásokra és verziókra.
Mint az adatadminisztrátornál, itt is vannak olyan ismeretek amelyeket a rendszer-adminisztrátornak és az adatbázis-adminisztrátornak is tudnia kell. De a rendszer-adminisztrátor általában nem fogja megérteni a fizikai adatbázist úgy, mint az adatbázis-adminisztrátor, és az adatbázis-adminisztrátor valószínűtlen, hogy megérti a telepítést és a rendszerszoftver részletes technikai kapcsolatait, mint a rendszer-adminisztrátor. Azonban mindkét munkakör hatékonyabb, ha van bizonyos ismeret a másik területéről.
Ha nem létezik rendszer-adminisztrációs csoport, vagy nem az adatbázis-kezelő rendszerre összpontosít, akkor általában az adatbázis-adminisztrátor vállalja az adatbázis-kezelő rendszerrel kapcsolatos rendszer-adminisztrátori és programozói felelősséget is.

Adatbázis-adminisztrátori feladatok

Az adatbázis-adminisztrátornak különböző területeken változatos feladatokat kell elvégeznie ahhoz, hogy egy cég adatai és adatbázisai hasznosak, használhatóak, elérhetőek, és korrektek legyenek. Ezek a területek magukban foglalják az adatbázistervet, a teljesítménymonitorozását, a hangolást, az adatbázis-elérhetőséget, a biztonságot, a mentést és a helyreállítást, az adatintegritást, a migrációt és igazában mindent, ami érinti a cég adatbázisait.

Adatbázisterv

A relációs adatbázisok megfelelő tervezéséhez és létrehozásához az adatbázis-adminisztrátornak meg kell tanulnia a relációs tervezést és ragaszkodnia kell a bevált gyakorlathoz. Az adatbázis-adminisztrátornak meg kell értenie a relációs elméletet és a relációs adatbázis-kezelő rendszer (Relational Database Management System, RDBMS) jellegzetes eszközrendszerét, amelyeket az adatbázis létrehozásakor használ. Az adatbázisterv megköveteli a koncepcionális és logikai adatmodellezési technikák alapos megértését. Egy relációs adatbázis tervezésében alapvető az ER modell elkészítésének és értelmezésének a képessége.
Az adatbázis-adminisztrátornak képesnek kell lennie egy logikai adatmodellt fizikai adatbázisba leképezni. Az adatbázis-adminisztrátornak törekednie kell arra, hogy az általa létrehozott adatbázisterv és annak a fizikai leképezése az adatbázis-alkalmazások és a kliensek számára az adatbázis teljes élettartama alatt jól használható legyen.
Az adatbázis-tervezés ugyan fontos ismeret az adatbázis-adminisztrátor számára és az optimális adatbázis-tervezés fontos, de az adatbázis-adminisztrátor feladatkörének az adatbázis-tervezés csak egy kis részét teszi ki. Egy adatbázis-adminisztrátor több időt tölt adminisztrálással és hangolással, mint adatbázisok tervezésével és építésével.
Ez nem jelenti azt, hogy az adatbázis-tervezés nem fontos. Egy rossz relációs adatbázisterv olyan gyenge teljesítményű adatbázist eredményezhet, amely nem felel meg a cég szükségleteinek és amely nem hiteles adatokat tárol.
A jegyzet ezzel a témakörrel a továbbiakban nem foglalkozik, mert előtanulmányaink során ezeket az ismereteket alapszinten már megszereztük.
A következő feladatokat a jegyzet a későbbiekben egy vagy több fejezetben részletesen tárgyalja:

- **Teljesítménymonitorozás és hangolás,**
- **Elérhetőség,**
- **Adatbázis-biztonság és feljogosítás,**
- **Mentés és helyreállítás,**
- **Adatbázis-kezelő rendszer verziómigrációja.**

Ez utóbbi témakört egyrészt az Adatbázis-környezet létrehozása című fejezetben, másrészt a Változáskezelés című fejezetben találhatjuk meg.

Adatintegritás

Az adataink tárolásához az adatbázist különböző szempontok szerint kell megtervezni. A megtervezésnél azt is figyelembe kell venni, hogy az adatokat olyan módon tároljuk, hogy azok ne sérüljenek és ne romoljanak meg. Ehhez az adatbázis-kezelő rendszer eszközrendszerét kihasználva az adatbázis-adminisztrátor integritási szabályokat alkalmaz az adatainkon. Az adatbázisoknál háromféle integritásról beszélhetünk: a fizikai, a szemantikai és a belső integritásról.
A *fizikai integritás*t az adatbázis-kezelő rendszer eszközeit használva lehet alkalmazni, mint például a tartományok és az adattípusok megadása. Az adatbázis-adminisztrátor kiválasztja a megfelelő adattípust a táblák egyes oszlopaihoz. Ez a művelet biztosítja azt, hogy csak annak a típusnak megfelelő adatok lesznek tárolva abban az adatbázis-táblabeli oszlopban. Az adatbázis-kezelő rendszer kikényszeríti az adatintegritást a típusokra vonatkozóan. Egész típusúként definiált oszlopban csak egész értékeket tárolhatunk. Ha nem számot vagy nem egész típusú értéket szúrunk be az oszlopba, akkor az hibát fog okozni. Az adatbázis-adminisztrátor további megszorításokat adhat meg, hogy még jobban meghatározza annak az adatnak a típusát, amelyet az adatbázis-táblabeli oszlopban lehet tárolni. A legtöbb relációs adatbázis-kezelő rendszer biztosítani tudja a következő típusú megszorításokat:

- Hivatkozási megszorítás, amelyet arra használ az adatbázis-adminisztrátor, hogy megadja azokat az oszlopokat, amelyekkel a táblák közötti kapcsolatot definiálja. A hivatkozási megszorítást a hivatkozási integritás megvalósítására használjuk.

- Egyediségi (Unique) megszorítás, amely azt biztosítja, hogy egy oszlop vagy oszlopok halmazának értékei csak egyszer fordulnak elő egy táblában.
- Ellenőrző (Check) megszorítás, amelyet akkor használ az adatbázis-adminisztrátor, ha egy összetettebb megszorítási szabályra van szüksége egy tábla oszlopán vagy oszlopainak halmazán. Tipikusan SQL-lel definiáljuk, és egy feltételt határozunk meg vele azokra az értékekre, amelyek egy oszlopon vagy egy oszlop halmazán megengedhetőek.

A *szemantikus integritás*t sokkal bonyolultabb felügyelni és kevésbé könnyű definiálni. A szemantikus integritásra példa az adatok minősége az adatbázisban. A legtöbb esetben nem elegendő a fizikai integritási szabályokat megadni ahhoz, hogy pontosan leírjuk, hogy milyen adatokat kívánunk tárolni. Az adatminőség biztosításához még más egyéb módszerekre is szükség van. Például egy ügyfél adatbázis nagyon gyenge minőségű, ha az ügyfél rekordok 25%-ánál rossz címet vagy telefonszámot tartalmaz. Nem létezik olyan fizikai módszer, amely szisztematikusan biztosítani tudná az adatok pontosságát. Az adatminőséget megfelelő alkalmazáskódokkal, üzleti fogásokkal, és különleges adatpolitikával, illetve adattisztítással és jól definiált folyamatokkal lehet támogatni. Másik példa a szemantikus integritásra a redundancia. Ha egy adatelem az adatbázisban redundánsan van tárolva, az adatbázis-adminisztrátornak dokumentálnia kell ezt a tényt és dolgoznia kell azon, hogy a redundáns adatokat valamilyen módon szinkronizáltan tartsa.

A *belső integritás*t az adatbázis-kezelő rendszer a belső struktúrákban és kódokban lévő hivatkozások és mutatók konzisztenciájának biztosításával tartja fönn. A legtöbb esetben az adatbázis-kezelő rendszer jól végzi a munkáját ezen a téren. Az adatbázis-adminisztrátornak tisztában kell lennie a belső integritás fontosságával, és azt is tudnia kell, hogy hogyan birkózzon meg azzal, ha az adatbázis-kezelő rendszer ezen a téren hibázik. Az adatbázis-kezelő rendszerbeli belső integritás létfontosságú a következő területeken:

- Indexkonzisztencia: egy index igazából nem más, mint adatbázistáblák adataira mutató mutatók rendezett listája. A rendezés az adatbázistábla valamely oszlopa vagy oszlopai szerint történik. Ha valamilyen okból az index nem lesz az adatokkal szinkronban, akkor indexelt eléréskor nem biztos, hogy az adatbázis-kezelő rendszer a megfelelő adattal tér vissza. Az adatbázis-adminisztrátornak vannak olyan eszközei, amelyekkel ellenőrzi és kijavítja ezeket a hibákat.
- Mutatókonzisztencia: Néha a nagy multimédiás objektumok nem ugyanabban a fizikai állományban vannak tárolva, mint a többi adat. Ebben az esetben az adatbázis-kezelő rendszer mutatóstruktúrát használ arra, hogy a multimédiaadatokat szinkronban tartsa az alaptábla adataival.
- Mentési konzisztencia: Néhány adatbázis-kezelő rendszer esetenként helytelen mentési másolatot készít, amely a helyreállításkor nem használható. Fontos, hogy ezeket az eseteket az adatbázis-adminisztrátor felismerje és kijavítsa.

Ezermester

Az alkalmazások központjában az adatbázis áll. Ha az adatbázis nem megy, az alkalmazások sem mennek, és akkor az üzlet sem megy. Ezért áll az adatbázis-adminisztrátor is az üzlet középpontjában.

Az adatbázisok az IT infrastruktúra majdnem minden komponensével kapcsolatban állnak. Az IT infrastruktúra magában foglalhatja a következőket:

- programozási nyelvek és környezetek
- adatbázis- és folyamattervező eszközök
- tranzakciófeldolgozó eszközök
- üzenetsorba-állító eszközök
- hálózati szoftverek és protokollok
- hálózati hardverek

- különböző operációs rendszerek
- adattároló hardverek és szoftverek
- operációs rendszer biztonsági csomagjai
- különféle tárolási hardverek, mint például szalagos egység
- nem adatbázis-kezelő rendszerbeli adathalmaz- és állománytárolási technikák
- adatbázis-adminisztrációs eszközök
- rendszerkezelő eszközök és keretrendszerek
- működést vezérlő szoftverek, mint kötegelt ütemező szoftver
- szoftverelosztó megoldások
- internetes és webes adatbázisok és alkalmazások
- kliens/szerver fejlesztési technikák
- objektumorientált és komponens alapú fejlesztési technológiák és technikák
- a számítási kapacitásért felelős számítógép

Lehetetlen mindezeket a technikákat jól megtanulni, de az adatbázis-adminisztrátornak szükséges némi ismerettel rendelkeznie ezekről a területekről. És ettől sokkal fontosabb, hogy az adatbázis-adminisztrátornak tudnia kell néhány olyan embernek a telefonszámát, aki szakértő az adott területen. Ha az adatbázis-adminisztrátor hibát észlel az adatbázis elérésében vagy a teljesítményében, és az ok nem az adatbázisból ered, akkor találnia kell olyan szakembert, aki segít megoldani az adott problémát vagy tudnia kell, hogy a hibát hova kell jelentenie.

Hány adatbázis-adminisztrátorra van szüksége egy cégnek ?

Az egyik legnehezebb dolog meghatározni egy cég adatbázisainak online állapotához és hatékony működéséhez szükséges adatbázis-adminisztrátorok optimális számát. Sok cég minimális adatbázis-adminisztrátori személyzettel próbál dolgozni, mert azt gondolják, hogy minél kevesebb az alkalmazott, annál kevesebb a költség. De a feltételezés nem mindig igaz. Egy túlórázó adatbázis-adminisztrátor hibázhat, amely leálláshoz és műveleti problémákhoz vezethet. Az ilyen hibák költsége több lehet, mint egy plusz adatbázis-adminisztrátori fizetés.

Az adatbázis-adminisztrátorok optimális száma sok tényezőtől függ:

- Az adatbázisok száma: minél több adatbázist kell támogatni, annál összetettebb az adatbázis-adminisztrátor feladata.
- Az adatbázisok mérete: minél nagyobbakat kell támogatni, annál nehezebb az adatbázis-adminisztrátor feladata. Ha egy SQL utasítás sokáig fut, több idő kell a hangolásra is.
- A felhasználók száma: Több felhasználó esetén nehezebb biztosítani az online, optimális teljesítményű adatbázist. Ráadásul több a problémák, a hívások száma.
- Az alkalmazások száma: Minél több az alkalmazás, annál nagyobb nyomás nehezedik az adatbázisra teljesítmény, erőforrás és elérhetőség értelemben.
- Szolgáltatási szintről szóló megállapodás (service-level agreement) (SLA): minél szigorúbb a szolgáltatási szintről szóló megállapodás, annál nehezebbé válik az adatbázis-adminisztrátornak a szolgáltatást szállítania. Például ha a szolgáltatási szintről szóló megállapodás a tranzakcióktól másodperc töredéknyi idejű válaszidőt követel, az sokkal bonyolultabb, mint ha három másodpercnyi válaszidőt követelnének.
- Elérhetőségi követelmények: Az adatbázis-adminisztráció egyszerűbbé válik, ha van engedélyezett ütemezett leállási idő. Néhány adatbázis-adminisztrátor feladathoz leállás kell, vagy egyszerűbb megtenni, ha áll az adatbázis. Az e-business-hez és a webszolgáltatásokhoz 24/7 adatbázis-elérhetőség szükséges.
- A leállás hatása: Minél nagyobb a pénzügyi hatás egy elérhetetlen adatbázis esetén, annál nagyobb a nyomás az adatbázis-adminisztrátorra, hogy nagyobb adatbázis-elérhetőséget biztosítson.

- Teljesítménykövetelmények: Minél teljesítményorientáltabb az adatbázis-elérés, annál bonyolultabb lesz az adatbázis-adminisztráció.
- Alkalmazások típusai: A támogatott alkalmazástípusoknak közvetlen hatása van a szükséges adatbázis-adminisztrátorok számára. Az üzletmenet tekintetében kritikus alkalmazásokhoz szükséges adatbázis-kezelő rendszer és az adatbázis különbözik a nem kritikus alkalmazásoktól. Üzletkritikus alkalmazás az olyan alkalmazás, amelyhez ha nem férünk hozzá, akkor az az üzlet menetében veszteséget okoz. Az üzletkritikus alkalmazások sokkal jobban megkövetelik az állandó monitorozást az elérhetőség biztosítása érdekében. Egy OLTP (online transaction processing, online tranzakciófeldolgozó) rendszernek más jellemzői vannak és más adminisztrációt követel meg, mint egy OLAP (online analitical processing, online analitikus feldolgozó) rendszer. Az OLTP rendszer tranzakciói rövidebb ideig tartanak, mint az OLAP lekérdezések, OLTP rendszerek írási és olvasási műveleteket végeznek, az OLAP rendszereknél az olvasás a domináns. Ezért különböző adatbázis-adminisztrátori eljárásokra van szükségük.
- Változékonyság: a gyakran változó adatbázisok több adatbázis-adminisztrátort követelnek meg. Egy statikus adatbázis-környezet kevés változtatást igényel, amely nem ugyanolyan szintű adatbázis-adminisztrátori hatékonyságot követel meg, mint egy gyakran változó adatbázis-környezet. Sajnos a legtöbb adatbázis és alkalmazás változékonysági szintje hajlamos az idővel drámaian változni. Gyakran nagyon nehéz megállapítani, mennyire lesz változékony egy teljes adatbázis-környezet az életciklusa alatt.
- Az adatbázis-adminisztrátori személyzet tapasztalata: A létező adatbázis-adminisztrátor csoport tudása határozza meg a további adatbázis-adminisztrátorok szükségét. Egy sok ismerettel rendelkező adatbázis-adminisztrátori csoport többet teljesít, mint egy kezdő csapat. A személyzeti követelmények tekintetében a tudás többet jelent, mint a tapasztalat. Egy jó szakértelemmel rendelkező adatbázis-adminisztrátor két év tapasztalattal könnyen felülmúlhatja a 10 éve dolgozó veteránt, aki kiégett és nem motivált.
- A programozói személyzet tapasztalata: Ha az alkalmazásfejlesztőknek nincs nagy szakértelme az adatbázisokhoz és az SQL programozáshoz, akkor az adatbázis-adminisztrátornak az alkalmazásfejlesztés folyamatában jobban részt kell vennie. Az adatbázis-adminisztrátornak kell olyan feladatokat ellátni, mint összetett SQL utasítások összeállítása, SQL és alkalmazáskód elemzése, nyomkövetés, hangolás, és a kapcsolat biztosítása. Ahogy a programozó személyzet tapasztalata nő, úgy csökkennek az adatbázis-adminisztrátortól elvárt programozói feladatok.
- Végfelhasználók tapasztalata: Ha a végfelhasználók ad hoc SQL-lel közvetlenül elérhetik az adatbázist, akkor a tudásuk szintje közvetlenül hat az adatbázis-adminisztrátor feladatainak összetettségére.
- Az adatbázis-kezelő rendszerek változatossága: Minél heterogénebb a környezet, annál nehezebbé válik adminisztrálni azt. Például tapasztalatot szerezni és fenntartani Oracle-ben és DB2-ben is, sokkal nehezebb, mint csak az egyikben gyakorlatot szerezni. Továbbá, ha több különböző típusú adatbázis-kezelő rendszer van telepítve, az adatbázis-adminisztráció sokkal nehezebbé válik.
- Adatbázis-adminisztrációs eszközök: Az adatbázis-kezelő rendszerek szállítói és más cégek ajánlanak olyan eszközöket, amelyek automatizálják az adatbázis-adminisztrátor feladatait. Az automatizált eszközök megkönnyítik az adatbázis-adminisztrációt. Az adatbázis-adminisztrátor feladatai kevésbé lesznek összetettek, ha több eszköz érhető el.

Fejlesztő, teszt és termelési rendszer

Az adatbázis-adminisztrátor legalább két különböző környezetet hoz létre és támogat egy minőségi adatbázis megvalósítása esetén: a fejlesztő és teszt rendszert, valamint a termelési rendszert.

Nagyon gyakran a fejlesztő és teszt rendszer is el van különítve. A teljesen elkülönített teszt és termelési környezet segítségével lehet a termelési rendszer integritását és a teljesítményét biztosítani. Az új fejlesztés és a karbantartási munkálatok először mindig a teszt környezetben valósulnak meg. A működő alkalmazások pedig a termelési környezetben futnak. Ha a két környezet nem megfelelően van elkülönítve, akkor a cég fejlesztési tevékenysége az üzleti működést károsíthatja. Ha a fejlesztés egy korai állapotában egy eltévedt programkód elérheti vagy módosíthatja a termelési rendszer adatait, az teljesítményproblémákat, érvénytelen adatokat vagy akár üzleti veszteséget, jogi problémákat okozhat.
A teszt és a termelési környezetnek adatszinten nem kell teljesen azonosnak lennie. A termelési környezet tartalmazza az összes olyan adatot, amely a működő alkalmazások támogatásához kell. Azonban a teszt környezetben csak az adatok egy részhalmazára van szükség az elfogadható alkalmazásteszteléshez. Továbbá a teszt adatbázis-kezelő rendszer megvalósítás nem követel ugyanannyi erőforrást, mint a termelési rendszer. Például kevesebb memóriára van szükség, vagy az adatoknak kevesebb helyet kell lefoglalni, amelyhez kevesebb eszköz szükséges. A tesztrendszerre esetleg az adatbázis-kezelő rendszer szoftverének egy újabb verziója van feltéve. Ennek az az egyik lehetséges oka, hogy így az adatbázis-kezelő rendszer új verziója miatt keletkező hibákat ki lehet javítani, mielőtt azok a termelési rendszerre kerülnének.
A teszt és a termelési környezet felépítésének hasonlónak kell lennie. A programozók az alkalmazásokat a fejlesztői környezetben hozzák létre, és a tesztkörnyezetben tesztelik. Ezeket az alkalmazásokat majd az adatbázis-adminisztrátor fogja a termelési rendszerbe áthelyezni. A termelési rendszerben lévő alkalmazásokat a programozók már nem módosíthatják, sőt el sem érhetik azokat, nem kapnak hozzá jogot. A programozó csak akkor tudja az alkalmazásokat megfelelően megírni és tesztelni, ha a termelési és a tesztrendszernek ugyanaz a felépítése.
Lehet, hogy az adatbázis-adminisztrátornak a tesztkörnyezetben több adatbázis-másolatot kell készítenie, hogy a párhuzamosan folyó fejlesztéseket támogassa. A programozóknak tudniuk kell kezelni a tesztadatbázisok tartalmát. A programozóknak a fejlesztés során többször, ugyanolyan adatokon kell az alkalmazásokat futtatniuk, hogy ellenőrizni tudják, hogy mit csinál az alkalmazás. Természetesen az alkalmazás a legtöbb esetben adatot is fog módosítani. A programozónak ezért szüksége van arra, hogy a kezdeti tesztadatokkal tudjon újra dolgozni, különben nem látja, hogy az alkalmazáskód változtatása valóban a szükséges változtatást eredményezte. Az adatbázis-adminisztrátornak segítenie kell a programozók munkáját. Az adatbázis-adminisztrátor szolgáltatja a kezdeti tesztadatokat. A kezdeti tesztadatok ismételt betöltését azonban a programozó el tudja végezni, de mindenképp meg tudja tanulni az adatbázis-adminisztrátortól. Nagy segítség lehet a tesztadatok visszaállításában a betöltő (LOAD) és kimentő (UNLOAD) segédprogramok. A programozó vagy az adatbázis-adminisztrátor a tesztfutás előtt az adatbázist tesztadatokkal feltölti, majd a tesztfutás után megvizsgálja, hogy a programlogika helyes-e. A vizsgálatot a program kimenetére, illetve az adatbázis tesztfutás előtti és utáni tartalmára alapozhatja. Ha nem helyes a programlogika, akkor a programozó megismételheti a folyamatot, amihez először újra betölti a tesztadatokat az adatbázisba majd újrateszteli az alkalmazást.
Nehéz megjósolni, hogy a tesztalkalmazásoknak milyen lesz a teljesítménye az első futáskor a termelési környezetben. Az adatbázis-adminisztrátor azonban itt is segít. Egy relációs adatbázis-kezelő rendszer általában biztosít egy eszközt, amely statisztikai információt gyűjt az adatbázis tartalmáról. Ezt a statisztikát használja a relációs optimalizáló, hogy meghatározza azt, hogy az SQL hogyan nyerje ki az adatokat. De ne feledjük el, hogy tesztadatbázisban kevesebb adat van, mint az élesben. Az adatbázis-adminisztrátor ilyen esetben szkripteket írhat, amellyel a termelési rendszer statisztikáit felolvassa és azokat a tesztkörnyezetbe másolja. Így segít a fejlesztőknek pontosabban becsülni, hogy a tesztalkalmazásoknak milyen lesz a teljesítménye a termelési rendszerben.
Általában nem elég csak ezt a két (termelési és fejlesztési-teszt) adatbázis-környezetet megvalósítani egy cég adatbázisa esetén. Például az összetett alkalmazásfejlesztési projektek miatt

szükség lehet két vagy több különböző fejlesztési környezetre. Szükséges lehet egy környezetre, ahol az alkalmazásokat külön-külön fejlesztik, és tesztelik, illetve egy másik környezetre, ahol ezeket az egyedi alkalmazásokat integrálják a már meglévőkkel, és azt vizsgálják, hogy hogyan tudnak együttműködni. Más esetben minőségbiztosítási környezetre lehet szükség. Ebben a környezetben szigorú teszteléseket hajtanak végre az új és a módosított programokon, mielőtt azok a termelési környezetbe kerülnének.
A termelési rendszerből a tesztrendszerbe általában nem az éles adatok kerülnek át. A bizalmas adatokat nagyon gyakran megszűrik, vagy tisztítják, hogy illetéktelenek ne férhessenek hozzá olyan adatokhoz, amelyekhez nincs jogosultságuk.

Összefoglalás

A fejezet bemutatta az alapvető alapfogalmakat: adatbázis, adatbázis-kezelő rendszer, adatbázis-adminisztrátor. Különbséget tettünk az adatbázis-adminisztrációs feladatokhoz való proaktív és reaktív hozzáállás között. Megnéztük, hogy milyen feladatai vannak az adatbázis-adminisztrátornak az alkalmazásfejlesztés életciklusa alatt. Különbséget tettünk az adatadminisztrátor, az adatbázis-adminisztrátor és a rendszer-adminisztrátor között, illetve részleteztük a feladataikat. Az adatbázis-adminisztrátor feladatait csak röviden részleteztük ebben a fejezetben, hiszen a teljes jegyzet az ő feladatairól szól. Azonban egyes feladatokkal mégis foglalkoztunk, mint az adatbázisterv, az adatintegritás, vagy az ezermester, mert ezekkel a témákkal ebben a jegyzetben a továbbiakban nem foglalkozunk. Felsoroltuk, hogy milyen tényezőktől függ, hogy egy cégnek hány adatbázis-adminisztrátorra van szüksége. És legvégül felhívtuk a figyelmet arra, hogy ha egy cégnek van egy adatbázisa, akkor az általában valójában nem egy adatbázis-környezetet jelent, hanem legalább kettőt, de még inkább annál is többet.

Ellenőrző kérdések

1. Mi az az adatbázis?
2. Mi az az adatbázis-kezelő rendszer?
3. Ki az az adatbázis-adminisztrátor?
4. Mit jelent a reaktív közelítés az adatbázis-adminisztrációban?
5. Mit jelent a proaktív közelítés az adatbázis-adminisztrációban?
6. Milyen feladatai vannak az adatbázis-adminisztrátornak az alkalmazásfejlesztés életciklusának a fázisai alatt?
7. Mi a koncepcionális adatmodell?
8. Mi a logikai adatmodell?
9. Mi a fizikai adatbázisterv?
10. Mi a feladata az adatadminisztrátornak? Miben különbözik az adatbázis-adminisztrátortól?
11. Mi a feladata a rendszer-adminisztrátornak? Miben különbözik az adatbázis-adminisztrátortól?
12. Soroljuk fel röviden az adatbázis-adminisztrátor feladatait!
13. Mit jelent a fizikai integritás? Mit jelent a szemantikai integritás? Mit jelent a belső integritás?
14. Milyen IT infrastruktúrabeli komponensekkel áll kapcsolatban az adatbázis? Soroljuk fel őket!
15. Mitől függ az, hogy egy cégnek hány adatbázis-adminisztrátorra van szükséges?
16. Mik a különbségek és a hasonlóságok a teszt, a fejlesztői és a termelési rendszerek között?

2. fejezet – Az adatbázis-környezet létrehozása

Az adatbázis-adminisztrátor munkájának az egyik feladata az adatbázis-kezelő rendszer kiválasztása és annak telepítése, frissítése. A kiválasztásban a szakmai szempontok mellett természetesen nagy szerepet játszik a teljes felépítendő környezetnek, és a támogatásnak a költsége is. Ezért azt kell mondjuk, hogy a legtöbb cégnél a cég vezetősége dönt arról, hogy milyen adatbázis-kezelő rendszert választanak. Az adatbázis-adminisztrátornak nagy feladat a szakmai érvek felsorolása az egyes adatbázis-kezelő rendszerek, illetve környezetek mellett vagy ellen. Természetesen a telepítés és a frissítés az ő dolga lesz, ha nincs külön rendszer-adminisztrátori csoport.

Stratégia az adatbázis-kezelő rendszerek terén

Az adatbázis-kezelő rendszer kiválasztása

Az adatbázis-adminisztrátor csoport feladata a cégen belül a támogatott adatbázis-kezelő rendszer, és a hozzá kapcsolódó különböző termékek politikájának meghatározása. Ha lehetséges minimalizáljuk a különböző típusú adatbázis-kezelő rendszerek és termékek számát. Ha többféle operációs rendszerre és többféle típusú hardverre kell adatbázis-kezelő rendszert telepíteni, akkor is érdemes egy alapértelmezett típusú adatbázis-kezelő rendszert választani. Ettől az alapértelmezettől csak akkor érdemes eltérni, ha a cég üzletmenete ezt megkívánja. De akkor is olyan adatbázis-kezelő rendszert válasszunk, amely megfelel az adatbázis-adminisztrátori csoport technikai elvárásainak.

A legtöbb nagyobb adatbázis-kezelő rendszernek hasonló eszközei vannak. Ha egy adatbázis-kezelő rendszernek valamilyen eszköze vagy funkcionalitása ma nem létezik, akkor 18-24 hónapon belül valószínűleg a piacra fogják dobni. Ezért nem érdemes a döntést kizárólag arra alapozni, hogy az adott eszközt csak az adott adatbázis-kezelő rendszer ismeri.

Napjainkban 3 nagy zárt adatbázis-kezelő rendszert szállító cégről szoktunk beszélni: az IBM-ről amely a DB2-t szállítja, az Oracle-ről, és a Microsoftról, amelynek a terméke az SQL Server. Ha adatbázis-kezelő rendszert szeretnénk választani, akkor érdemes először az ő kínálatukat megnézni. Ennek az is az oka, hogy ezeknek a termékeknek van a legnagyobb támogatottsága a piacon, azaz ezekhez a termékekhez találunk a legkönnyebben támogató, programozó szakembereket, helpdesk szolgálatot, assistance-t. A döntésnél vegyük figyelembe azt is, hogy milyen platformra lehet őket telepíteni: a Microsoft SQL Server csak Microsoftos operációs rendszeren fut, míg a DB2 és az Oracle esetén ennyire erős operációsrendszer-megszorítás nincs. A döntésnél érdemes azt is megvizsgálni, hogy a szállító cégeknek milyen széles a technológiai portfóliója. Azaz nézzük meg, hogy a cég tud-e ajánlani hardvert, operációs rendszert, adatbázis-kezelő rendszert, alkalmazásszervert és más adatbázisra épülő alkalmazásokat, szolgáltatásokat. Ha a cégünk ezeket be tudja építeni a saját hardver- és szoftverstratégiájába, akkor érdemes egy cégtől több megoldást is választani. Ezeknek a termékeknek így jobb lesz a támogatottsága, illetve mivel egymáshoz vannak igazítva, jobban ki tudják használni egymás lehetőségeit.

A piacon vannak más kiváló adatbázis-kezelő rendszer termékek is, amelyeket érdemes tekintetbe venni bizonyos helyzetekben. Nyílt kódú szoftverek közül például választhatjuk a PostgreSQL-t vagy a MySQL-t, vagy minialkalmazásokhoz az SQLLite-t.

Ha a 3 nagy szállító termékei közül választunk, akkor egy olyan adatbázis-kezelő rendszert kapunk, amely elegendő funkcionalitással rendelkezik, ugyanakkor kevés kockázattal jár. Ha kisebb cégek termékei közül választunk adatbázis-kezelő rendszert, akkor nagyobb kockázatnak tesszük ki magunkat. A kisebb cégek adatbázis-kezelő rendszereinek kisebb a funkcionalitása és a támogatottsága is.

A megfelelő adatbázis-kezelő rendszer kiválasztásához stratégiára és tervre van szükség. Az adatbázis-kezelő rendszereket több szempontból érdemes megvizsgálni, összehasonlítani. A következő tényezőket mindenképp vegyük figyelembe:

- Operációsrendszer-támogatás: Támogatja-e az adatbázis-kezelő rendszer a cégünknél használt operációs rendszert abban a verzióban, amelyet most a használunk és később használni tervezünk?
- A cég típusa: Vegyük figyelembe a cég filozófiáját, amikor adatbázis-kezelő rendszert választunk. Néhány cég nagyon konzervatív és ragaszkodik a megszokott környezetéhez. A kormányzatok, a pénzügyi cégek, a biztosító és egészségügyi cégek általában konzervatívak. A liberálisabb cégek gyakran választanak alternatív megoldásokat. Nem szokatlan, hogy a gyárak, az egyetemek kevésbe konzervatívak. És végül vannak cégek, amelyek nem bíznak a Windowsban és a Unixot részesítik előnyben, ez a választás eleve kizár néhány terméket.
- Benchmarkok: Milyen teljesítmény benchmarkok (mérések) érhetőek el az adatbázis-kezelő rendszer szállítójánál és az adatbázis-kezelő rendszer használóinál?
- Skálázhatóság: Tud-e az adatbázis-kezelő rendszer annyi felhasználót és akkora adatbázisméretet támogatni, amelyet meg szeretnénk valósítani?
- Támogató szoftvereszköz elérhetősége: Vannak-e elérhető eszközök az adatbázis-kezelő rendszerhez, mint például lekérdező és elemző eszköz, adattárház-támogató eszköz, adatbázis-adminisztrációs eszköz, mentési és helyreállítási eszköz, teljesítménymonitorozó eszköz, kapacitástervező eszköz, adatbázis-segédprogramok, illetve támogat-e az adatbázis-kezelő rendszer különböző programozási nyelveket?
- Szakemberek: Van-e elegendő megfelelően képzett adatbázis-szakember az adatbázis-kezelő rendszerhez? Vegyük figyelembe az adatbázis-adminisztrátorokat, a technikai támogatást nyújtó személyzetet (rendszerprogramozó és -adminisztrátor, stb.) és az alkalmazásprogramozókat.
- A tulajdonjog költsége: Az adatbázis-kezelő rendszer tulajdonjoga összesen mennyibe kerül? A tulajdonjog teljes összege tartalmazza az adatbázis-kezelő rendszer licencének az árát, a támogató szoftverek licencének az árát, a programozó, támogató, adminisztráló szakemberek költségét és az adatbázis-kezelő rendszer működéséhez szükséges erőforrások költségét.
- Új verziók ütemezése: Milyen gyakran ad ki az adatbázis-kezelő rendszer szállítója új verziót? Néhány adatbázis-kezelő rendszer esetén az új verziók nagyon gyakran, akár minden 12-18 havonta megjelennek. Ez lehet jó is és rossz is a mi szempontunkból. Ha a cégünk konzervatív, akkor a gyakran változó adatbázis-kezelő rendszert nehéz támogatni.
- Ügyfél-referencia: Van-e az adatbázis-kezelő rendszer szállítójának jelenleg felhasználói referenciája? Találunk-e más felhasználókat, akik elfogulatlan válaszokat adnak? Beszéljünk a jelenlegi felhasználókkal, és tudjuk meg például a következőket: Általában úgy működnek-e a dolgok, ahogy reklámozták? Milyen a támogatás? Sok hiba van-e az adatbázis-kezelő rendszerben? Az új verzióknak milyen a minősége?
- Ár/érték hányados: Lehet, hogy az adott termék nem a legmegfelelőbb, de az adott üzleti modellben ez térül meg.

Adatbáziskezelőrendszer-architektúrák

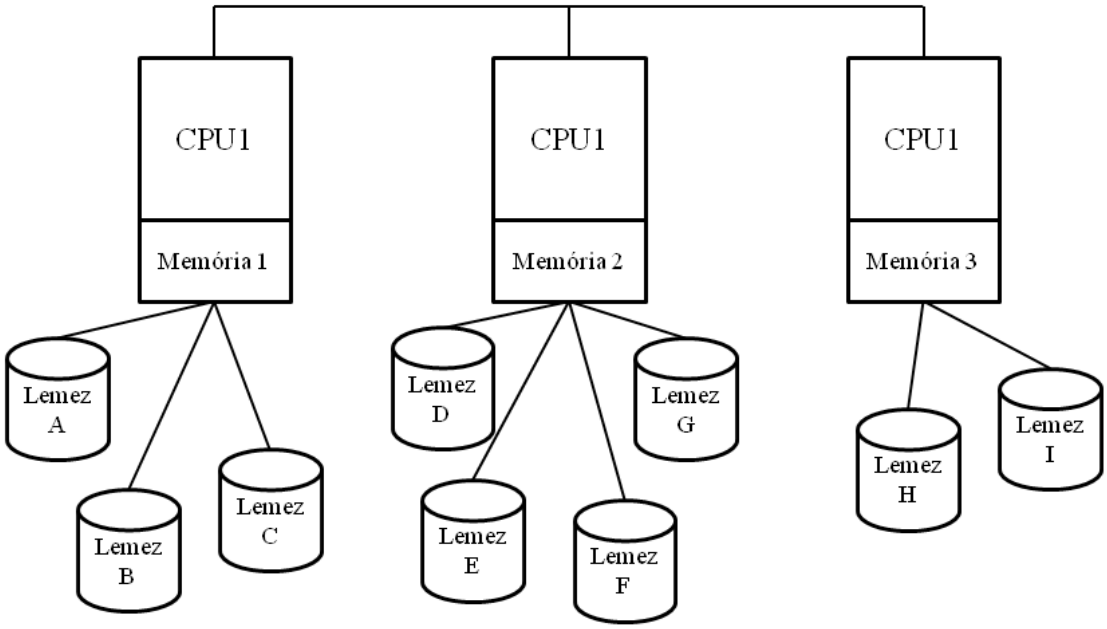
Az adatbáziskezelőrendszer-architektúráknak általában 3 szintje áll rendelkezésre: vállalati (enterprise), személyi (personal) és mobil.

A vállalati adatbázis-kezelő rendszer skálázhatósághoz és nagy teljesítményhez van tervezve. A vállalati adatbázis-kezelő rendszer képes nagyon nagy adatbázisokat, nagyon sok konkurens felhasználót, és többféle típusú alkalmazást támogatni. A vállalati adatbázis-kezelő rendszer nagygépeken fut, több processzort, párhuzamos lekérdezéseket támogat, stb.

A személyi adatbázis-kezelő rendszer egyfelhasználós, általában PC-n fut. Ilyen például a Microsoft Access. De a nagy adatbázis-kezelő rendszerek szállítói is tudnak személyi változatot ajánlani, mint amilyen az Oracle Express. Ezek általában sokkal kevesebbe kerülnek, mint a vállalati adatbázis-kezelő rendszerek, nem skálázhatóak, és nem alkalmasak többfelhasználós alkalmazásokhoz.
A mobil adatbázis-kezelő rendszer egy különleges változata a vállalati adatbázis-kezelő rendszernek. A távoli felhasználókhoz tervezték, akik nem rendszeresen kapcsolódnak a hálózathoz. A mobil adatbázis-kezelő rendszer a laptop vagy a kézi eszköz helyi adatbázisát éri el és módosítja. Amikor a laptopot vagy a kézi eszközt a hálózathoz kapcsoljuk, akkor az adatbázis-kezelő rendszer lehetőséget kap, hogy a központi adatbázis-szerverrel szinkronizálja a helyi adatbázist.

Klaszterezés (fürtözés)

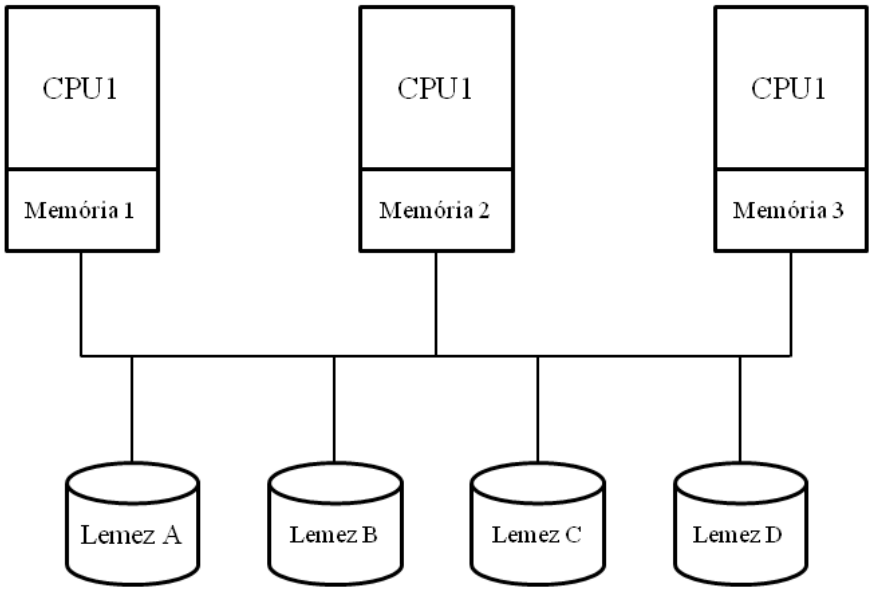
A klaszterezés több független számítógép összekapcsolása, melyeknek az együtt végzett munkája a felhasználó számára úgy tűnik, mintha egy önálló, magas elérhetőségű rendszert használnánk. A skálázhatóság és elérhetőség növelése érdekében az adatbázis-kezelő rendszerek általában rendelkeznek klaszter támogatással. Két domináns architektúra létezik: a megosztás nélküli (shared-nothing) (*lásd* 2-1. ábra) és a megosztott-lemez (shared-disk) (*lásd* 2-2. ábra) architektúra.



2-1. ábra Megosztás nélküli (shared-nothing) architektúra

A megosztás nélküli architektúrában az egyes rendszereknek saját privát erőforrásaik vannak (memória, lemez, stb.). A folyamatok a gépeket összekötő hálózaton keresztül küldött üzenetekkel kommunikálnak. Egy kliens kérés automatikusan az erőforrásgazdához lesz irányítva. Ha hiba történik az egyik csomóponton, akkor az erőforrás tulajdonlása automatikusan egy másik rendszerhez kerül. Ez úgy történhet meg, hogy a csomópontokon tárolt adatok a klaszter legalább egy másik csomópontján többszörözve van. A fő előnye a skálázhatóság. Akár ezer processzorig

skálázható a rendszer, mivel nem akadályozzák egymást.



2-2. ábra Megosztott-lemez (shared-disk) architektúra

Egy megosztott-lemez architektúrában az összekapcsolt rendszerek ugyanazokon a lemezeszközökön osztoznak. Minden processzornak van saját memóriája. Minden processzor közvetlenül címezhet minden tárat. Általában kisebb gépek esetén ez az architektúra kevésbé skálázható, mint a megosztás nélküli architektúrában. A megosztott-lemez architektúra nagygépes környezetben nagyvállalati folyamatokhoz alkalmas. Ha néhány nagygépet megosztott-lemez architektúrában kapcsolunk össze, nagyobb elérhetőséget és skálázhatóságot érünk el, mint sok közepes processzorú PC megosztás nélküli architektúrában történő összekapcsolásával.
A 2-3. ábra összehasonlítja a kétféle architektúrát.

Megosztott-lemez architektúra	Megosztás nélküli architektúra
gyorsan alkalmazkodik a változó munkaterheléshez	az egyszerűbb, olcsóbb hardvert jól ki tudja használni
magas elérhetőség	majdnem korlátlan skálázhatóság
a legjobban olyan környezetben teljesít, ahol nagyon sok olvasás történik	jól működik olyan környezetben, ahol nagy tömegű olvasási és írási művelet történik
az adatokat nem kell elosztani	az adatokat el kell osztani a klaszter tagjai között

2-3. ábra A megosztás nélküli architektúra és a megosztott-lemez architektúra összehasonlítása

Bármely architektúrát tekintve a klaszterezés legfontosabb előnye az elérhetőség növekedése. Sok esetben ez 99.999%-os elérhetőséget jelent.
A klaszterezésről további ismereteket még az Adatok elérhetősége című fejezetben találhatunk.

Az adatbázis-kezelő rendszer telepítése

A kiválasztott adatbázis-kezelő rendszert fel kell telepíteni. Céges környezetben a telepítés nem egyszerűen úgy néz ki, hogy betesszük a szállítótól kapott CD-t és a telepítő szoftver magától elvégzi a telepítést. Egy adatbázis-kezelő rendszer a cég szoftvereinek egy szerves része. Ahhoz, hogy hosszútávon jól működjön, a telepítést előre meg kell tervezni. Az adatbázis-adminisztrátornak ismernie kell a telepített adatbázis-kezelő rendszernek a követelményeit és elő kell készítene a környezetet az új adatbázis-kezelő rendszer fogadására.

Hardver- és operációsrendszer-követelmények

A telepítés előtt először az előfeltételeket kell megismerni. Minden adatbázis-kezelő rendszernek van telepítési útmutatója, amely az adatbázis-kezelő rendszer megfelelő működéséhez szükséges hardver- és operációsrendszer-követelményeket tartalmazza. Leírja például, hogy az adatbázis-kezelő rendszer megfelelő futásához milyen processzor, mennyi memória, illetve az operációs rendszernek melyik verziója szükséges. Azt is leírja, hogy az adatbázis-kezelő rendszer szoftverének telepítéséhez mennyi tárterület szükséges.
Ezenkívül leírja, hogy milyen környezet szükséges az adatbázis-kezelő rendszer különböző eszközeinek használatához, mint például a párhuzamos feldolgozásokhoz, az adattárházakhoz, stb.
Bizonyosodjunk meg arról, hogy az igényeinkhez szükséges adatbázis-kezelő rendszerbeli eszközök követelményeinek megfelel az új adatbázis-kezelő rendszer fogadására kialakított hardver- és operációsrendszer-környezet.

Tárolási követelmények

Egy adatbázis-kezelő rendszernek lemezterületre van szüksége, hogy fusson. Az adatbázis-kezelő rendszernek nem csak amiatt az egyszerű ok miatt van szüksége tárterületre, hogy az adatokat és az

azokhoz tartozó indexeket tárolja. Az adatbázis-kezelő rendszer megfelelő működéséhez és az adatbázis hiba nélküli kezeléséhez szükség van a következő struktúrákra, amelyek ugyancsak helyet foglalnak el a lemezen:

- Rendszerkatalógus vagy adatszótár, hogy az adatbázis-kezelő rendszer kezelje és kövesse az adatbázisokat, az adatbázis-objektumokat és a kapcsolódó információkat. Minél több adatbázis-objektumot tervezünk létrehozni, annál több helyre lesz szüksége a rendszerkatalógusnak. A tartalmát a Metaadatok kezelése című fejezetben részletezzük.
- Naplóállományok, amelyek az adatbázisban végrehajtott változásokat tartalmazzák. Ez aktív és archív naplókat, visszagörgető szegmenseket, és más naplókat tartalmaz. Az adatbázisnaplóhoz tartozó állományokról részletesebben az Adat- és tároláskezelés című fejezetben van szó.
- Indító és vezérlő állományok, amelyekre az adatbázis indításakor és inicializálásakor van szükség. Ezek az állományok olyan egyszerű információkat tartalmaznak, mint például az adatbázis neve, a létrehozásának a dátuma, információk az adatállományokról, a naplóállományokról, mentési információk, rendszerparaméterek értékei. Ezeknek az állományoknak a tartalmát az adatbázis-kezelő rendszer akkor is tudja használni, hogy ha az adatbázisbeli adatok nem érhetőek el a felhasználók számára. Karbantartási feladatoknál nagy szükség lehet rájuk.
- Ideiglenes vagy munkaállományok, melyekre az adatbázis-kezelő rendszernek adatrendezésre vagy más folyamatokhoz van szüksége.
- Alapértelmezett adatbázisok a rendszerstruktúrák tárolására. Ezek a struktúrák létfontosságúak az adatbázis-kezelő rendszer normál működéséhez és az adatbázis megfelelő kezeléséhez.
- Rendszer-nyomkövetési és hibadiagnosztikai állományok, amelyekbe az adatbázis-kezelő rendszer folyamatosan ír, információt szolgáltat a működéséről. Az adatbázis-kezelő rendszerbeli hibákat ezekből az állományokból fedezheti fel az adatbázis-adminisztrátor, még mielőtt azok nagyobb működésbeli hibát okoznának.
- Adatbázis-adminisztrátori adatbázisok az adminisztráláshoz, monitorozáshoz és hangoláshoz.

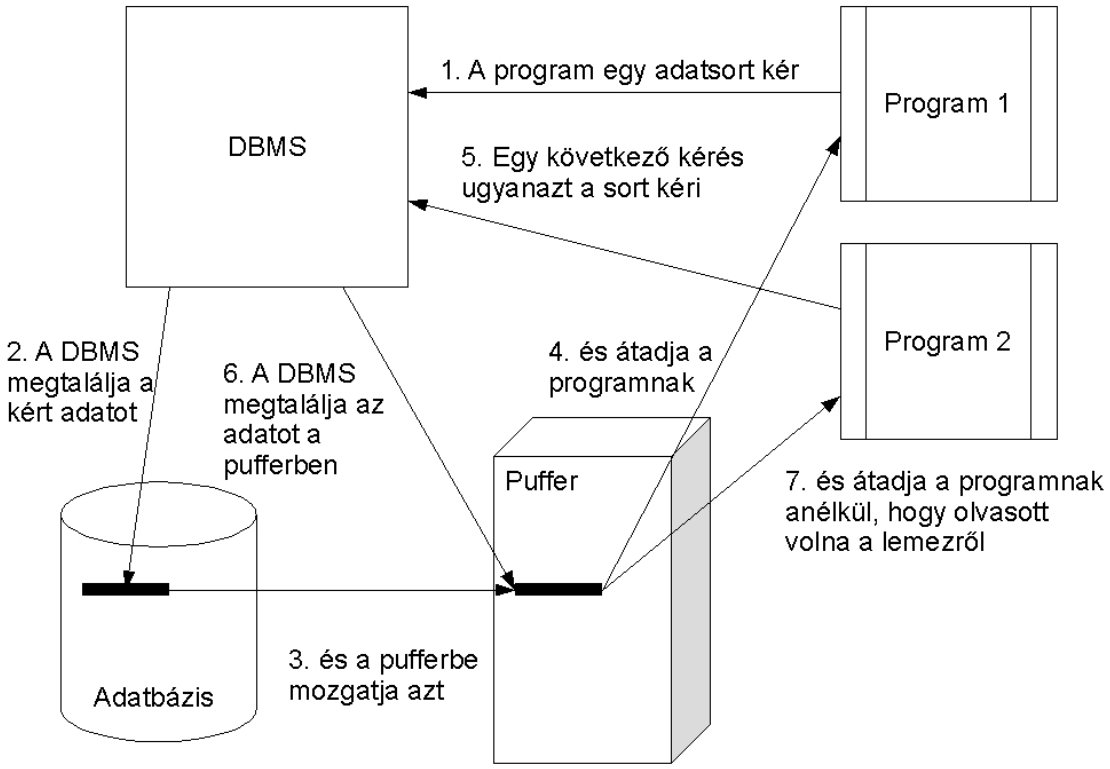
Az adatbázis-adminisztrátornak az adatbázis-kezelő rendszer minden tárolási követelményét meg kell ismernie, és ügyelni kell rá, hogy mindig legyen az állományoknak elegendő helye. Az adatbázis-kezelő rendszer a különböző állományokat (adat, napló, vezérlő, stb.) képes párhuzamosan olvasni és írni. Emiatt jó az, ha több adattároló eszközt használunk, még akkor is, ha a kapacitásukat nem használjuk ki. Megfelelő állományelhelyezéssel az adatbázis-kezelő rendszer sokkal hatékonyabban működik, mert a párhuzamos adatelérések hatékonyságát nem a lemez fizikai korlátai szabják meg.

Az adatbázis-kezelő rendszernek a lemezeken kívül szalagos egységre is szükségük van olyan feladatokhoz, mint adatbázismentések, naplómentések. Az adatbázismentéseket akkor fogjuk használni, ha az adatbázisban valamilyen hiba történik és helyreállítás szükséges. Ahhoz, hogy az adatbázisunkat bármilyen hiba esetén, bármikor konzisztens állapotba tudjuk helyreállítani, szükség van az adatbázisnapló archiválására is. Az adatbázisnapló archiválása akkor történhet meg, amikor az aktív naplóállomány megtelik. Ekkor naplóváltás történik, egy új naplóállományba kerülnek az adatbázis-módosítások bejegyzései, a régi naplóállományt pedig ekkor lehet archiválni. Az adatbázisnaplóról bővebben az Adat- és tároláskezelés című fejezetben, illetve a Rendszerteljesítmény című fejezetben olvashatunk.

Memóriakövetelmények

A relációs adatbázis-kezelő rendszerek adatbázisai és alkalmazásai imádják a memóriát. Egy adatbázis-kezelő rendszernek az alaptevékenységéhez szüksége van memóriára, ami azt jelenti,

hogy a legtöbb memóriát a belső feldolgozásra használja, azaz arra, hogy az adatbázis-kezelő rendszerhez tartozó memóriaterületeket karbantartsa, és a különböző belső feladatokat teljesítse.
Egy adatbázis-kezelő rendszernek jelentős mennyiségű memóriára van szüksége a gyorsítótárhoz (cache), hogy elkerülje a gyakori input-output műveleteket (I/O-t). A lemezes tárolóeszközökről mindig sokkal költségesebb és lassabb beolvasni az adatokat, szemben az adatok memóriabeli átmozgatásával. A 2-3. ábra mutatja, hogy az adatbázis-kezelő rendszer a fizikai input/output kérések csökkentésére hogyan használja az adatgyorsítót, vagy puffert (buffer pool vagy data cache). Az adatbázis-kezelő rendszer a beolvasott adatokat igyekszik a pufferben tartani, hogy ha később ugyanezekre az adatokra lesz szükség, akkor ne kelljen beolvasnia újra. Így csökkenti az input/output kérések számát. Általában minél nagyobb a puffer, annál tovább maradnak az adatok a memóriában és az adatbázis-folyamatok annál gyorsabbak lesznek.



2-4. ábra Memóriahasználat

Az adatok mellett az adatbázis-kezelő rendszer más struktúrákat is beolvas a memóriába, mint például programstruktúrákat. Ez például jelentheti az aktuálisan futó programok által használt, lefordított SQL utasításokat, vagy végrehajtási tervet. Az adatbázis-kezelő rendszer ezeknek a struktúráknak a memóriában való tárolásával az adatbázis-folyamatok teljesítményén javít, mert elkerüli a felesleges input/output kéréseket.
Az adatbázis-kezelő rendszer a memóriát zárolási kérések, rendezés, folyamatok optimalizálása, SQL feldolgozás, stb. támogatására is használja. A memóriáról még további ismereteket olvashatunk a Rendszerteljesítmény című fejezetben.
Ha az adatbázis-kezelő rendszernek az elegendőnél több memóriát biztosítunk, akkor már sokat tettünk az adatbázis-folyamatok optimalizálásáért és a lehetséges teljesítményproblémák minimalizálásáért.

Adatbázis-kezelő rendszer telepítése

Ha az előbb említett feltételeket biztosítani tudjuk az adatbázis-kezelő rendszer számára, és a tárolási és memóriakövetelményeknek megfelelően a telepítést megterveztük, akkor olvassuk tovább a telepítési útmutatót. A telepítés megkezdése előtt bizonyosodjunk meg róla, hogy megértettük az utasításokat, amelyek szerint a telepítés előkészítését el kell végezni, majd végezzük el őket. Tekintsük át, hogyan működik a telepítőprogram és kövessük az explicit utasításokat, amelyeket a telepítési útmutató ad.

Az adatbázis-kezelő rendszer konfigurálása

Rendszerparaméterek konfigurálásával válnak az adatbázis-kezelő rendszer funkciói és erőforrásai elérhetővé a felhasználók számára. Az adatbázis-kezelő rendszerek paramétereit különböző módokon lehet módosítani. A telepítéskor általában rádiógombok, menük és panelek segítségével tehetjük ezt meg.
Az adatbázis-kezelő rendszer a paraméterek módosítására biztosít valamilyen eszközt. A módosítás megtörténhet egy utasítás kiadásával, vagy egy állomány módosításával. Állomány módosítása esetén egy téves rendszerparaméter beállítás végzetes lehet az adatbázis-kezelő rendszer működésére.
A rendszerparaméterek segítségével lehet például az aktív adatbázisnaplók számát meghatározni, megadni az adatokhoz és a programokhoz használt memóriaterület, vagy adatbázis-kezelő rendszer eszközöket ki- és bekapcsolni.
Az adatbázis-adminisztrátornak az adatbázis-kezelő rendszer konfigurálásához ismernie kell az adatbázis-kezelő rendszer paramétereit. Ha hibázunk a rendszerparaméterek konfigurálásában, egy rosszul konfigurált adatbázis-környezetet kapunk, amely teljesítményproblémákat, adatintegritási problémákat vagy akár az adatbázis-kezelő rendszer működésének a hibáját is okozhatja.
A rendszerparaméterekről a Rendszerteljesítmény című fejezetben olvashatunk még.

A támogató infrastruktúraszoftverek kapcsolása az adatbázis-kezelő rendszerhez

Az adatbázis-kezelő rendszer telepítésének a része, hogy más, az adatbázis-kezelő rendszerrel kölcsönhatásban álló rendszerszoftvereket az adatbázis-kezelő rendszerhez kapcsoljunk. Ahhoz, hogy az adatbázis-kezelő rendszerrel együtt tudjanak dolgozni, konfigurálni kell az általános infrastruktúraszoftvereket, melyek az alábbiak: hálózati szoftverek, tranzakciófeldolgozás-monitorozó, üzenetsorba-állító szoftverek, programozási nyelvek, rendszerkezelő szoftverek, művelet- és feladatfelügyelő szoftverek, webszerverek és alkalmazásszerverek.
A támogató infrastruktúraszoftverek különböző követelményeket támasztanak, hogy az adatbázis-kezelő rendszerrel kapcsolatba kerüljenek. Általában a konfigurálás DLL állományok telepítését, új paraméterállományok létrehozását, vagy a támogató szoftver telepítőjének újrafuttatását jelenti.

A telepítés felülvizsgálata

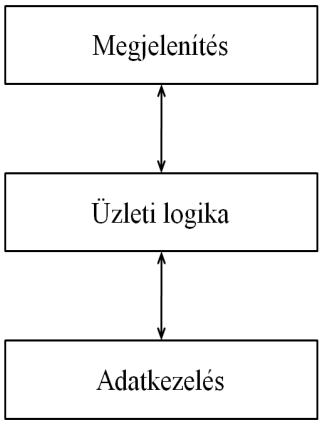
Az adatbázis-kezelő rendszer telepítése után teszteket kell futtatni, hogy megbizonyosodjunk, hogy az adatbázis-kezelő rendszer megfelelően lett telepítve és konfigurálva. A legtöbb adatbázis-kezelő rendszer szállítójának erre a célra van példaprogramja, de a megfelelő telepítést az adatbázis-kezelő rendszer szabványos interfészének a tesztelésével is elvégezhetjük. Majdnem minden adatbázis-kezelő rendszernek van egy szabványos interfésze, ez egy interaktív SQL interfész, ahol SQL utasításokat lehet futtatni.

Adatbázis-kapcsolódások

A telepített adatbázis-kezelő rendszerhez a felhasználóknak kapcsolódniuk kell. Előismereteink alapján elevenítsük fel az üzleti alkalmazásoknak a 3 logikai rétegét:

- a megjelenítési réteg tartalmazza a számítógép képernyőjén megjelenő információkhoz szükséges eszközöket. Általában magában foglal egy grafikus felhasználói felületet interaktív segítséggel, és más egyszerűen használható eszközökkel.
- az üzleti logika réteg szállítja a végfelhasználóknak az alkalmazáshoz szükséges magelemeket, amelyek az üzlet kezeléséhez szükséges információkat manipulálják. Ez az üzleti logika az üzleti stratégiák megvalósítására, üzleti tranzakciók vezérlésére és a cégpolitika erősítésére szolgáló metódusokat egyesíti.
- az adatkezelő réteg a strukturált adatok biztonságos módon történő gyors elérését szolgálja, az adatmódosításokat segíti elő, és megőrzi az adatintegritást.

Kliens-szerver architektúra alkalmazása esetén a szoftverrétegek megvalósulhatnak egy, kettő, három vagy több csomóponton. A 2-5. ábrán a három logikai réteg kapcsolatát láthatjuk.



2-5. ábra Az üzleti alkalmazások logikai rétegei

Az adatbázis-adminisztrátornak nem feladata a megjelenítést és az üzleti logikát megvalósító szoftverek adminisztrálása, de az adatkezelést ténylegesen végző adatbázis-kezelő rendszerhez való kapcsolódás megvalósítását támogatnia kell.

Az adatbázis-kezelő rendszer frissítése

Az élettel együtt jár a változás. A nagyobb adatbázis-kezelő rendszerek szállítóinak a termékei elég gyorsan változnak. Egy adatbázis-kezelő rendszer szoftverének általában 12-18 havonta jelenik meg új verziója. Hibajavításokat és karbantartó módosításokat azonban a szállító cég folyamatosan küld. Az adatbázis-adminisztrátornak a feladata az adatbázis-kezelő rendszer szoftverének a naprakészen tartása.

Egy adatbázis-kezelő rendszer verziófrissítése megfelel egy új telepítés speciális esetének. Minden termékfrissítés esetén biztosítani kell a megfelelő erőforrásokat, felül kell vizsgálni a rendszerparamétereket, és biztosítani kell, hogy a támogató szoftverek megfelelően kapcsolódjanak a frissítés után is. A létező alkalmazásokat és felhasználókat is tekintetbe kell venni, azaz ügyeljünk rá, hogy az új telepítés a lehető legkisebb megszakítással járjon. Emiatt a frissítés egy kényes és bonyolult feladat az adatbázis-adminisztrátor számára.

Egy összetett, heterogén, elosztott adatbázis-környezetben szükség van egy következetes frissítési stratégiára.

A frissítéssel a következő előnyökhöz juthatunk:

- A fejlesztőknek az új eszközök hasznosak lehetnek és van olyan funkció, amit csak az új

verzió nyújt. Fejlesztésnél az új eszköz csökkentheti a programozási időt és így költséghatékonyabb lehet.

- A fizetős alkalmazásokhoz az alkalmazás szállítója megkövetelheti az adatbázis-kezelő rendszer egy bizonyos verzióját az alkalmazás bizonyos verzióihoz.
- Az adatbázis-kezelő rendszer új verziójával általában javul a teljesítmény és az elérhetőség, amely a meglévő alkalmazásokat optimalizálja.
- Az adatbázis-kezelő rendszer szállítója az új verziók esetén gyakran jobb támogatást biztosít és a problémákra gyorsabban válaszol. A szállítók nem szeretik, ha az új és reklámozott termékük a hibák miatt rossz reklámot kap.

Az frissítés veszélyei:

- Egy frissítés általában az üzleti műveletek megszakítását jelenti. Amíg az adatbázist frissítik, az adatbázis nem érhető el. Ha a frissítés a normál üzleti időszak alatt történik, akkor az leállást eredményezhet és így elveszett üzleteket. A klaszterezett adatbázis-megvalósítások segítségével az adatbázis elérhető marad a frissítés alatt is, bár a frissítési folyamat lassabb lesz az egyes klasztercsomópontok új verzióra való migrálása miatt.
- Az előző verzióban támogatott eszköz az új verzióból eltűnhet, ami alkalmazáshibát okozhat.
- A frissítés költsége jelentős gátja lehet az adatbázis-kezelő rendszer új verziójára történő migrációnak. Az adatbázis-kezelő rendszer ára 10-25%-kal nőhet. A frissítési költség a tervezés, telepítés, tesztelés, fejlesztés költségében is jelentkezik, nem csak az adatbázis-kezelő rendszerében. A frissítéssel olyan költségek is felmerülhetnek, mint új erőforrások, azaz memória- és tárbővítés, processzorbeli teljesítménynövelés.
- Az adatbázis-kezelő rendszerek szállítói az új verziókban teljesítménynövelést hajtanak végre. Ha az SQL optimalizálási technika változik, lehet, hogy, az adatbázis-kezelő rendszer új verziója rosszabb teljesítményű SQL elérést generál, mint előtte. Az adatbázis-adminisztrátornak szigorú teszteket kell végrehajtania, hogy biztos legyen benne, hogy az új elérési út segíti, és nem hátráltatja az alkalmazás teljesítményt. Ha a teljesítmény csökken, lehet, hogy alkalmazáskódot kell cserélni, ami egy nagyon költséges és időfogyasztó erőfeszítés.
- A támogató szoftvertermékek hiányosak lehetnek az új adatbázis-kezelő rendszer verziókhoz, amikor azt kiadják.

Összefoglalás

Egy cégnél a telepített adatbázis-kezelő rendszerek területén stratégiát kell kialakítani. Ehhez megnéztük, hogy milyen szempontokat kell figyelembe venni az adatbázis-kezelő rendszerek kiválasztásában. Felsoroltuk napjaink említésre méltó adatbázis-kezelő rendszereit, és azok szállítóit. Továbbá részleteztük az adatbázis-kezelő rendszerek architektúráit, és figyelembe vettük az elérhetőség növelése érdekében a klaszterezés lehetőségét.

A fejezet további részében a telepítés lépéseit vizsgáltuk meg. Részleteztük a hardver- és operációsrendszer-követelményeket, a tárolási követelményeket, a memóriakövetelményeket. Megnéztük, hogy hogyan lehet konfigurálni az adatbázis-kezelő rendszert, milyen támogató infrastruktúraszoftvereket kell az adatbázis-kezelő rendszerhez kapcsolni. A telepítés után a kész adatbázis-kezelő rendszert meg kell nézni, hogy jól működik-e. Az adatbázis-kapcsolódásokhoz felelevenítettük a kliens-szerver architektúrát és az alkalmazások logikai rétegeit.

A fejezet végén az adatbázis-kezelő rendszerek frissítésének gyakoriságáról, előnyeiről és veszélyeiről olvashattunk.

Ellenőrző kérdések

1. Milyen szempontok alapján választunk adatbázis-kezelő rendszert?
2. Soroljuk fel a napjainkban ismert nagy adatbázis-kezelő rendszert szállító cégeket és a termékeiket!
3. Milyen adatbáziskezelőrendszer-architektúrák vannak, és melyiknek mi a jellemzője?
4. Mit jelent a klaszterezés (fürtözés)? Milyen két domináns architektúrát említettünk? Melyiknek mi a jellemzője?
5. Az adatbázis-adminisztrátornak milyen feladatai vannak az adatbázis-kezelő rendszer telepítése előtt és után?
6. Amikor az adatbázis-adminisztrátor az adatbázis-kezelő rendszerhez szükséges tárterületet számolja, az adatokon és az indexeken kívül még milyen más struktúrákat kell figyelembe vennie?
7. Miért olyan fontos az adatbázis-kezelő rendszer számára a memória?
8. Mik a rendszerparaméterek? Mire szolgálnak?
9. Hogyan lehet az adatbázis-kezelő rendszer telepítését felülvizsgálni?
10. Az alkalmazásoknak milyen szoftverrétegei vannak?
11. Milyen előnyei és veszélyei vannak az adatbázis-kezelő rendszer frissítésének? Hogyan kell elvégezni a frissítést?

3. fejezet – Metaadatok kezelése

Az adatbázis-adminisztrátornak nem csak az adatokkal kell dolgoznia, hanem az adatelemek definícióival is. Az adatok leírásának, struktúrájának, korlátozásainak és definíciójának megértése nélkül az adatok helytelenül lesznek értelmezve vagy rosszul lesznek használva. A nem helyesen definiált adatok integritási problémákat okozhatnak.

Mi a metaadat?

Ahhoz, hogy a felhasználói adat hasznos információvá váljon, el kell helyeznünk egy adott környezetbe. Az adatról szóló információt metaadatnak nevezzük. A metaadat legegyszerűbb definíciója: adat az adatról. Pontosabban, a metaadat írja le az adatot, azaz olyan információkkal szolgál, mint típus, hosszúság, szöveges leírás és más jellemzők. A metaadat válaszol az adat felhasználóinak a ki, mit, mikor, hol, miért és hogyan kérdésekre.

Adat és információ

Az adat egy elem vagy esemény által reprezentált tény, amely nincs kapcsolatban más tényekkel, és nincs meghatározva a környezete. Példák az adatra: 27, Jan vagy 010110. További részletek nélkül semmit nem tudunk erről a három adatról. Tekintsük a következőket:

- a 27-es szám tízes vagy 8 számrendszerbeli?
- A 27-es 10-es számrendszerben mit jelent? Egy kort, pénzösszeget, IQ-t, cipőméretet, vagy valami mást?
- Mit jelent a Jan? Egy női név? Vagy férfi név? Vagy az év első hónapja? Vagy valami más?
- És a 010110 mit jelent? Bináris szám? Vagy dátum? Talán 1910. január 1.? Vagy 2010. január 1.? Vagy valami más?

A környezet hiánya miatt ezek mind csak adatok. Az információhoz szükség van a környezetre, amely meghatározza az adatok közötti kapcsolatot és más információkat. Az adat, ha metaadat környezetbe kerül, információvá válik. A kapcsolatok információkat reprezentálhatnak, de nem alkotnak információkat, amíg meg nem értjük őket.
Hogy az adat ne csak egyszerű adat legyen, szükségünk van metaadatokra. Metaadatok nélkül nincs azonosítható jelentés, csak számjegyek, betűk és bitek gyűjteménye. A metaadat az adatoknak formát ad és hasznos információvá teszi őket.

Metaadat-stratégia

Az adatbázis-adminisztrátor, vagy ha létezik, akkor az adatadminisztrátor feladata, hogy a cégnél egy metaadat-stratégiát fejlesszen ki. A metaadat-stratégia határozza meg, hogy a cégnél a metaadatokat hogyan gyűjtik, kezelik és érik el. A metaadat-stratégia a következőket tartalmazza:

- vezérelveket, amely szerint a cégben a metaadatokat használják
- módszereket az adatok tulajdonosainak és gondnokainak definiálására és azonosítására
- az összegyűjtendő metaadattípusok megnevezéseit
- az egyes metaadattípusok céljának leírását – vagyis azt, hogy az egyes metaadatokra a cégnek miért van szüksége
- a metaadatok tárolására és gyűjtésére szolgáló eljárásmódokat
- módszereket a metaadat elérésére
- olyan vezérelveket, amelyek kikényszerítik a gondnoksági szabályokat és a metaadatok eléréséhez a biztonságot
- a külső és belső metaadat-források megnevezését

- mértékeket a minőség méréséhez és a metaadat használhatóságához

A metaadatok összegyűjtésével és kezelésével a cégünk saját adatairól olyan lényeges tényeket tudhat meg, amely a rendszereket használhatóbbá, az adatbázisokat pedig hasznosabbá teszik.
Az adatbázis-adminisztrátornak részt kell vennie a metaadat-stratégiát fejlesztő csoport munkájában, de az adatadminisztrátornak (ha létezik ilyen a cégnél) kell lennie a vezetőnek.

Metaadattípusok

Minden metaadat adatot ír le, de többféle metaadattípus létezik, amely többféle forrásból származhat. Alapszinten kétféle metaadattípusról beszélhetünk: technológiai és üzleti.
A technológiai metaadat az adatokat technikai oldalról írja le. Az adatok tárolásáról és az adatok számítógépes rendszerben való kezelésről ad információt.
Az üzleti metaadatok azt írják le, hogy az adatokat az üzlet hogyan használja. Ahhoz szükségesek, hogy az adat értékes legyen a cég számára.
Technológiai metaadat például az, hogy egy tábla SZEMELYISZAM nevű oszlopa egy 11 hosszú karaktersorozat tárolására képes, és csak számjegyeket tartalmaz. Az is technológiai még, hogy tudjuk, hogy az első számjegy csak 1,2,3,4-es lehet. Természetesen az üzleti felhasználónak is szüksége van erre a tudásra.
Üzleti metaadat például az, hogy a SZEMELYISZAM valójában egy személyi számot takar, hogy egy személyi szám csak egy adott személyhez tartozik, és az is, hogy kiolvashatjuk belőle, hogy mikor született az adott személy. Az adatbázis-adminisztrátor számára az üzleti metaadat is fontos, hogy az adatbázist megfelelően és hatékonyan hozza létre.

Rendszerkatalógus

Ha az adatbázis-adminisztrátornak metaadatokra van szüksége, akkor számára egy jó forrás az adatbázis-kezelő rendszer. A rendszerkatalógus adatbázis-objektumról szóló információk tárol, létfontosságú tár a technológiai metaadatokhoz. Az adatbázis-adminisztrátorok és a fejlesztők rendszeresen használják az adatbázis-kezelő rendszernek a rendszerkatalógusában lévő metaadatait, hogy az adatbázisban tárolt objektumokat és a bennük lévő adatokat értelemszerűen tudják használni. Az adatbázis-kezelő rendszer vagy táblákat és nézeteket, vagy tárolt eljárásokat biztosít ahhoz, hogy az adatbázis használói a rendszerkatalógusból metaadatokat nyerhessenek ki. A legtöbb adatbázis-kezelő rendszer a következő metaadatokat tárolja a rendszerkatalógusban:

- minden adatbázis, tábla, oszlop, index, nézet, tárolt eljárás, trigger stb. neve
- a táblákhoz az elsődleges kulcsok, és a külső kulcsok
- nézetek definíciói
- adattípusok, hosszúságok és megszorítások a táblák oszlopaihoz
- a fizikai állományok nevei, amelyekben az adatbázis adatai vannak és információk az állomány tárolásáról
- jogosultság és biztonsági információk
- más egyéb adatbázis szervezési információk

Az adatbázis-kezelő rendszer rendszerkatalógusa különösen hatékony forrása a metaadatoknak, mert rendelkezik három tulajdonsággal: aktív, integrált és védett. A rendszerkatalógus

- *aktív*, mert a metaadatok automatikusan felépülnek és karbantartódnak, ahogy az adatbázis-objektumok létrejönnek és módosulnak. Ha az adatbázis-adminisztrátor adatbázis-objektumokat hoz létre, az adatbázis-kezelő rendszer automatikusan összegyűjti és létrehozza a metaadatokat a rendszerkatalógusban.
- *integrált,* amely együtt jár az aktivitással, mert az adatbázis-kezelő rendszer a technológiai adatokat a rendszerkatalógusban pontosan és naprakészen tartja. Azaz nincs ellentmondás az adatbázis és a rendszerkatalógus között.

- *védett*, ami azt jelenti, hogy a normál adatbázis-kezelő rendszer működés az egyetlen mechanizmus amivel a rendszerkatalógushoz hozzáférhetünk. Természetesen ez adatbázis-kezelő rendszerről adatbázis-kezelő rendszerre változhat. Néhány adatbázis-kezelő rendszer egy opciót biztosít, amely lehetővé teszi a rendszerkatalógus módosítását, de az ilyen műveletek csak vész esetén és csak az adatbázis-kezelő rendszer technikai támogató személyzetének felügyelete alatt végezhetők.

Sok metaadat megtalálható az adatbázis-kezelő rendszer rendszerkatalógusában, de ezek a metaadatok általában nem elegendőek az adatok teljes leírásához. Például az adatbázis-objektumok üzleti leírása, dokumentációja általában nem található meg az adatbázis-kezelő rendszer rendszerkatalógusában. A következő metaadatok hasznosak, de a rendszerkatalógusban nem találhatóak meg:

- metaadatok az adatbázis-környezethez tartozó, de nem adatbázisbeli struktúrákhoz
- módosítási információk, hogy mikor és ki módosította utoljára az adatbázis-objektumait
- információk az adatbázistáblákhoz, például mely programok használják azokat
- információk a kötegelt feladatokról és a tranzakciókról
- az adatmodellek leírása, többek közt a logikai adatbázisterv, és annak a leírása, hogy ez hogyan rendelődik a fizikai adatbázis-megvalósításhoz
- adattárház és ETL (extract, transform, load) metaadatok
- az adatok üzleti tulajdonosait és gondnokait leíró metaadatok

Természetesen ez a lista nem teljes. Számtalan különböző metaadattípus létezik, amelyet kezelni lehet. Minél több metaadat áll az üzleti felhasználók rendelkezésére, annál több üzleti értéket lesznek képesek kinyerni az információs rendszerekből.

Repository

A repository napjaink informatikai megnevezése szerint általános tárolót jelent. A jegyzetünk szerint a repository-ban az adatbázis-környezethez tartozó metaadatok kerülnek tárolásra. Ezért tekinthetjük az rendszerkatalógust is egyfajta repository-nak. De a repository ettől bővebb tárolót jelent. A repository kerülhet az adatbázisba, vagy egy vagy több külön állományba. Kezelheti egy metaadatok kezelésére szolgáló szoftver, vagy lehet csak egyszerű szöveges állomány.
A metaadatok összegyűjtésével a repository információkat tárol a cég adatvagyonáról. Ha megfelelően van megvalósítva, akkor a cégnek minden metaadatát tárolhatja.
Általában a jó repository-tól és az azt kezelő szoftvertől a következő funkciók várhatóak el:

- információt tárol az adatokról, a folyamatokról és a környezetekről
- támogatja azt, hogy ugyanazt az adatot többféle nézőpontból vizsgáljuk. Például nyomon követhetünk egy attribútumot a koncepcionális, a logikai vagy a fizikai adatmodellben.
- nagyon alapos dokumentációt tárol, amely mellett még eljárásrészletek vagy kezelő riportok is megjelenhetnek
- támogatja a különböző adatmodellek létrehozását és adminisztrációját. Integrált ETL, adatmodellező és CASE eszközöket (Computer Aided System Engineering, automatikus tervező eszköz) tartalmazhat.
- támogatja a verziókezelést és felügyeli a változásokat. A verziókezelés segít szinkronizálni az alkalmazásfejlesztést és növeli a rugalmasságot.
- érvényre juttatja a névkonvenciókat
- összegyűjti és elemzi a több forrásból származó metaadatokat
- legenerálja az adatelem-definíciókat.

Ha az adatbázis-fejlesztéshez repository-t, és kezelő szoftvert kell választanunk, akkor olyat válasszunk, amely a következő tulajdonságokkal is rendelkezik:

- A repository-kezelő szoftver az általa használt adattárakat az adatbázisbeli táblákban tudja

tárolni. Így az integritását, a biztonságát, a mentését, és az esetleges helyreállítását az adatbázis-kezelő rendszer eszközeivel meg lehet oldani. Ez nem jelent igazán sok többlet feladatot az adatbázis-adminisztrátor számára.

- A repository-kezelő szoftver több típusú adatbázis-kezelő rendszert is tud egyszerre támogatni. Ha a cégünknél több típusú adatbázis-kezelő rendszerrel dolgozunk, akkor is elegendő egy repository.
- A repository-kezelő szoftver képes közvetlenül olvasni a rendszerkatalógust. Ez biztosítja, hogy a repository-kezelő szoftvernek aktuális információi legyenek az adatbázis-objektumokról.
- A repository rendelkezik interfésszel azokhoz a modellező és tervező eszközökhöz, amelyeket az adatbázis-objektumok generálásához használunk.

A repository előnyei

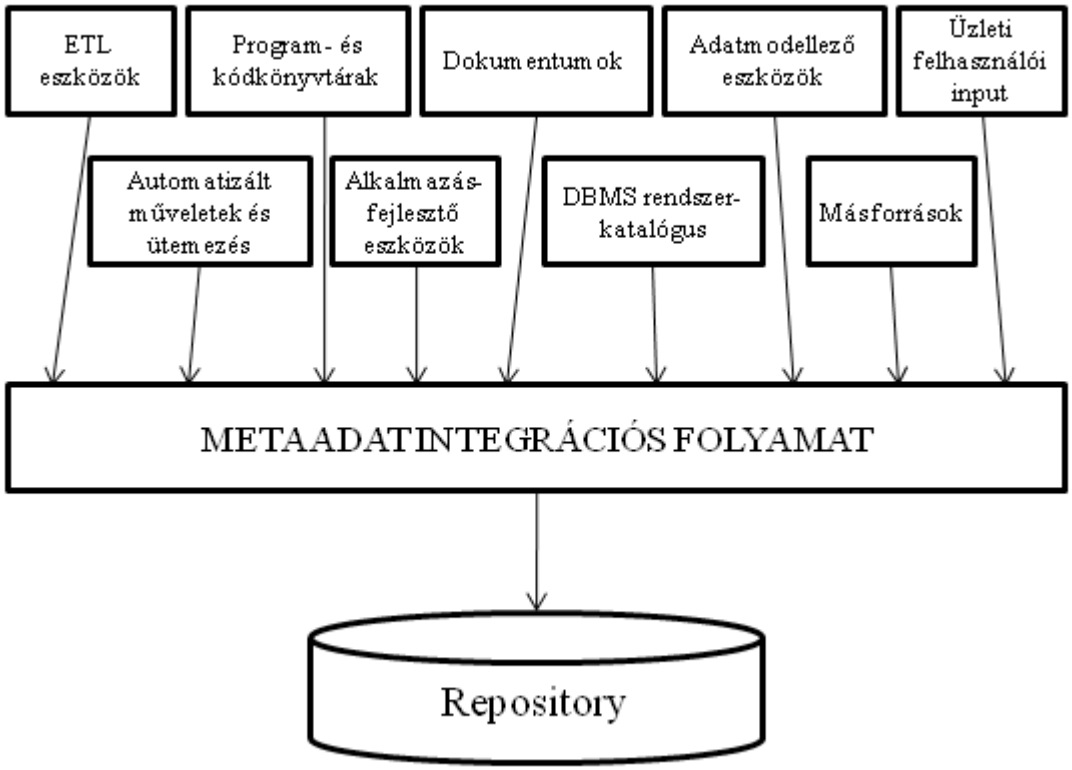
A repository-technológiák a cégek számára nagyon sok előnyt biztosítanak. A repository-ban lévő metaadatok egyesített információkat tartalmaznak a cég különböző rendszereiről. Ezeknek az egyesített információknak a segítségével a fejlesztők könnyebben megérthetik, hogy az adott rendszerek mire és hogyan használják az adatokat. Az egyesítés segítségével olyan kapcsolatokat lehet felfedezni az adatok között, amelyek még esetleg nem ismertek a cégen belül. A kapcsolatok felfedezésével a cég új üzleti folyamatokat vezethet be, vagy a meglévőeket hatékonyabbá teheti.
A repository-nak a másik nagy előnye, hogy az adatelemek és az üzleti szabályok dokumentációjában konzisztenciát biztosít.
A repository a gyorsan változó környezeteket is támogatni tudja. A repository-ban lévő metaadatok alapján gyors riportokat lehet készíteni, amelyekből gyorsan meg lehet határozni, hogy az egyes területeken történő változások milyen hatással lesznek más területekre.
Az újrafelhasználhatósággal időt, és erőforrást spórolhatunk. A repository megkönnyíti az alkalmazáskomponensek újrafelhasználását. A repository könnyedén felismerteti azt az értéket, hogy az örökölt rendszerek programdokumentációját és a működési metaadatokat hasznosítani lehet az új alkalmazásfejlesztésnél.
A repository felbecsülhetetlenek az adattárház bevezetésénél. A repository-ban tárolt információk segítenek a többféle adatforrásból származó adatokat az adattárházba migrálni.

A repository nehézségei

Az egyik legnagyobb nehézség a repository-technológia megvalósításában és használatában, a repository naprakészen tartása. A repository több forrásból származó adatokat tartalmazhat. A forrásadatok közül bármelyik bármikor változhat. Ha a forrásadatok változnak, akkor a hozzá tartozó repository-beli adatokat is változtatni kell.
A repository-ban lévő metaadatok a cég több területéről származhatnak:

- a programfejlesztési eszközökből, az alkalmazásprogramokból és a kódkönyvtárakból az alkalmazáskomponensek metaadatai,
- üzleti felhasználóktól üzleti metaadatok,
- az adatmodellező eszközökből adatmodellező metaadatok,
- az adatbázis-kezelő rendszer rendszerkatalógusából adatbázis-metaadatok,
- adattárház eszközből ETL metaadatok,
- automatizált műveletekből és feladatütemező eszközökből működési metaadatok
- egyéb területekről, mint például a lekérdező eszközökből adathasználati metaadatok

A 3-1. ábra a különböző területekről származó metaadatok egy repository-ban való tárolását mutatja be.



3-1. ábra Repository és a forrásrendszerek

A fenti, nagyon heterogén területekről származó információkat kell összegyűjteni és elemezni, majd a repository-ban tárolni. Az egyesítési folyamatnak figyelembe kell vennie, hogy az egyes metaadatforrások tartalma gyakran változik. Ha változik a metaadat a forráshelyen, akkor a repository-t szinkronizálni kell, át kell vezetni ezt a változást.
Sok cégnek nincs repository-ja, vagy nem valósítja meg megfelelően az egyesítést és a szinkronizálást, és így bár van repository-ja, de nem tudja használt venni. Ha a repository metaadatai nem pontosak, elavultak vagy nem léteznek, a repository elveszti az értékét. A hiba nem feltétlenül a repository-technológián alapszik, hanem a cégen, amely nem ügyel a repository-ban lévő metaadatok naprakészen tartására. Természetesen ez költséggel és elkötelezettséggel jár. A repository megvalósításával, és megfelelő kihasználásával azonban ezek az erőfeszítések maximálisan megtérülnek.
Célszerű már tervezéskor létrehozni a repository-t. Ekkor már abból lehet automatikusan kódokat generálni, illetve a kódok változása esetén automatikus visszavezetést alkalmazni.

Összefoglalás

A fejezetben definiáltuk a metaadatot, példák segítségével rávilágítottunk az adat és az információ közötti különbségre. Megnéztük, hogy mit tartalmaz a metaadat-stratégia. Különbséget tettünk üzleti és technológiai metaadat között.
Áttekintettük, hogy a rendszerkatalógus mit tartalmaz és mit nem tartalmaz azon metaadatokból, amire egyébként szüksége van. A rendszerkatalógus három legfontosabb tulajdonságával ismerkedtünk meg: aktív, integrált és védett.
Definiáltuk a repository-t, megnéztük, hogy általában az adatbázishoz kapcsolódó repository-k mit tudnak, és mi az ami még ezen felül elvárható tőlük. Felsoroltuk, hogy milyen forrásokból kerülnek

be a repository-ba a metaadatok. Áttekintettük a cégek tekintetéből a repository előnyeit, illetve a repository-val kapcsolatos nehézségeket.

Ellenőrző kérdések

1. Mi a metaadat?
2. Mikor válik az adat információvá?
3. Mit tartalmaz a metaadat-stratégia?
4. Mi a különbség a technológiai és az üzleti metaadat között?
5. Mit tartalmaz a rendszerkatalógus?
6. Soroljunk fel olyan metaadatokat, melyek nem találhatóak meg a rendszerkatalógusban!
7. Milyen tulajdonságai vannak a rendszerkatalógusnak? Melyik mit jelent?
8. Mi a repository?
9. Általában mit tud a repository? Mit várhatunk még ezen felül el tőle?
10. Milyen előnyei vannak a repository-nak?
11. Milyen nehézségekkel kell szembenézni a repository-val kapcsolatban?
12. Milyen forrásokból származnak a repository-k metaadatai?

4. fejezet – Adat- és tároláskezelés

A relációs adatok a felhasználók számára logikailag táblákba vannak szervezve. A táblákban lévő adatokat azonban nem elegendő csak a memóriában tárolni, hiszen onnan az adatok elvesznek, ha a számítógépet kikapcsoljuk. Azaz a táblabeli adatoknak valamilyen módon háttértárakra kell kerülniük. A háttértárak lehetnek merevlemezek, lemez alrendszerek, hordozható tárolók, optikai tárolók vagy szalagos eszközök. A háttértárakon az adatok állományok formájában jelennek meg.
Az adatbázis-adminisztrátor egyik kulcsfontosságú feladata az adatbázis adatainak a tárolását megtervezni. Ehhez az adatbázis-adminisztrátornak ismernie kell, hogy az adattárolás milyen fizikai mechanizmusokkal valósul meg.

A tárolási megoldás kiválasztása

Az adatbázis-kezelő rendszerek nem minden tárolási technológiával tudnak együtt dolgozni. Az adatbázis-adminisztrátornak sok tárolási technológiát kell kipróbálnia, hogy meghatározza, hogy a cég adatbázis-kezelő rendszere melyekkel működik a legjobban. A megfelelő tárolási technológia kiválasztásához vegyük figyelembe a teljesítményt, a megbízhatóságot, használhatóságot, és az árat.
Az adatbázis-kezelő rendszerek esetén a legalapvetőbb tárolási technológia a merevlemezen történő tárolás. A merevlemez hátránya, hogy a lemezmeghajtó mechanikai részei miatt sebezhetőbb, mint a számítógép más részei. Ennek a hátránynak a kiküszöbölésére azonban használhatunk RAID technológiát.
Az adatbázis-kezelő rendszer teljesítménye függ az input/output műveletek sebességétől. Az adatbázis-kezelő rendszer minél gyorsabban tud egy input/output műveletet elvégezni, az adatbázis-alkalmazás annál gyorsabban fut. Sokkal lassabban lehet az adatot egy háttértárolóról beolvasni, mint a memóriából. Néhány új háttértároló rendszerbe emiatt gyorsító mechanizmust építettek, amely úgy működik, hogy az adatok automatikusan beolvasásra kerülnek a memóriába. Ezzel csökken a klasszikus input/output műveletekhez rendelt várakozási idő.
Az adatbázis-kezelésben a tárolás központi kérdés, és az idő múlásával még többet kell vele foglalkozni, mert egyre több és több adatot szeretnénk tárolni. És az adatokat nem csak ideiglenesen tárolja a cég, hanem hosszú időn keresztül megőrzi és felhasználja. Ehhez azért manapság is már egyre több heterogén, több terabájtos tárolási rendszer érhető el a piacon. A növekvő tárak kezelésére különböző tárolási technológiák léteznek, mint a SAN (Storage Area Network) és a NAS (Network-attached Storage). A SAN egy speciális hálózat tárolási eszközök összekapcsolására. Segítségével a szerverek ezt a háttértárhálózatot egy normál háttértárolónak látják, és a megszokott módon tudják használni. (Közvetlenül a géphez kapcsolható, nagyobb rack, amelybe több winchestert teszünk. A szerver egy logikai háttértárként látja a háttértároló-rendszert.) A NAS egy olyan tároló eszköz, mely hálózaton keresztül biztosít tárhelyet a szervereknek (egy olyan eszköz, mint a rack, amibe a winchestert tesszük, csak ezt a hálózatban switch-hez kell kapcsolni.)
Az adatbázis-adminisztrátor egyik nagy feladata a fizikai adatbázisterv elkészítése. A fizikai adatbázis-tervezés egyik fontos része, hogy az adatbázis-adminisztrátornak hatékony tervet kell készítenie az adatok megfelelő tárolásához és a háttértároló területének a helykihasználásához. A következő célokat érdemes szem előtt tartani a tárolási rendszer kialakításakor:

- legfontosabb az adatvesztés megelőzése,
- biztosítani kell, hogy a megfelelő tárolókapacitás elérhető legyen és a tárolási megoldás könnyen skálázható legyen, ha a tárolási igény nő,
- olyan megoldást kell választani, amely az adatok gyors elérését biztosítja a szolgáltatás minimális megszakításával vagy megszakítás nélkül,
- olyan tárolási megoldást kell választani, amely hibatűrő és hiba esetén gyorsan javítható,
- olyan tárolási megoldást kell választani, ahol lemezeket cserélhetünk és bővíthetünk leállás

nélkül,

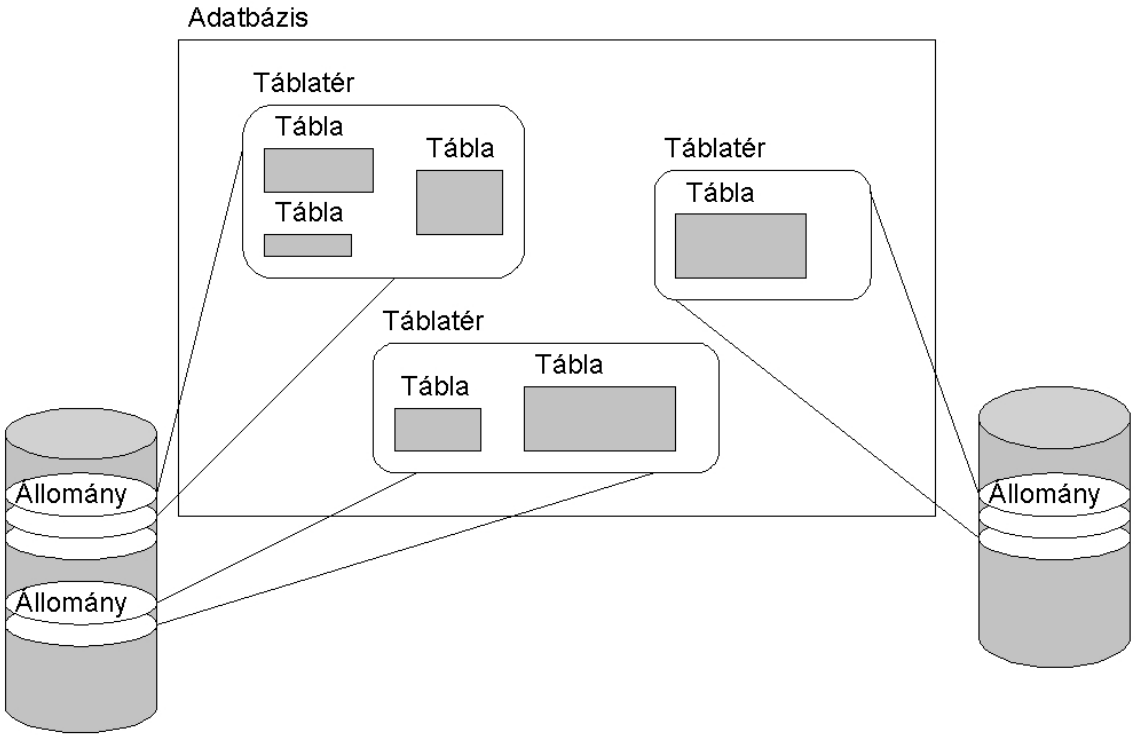
- olyan költséghatékony tárolási megoldást kell találni, amelyet a cég megengedhet magának.

Helykezelés

Sok tárolási kérdésben kell az adatbázis-adminisztrátornak döntenie, mielőtt egy adatbázist létrehoz. Az egyik legfontosabb kérdés, hogy mennyi helyet ad az adatbázisnak. Ennek becsléséhez szükséges azt tudnia, hogy mennyi adat kerül az adatbázisba, illetve azt, hogy az adatbázis-kezelő rendszernek ezen felül még milyen és mennyi járulékos információt szükséges tárolnia. Ehhez az adatbázis-adminisztrátornak ismernie kell az adatbázis-kezelő rendszer pontos működését.

Táblaterek

A relációs adatokat a felhasználók logikailag táblákba szervezetten tudják kezelni. Azonban ezeknek az adatoknak állományokban kell megjelenniük a háttértárolón. A fizikai tervezéskor az adatbázis-adminisztrátor a táblákhoz egy olyan logikai struktúrát rendel, amely azt fogja meghatározni, hogy a tábla adatai milyen fizikai állományokban lesznek tárolva. Ezeket a logikai struktúrákat általában táblatereknek hívjuk, és ezek az adatbázisnak objektumai. A 4-1. ábra azt mutatja, hogy egy adatbázis egy vagy több táblateret foglal magában és az egyes táblaterek egy vagy több táblát tartalmaznak. Az adatbázis-kezelő rendszertől függ az, hogy egy tábla adatai több táblatérbe is kerülhetnek-e. Általában az adatbázis-adminisztrátor egy táblát egy táblatérbe helyez el. Az adatbázis-adminisztrátor dönti el, hogy a táblákat hogyan rendeli a táblaterekhez, amely függ az adatok használatától, a táblatér típusától, és az adatbázis-kezelő rendszer tulajdonságaitól.



4-1. ábra Állományok, táblaterek, és a táblák kapcsolata

A táblaterekhez nem csak a táblákat kell hozzárendelni, hanem minden olyan adatbázis-objektumot,

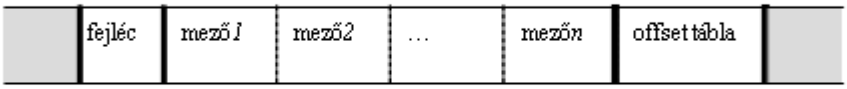
amelynek tárigénye van. Az adatbázis-objektumok a táblatereken keresztül vannak az állományokhoz rendelve, azaz a táblaterekkel határozhatjuk meg, hogy melyik adatbázis-objektum milyen lemezen és hol van tárolva. A táblán kívül az index az a másik nagyon fontos adatbázis-objektum, amely sok adatot tárol és táblatérhez kell rendelni.
A táblatér az adatbázisban ugyancsak egy adatbázis-objektumként jelenik meg. Az adatbázis-adminisztrátor feladata hozzárendelni a szükséges operációs rendszerbeli vagy állomány-rendszerbeli állományokat, amelyekhez az adatbázis-kezelő rendszer biztosítja a szükséges utasításokat. Egy táblatérhez lehet egy vagy több állományt is rendelni. Kezdetben általában az adatbázis-adminisztrátor egy állományt rendel hozzá, de az adatbázis növekedésével a tárigény kielégítésére új állományokat lehet hozzá rendelni.
A táblatérhez meg lehet adni a tárolásra vonatkozó paramétereket is, mint például azt, hogy a táblatérnek mekkorák legyenek az adatlapjai, vagy esetleg a mérete dinamikusan változhat-e.

Adatlap – adatrekord

A lemez tárolási egysége az operációs rendszerbeli blokk. A blokkok mérete fix. A lemezről az adatokat az adatbázis-kezelő rendszer a memóriába olvassa fel. Ennek a beolvasásnak az egysége az adatlap, amelynek a mérete a lemezblokk méretével egyezik meg, vagy annak az egész számú többszöröse. Vagyis az adatbázis-kezelő rendszer egyszerre egy adatlapnyi adatot tud a memóriába beolvasni.
A táblaterekhez rendelt állományok felépítésének alapja az adatlap. Azaz az adatállományok szerkezeti egységeként ugyancsak az adatlapot tekinthetjük.
Az adatbázis tábláiban lévő adatok azonban sorokba és oszlopokba vannak szervezve. Kérdés az, hogy hogyan kerülnek a tábla adatai az adatlapokra, milyen formában. A tábla egy-egy sorát az adatbázis-kezelő rendszer adatrekordba szervezi. A rekord mezőiben az adatok lehetnek fix vagy változó hosszúságúak, a tábla oszloptípusainak megfelelően. Ahhoz, hogy tudjuk, hogy az adott adatrekord melyik táblának az adatait vagy milyen felépítésű rekordot tárol, szükség van a rekord szerkezetét leíró információkra is. Ez azért is szükséges, mert az egy táblában lévő adatrekordok hossza is különböző lehet. Mégpedig úgy, hogy változó hosszúságú vagy opcionális mezőket tartalmaznak.
Az adatrekord pontos felépítése adatbázis-kezelő rendszerről adatbázis-kezelő rendszerre változhat. Azonban általánosan elmondhatjuk, hogy az adatrekord a következő részekből áll:

- *fejléc*: Egy rekord általában néhány bájtnyi adminisztrációs információval kezdődik, amely a rekord adattartalmának szerkezetét és elrendezését határozza meg. Ez tartalmazhat hosszúságot, információt a változó hosszúságú adatokról és más ellenőrző szerkezetet.
- *adatok*: A tábla egy sorának aktuális adattartalmát tartalmazzák az oszlopdefiníció sorrendjében. Az adatbázis-kezelő rendszertől függően a változó és a fix hosszúságú oszlopok elkülönítve lehetnek.
- *opcionális offset tábla*: a rekord tartalmazhat mutatókat, hogy a rekordban tárolt változó hosszúságú mezőket kezelje és felügyelje.

A 4-2-es ábrán az adatrekord szerkezetét figyelhetjük meg.



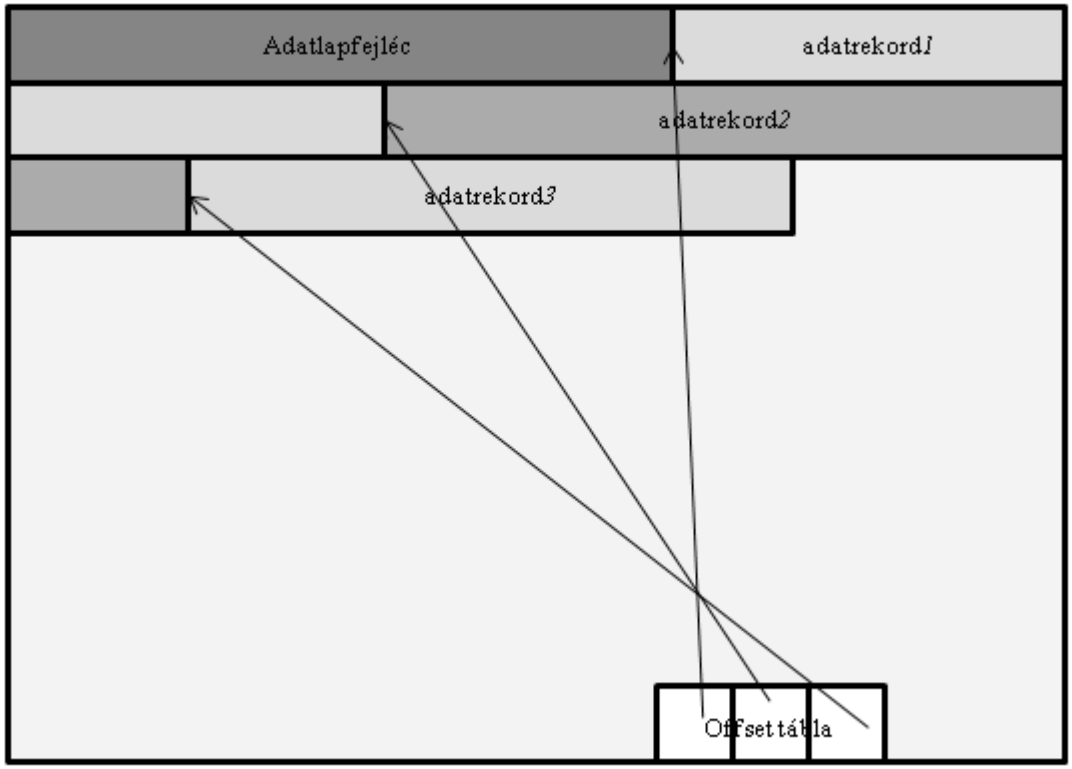
4-2. ábra Adatrekord-szerkezet

Az adatrekordokat azonban egyaránt el kell helyezni a memóriában és az állományokban is. A memória és az állományok felépítésének az alapegysége az adatlap. Azaz az adatrekord bele kell, hogy kerüljön az adatlapba. A legalapvetőbb kérdés az, hogy a kettő közül melyik a nagyobb. Az esetek nagy részében egy adatlapra több adatrekord kerülhet, de nagyméretű adatrekordnál előfordulhat, hogy az nem fér rá egy adatlapra. Nagyméretű adatrekord akkor fordulhat elő, ha egy táblában olyan adattípusú adatokat tárolunk, mint szövegek, képek, vagy más bináris nagyméretű objektumok. Ha más típusú rekordoknál is előfordul az, hogy az adatrekord nagyobb, mint az adatlap, akkor az adatbázis-adminisztrátor a telepítésnél, vagy a táblatér létrehozásánál nem jól választotta meg az adatlap méretét. Ez pedig teljesítményproblémákhoz vezethet. A teljesítményprobléma megoldására ilyenkor újra lehet szervezni a problémás táblateret. Vannak azonban olyan adatbázis-kezelő rendszerek is, amelyek nem engedik meg az adatlap méretének a változtatását.
Nézzük meg, mi van akkor, ha az adatlap mérete nagyobb, mint az adatrekord mérete. Tegyük fel, hogy az adatlap mérete L bájt, az adatrekord mérete R bájt. Vegyük L/R egészrészét, ami megadja, hogy egy adatlapra mennyi adatrekord kerülhet. Természetesen ekkor marad az adatlapon kihasználatlan terület. Ez nagy problémát nem okoz. Ha mégis ki akarná az adatbázis-kezelő rendszer használni ezt a kis területet, akkor ide elhelyezheti egy rekordnak egy részét. A rekordok részeit azonban mutatókkal kell összekapcsolni. Általában a teljesítmény szempontjából hatékonyabb, ha az a kis adatlaprész inkább kihasználatlan marad, minthogy a mutatók mentén kelljen összeszedni az adatrekordot. Ezért az adatbázis-kezelő rendszerek általában inkább kihasználatlanul hagyják azt az adatlaprészt.
Ha az adatrekord mérete nagyobb, mint az adatlap mérete, akkor a rekordot részekre vágja az adatbázis-kezelő rendszer és több adatlapon helyezi el. A szétvágott rekordokat mutatók segítségével kapcsolja össze.
A különböző adatbázis-kezelő rendszerek más-más felépítésű adatlapot használhatnak, de általában elmondhatjuk, hogy egy adatlap 3 alapkomponenst tartalmaz:

- *adatlapfejléc*: néhány bájtnyi fizikai gazdálkodási információ. Az adatlaplapfejléc tartalmazhat egy adatlap-azonosítót, mutatókat a megelőző és következő adatlapra, egy azonosítót, amely megmutatja, hogy az adatlap melyik táblához tartozik és szabad tár mutatókat.
- *adatrekordok*: a sorok adattartalmát tartalmazza.
- *opcionális offset tábla*: mutatókat tartalmaz, melyek az adatlap egyes adatrekordjait címzik. Néhány adatbázis-kezelő rendszer mindig használ offset táblát, mások csak akkor, ha változó hosszúságú rekordok vannak az adatlapon.

Az adatlapszerkezet és a komponensek hossza adatbázis-kezelő rendszerről adatbázis-kezelő rendszerre változhat.

A 4-3. ábrán az adatlap felépítésének vázlatát láthatjuk.



4-3. ábra Az adatlap felépítése

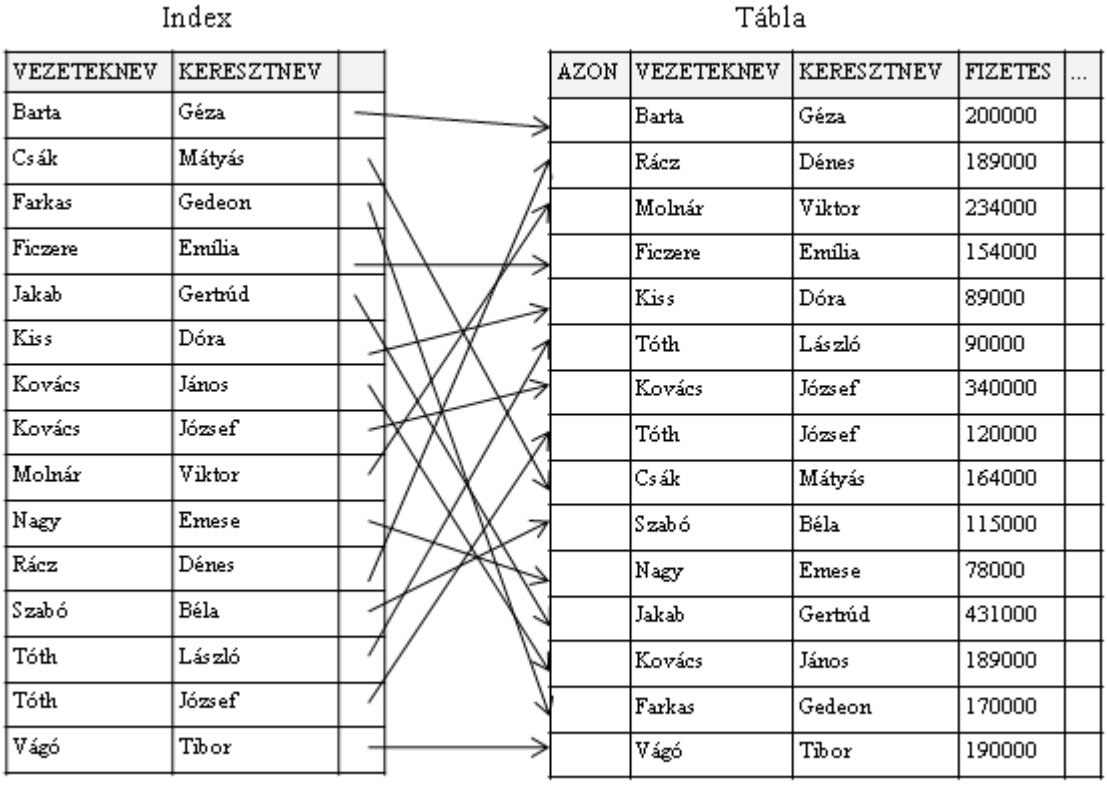
Index fogalma

Az indexrekord leírása előtt tisztáznunk kell, hogy egy adatbázis-kezelő rendszerben mi is az az index, miért olyan fontos.
Tekintsünk ehhez egy egyszerű lekérdezést:

```
SELECT VEZETEKNEV, KERESZTNEV
FROM DOLGOZO
WHERE VEZETEKNEV='Kovács';
```

Ha a táblánkon nincs index, akkor az adatbázis-kezelő rendszernek a DOLGOZO táblához tartozó összes sorhoz tartozó adatlapot fel kell olvasnia, hogy megtalálja a lekérdezésnek megfelelő értékeket. Ha még ráadásul a WHERE feltételben az elsődleges kulcsra vonatkozó feltételt írnánk, akkor ugyan tudjuk, hogy csak egy sort kell megtalálnunk, de ugyanúgy végig kell nézni a tábla összes sorát. Az ilyen típusú keresések megkönnyítésére hozhatunk létre egy táblához egy vagy több indexet.
Az index segítségével az adatbázis-kezelő rendszernek nem kell a tábla minden sorát végignéznie, hanem annak csak egy töredék részét. Az index alapjául szolgáló oszlopokat indexkulcsoknak nevezzük.
Az indexekre legegyszerűbb módon úgy tekinthetünk, mint kulcs-mutató párok halmazára. A kulcs a keresési kulcs lesz, amelynek az értékei az indexelt tábla oszlopértékeinek vagy transzformált oszlopértékeinek felelnek meg, a mutató pedig egyedi index esetén arra az adatrekordra mutat, amely tartalmazza a keresési kulcsot, egyébként pedig rekordcsoportot címez.
Az indexek lehetnek a kulcsérték szerint rendezettek, lehetnek összetettek, ebben az esetben a kulcs

összetett, több oszlopból tartalmaz adatokat.
Mindenesetre az adatbázis-kezelő rendszernek sokkal könnyebb dolga van akkor, ha az indexbejegyzések között kell megkeresnie egy adott értéket, majd a mutató alapján megtalálni a keresett adatrekordot.
A 4-4-es ábrán a DOLGOZOK tábla adatai alapján felépített rendezett indexet láthatunk.



4-4. ábra Index

A jegyzetnek nem témája az index fogalmát jobban leírni. A különböző indextípusok alapjául szolgáló adatszerkezeteket előismereteinkből ismerjük, a pontos megvalósításukat pedig az adott adatbázis-kezelő rendszer dokumentációja leírja.

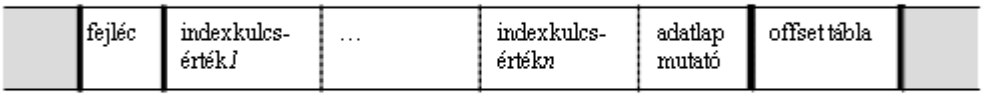
Indexrekord

Az indexeket a táblákhoz hasonlóan táblaterekhez kell rendelnünk. Ezek a táblaterek lehetnek ugyanazok, mint a táblákhoz rendelt táblaterek. Elképzelhető például, hogy egy táblatérbe van elhelyezve a DOLGOZO, és a RESZLEG tábla, illetve a MUNKAKOR táblának az adataira felépített MUNKAKOR_PK index. Ennek megfelelően a táblaterekhez tartozó állományok ugyanazok, azaz a felépítésük alapegysége továbbra is az adatlap.
Az adatlapokra indexrekordokat helyezünk el. Ezt hasonló módon tehetjük meg, mint azt az adatrekordoknál láttuk. Azonban az indexrekordok szerkezete más, mint az adatrekord felépítése. Egy általános indexrekord a következőket tartalmazza:

- *fejléc*: mint az adatlapoknál, az indexrekordok is általában néhány bájtnyi fizikai adminisztrációs információval kezdődnek, részletezve az indexrekord struktúráját és elrendezését.
- *indexkulcsértékek*: az indexkulcsokhoz tartozó aktuális adatértékek vannak felsorolva a definíció sorrendjében.

- *adatlapmutató*: ez mutatja azon táblához tartozó adatlap fizikai helyét, amelyben az indexelt adat jelenleg található.
- *opcionális offset tábla*: az indexrekordban tárolt változó hosszúságú mezők pozíciójának kezeléséhez és felügyeletéhez.

A 4-5-ös ábrán az indexrekord szerkezetét figyelhetjük meg.



4-5. ábra Indexrekord-szerkezet

Kiterjesztés

Alapvetően az operációs rendszerben az egy állományhoz tartozó blokkok nem folyamatosan helyezkednek el a lemezen. Az operációs rendszer az állomány blokkjait egymással mutatókkal kapcsolja össze. Azonban mégis szokott lenni valamennyi blokk, amely folyamatosan helyezkedik el a lemezen. Az adatbázis-kezelő rendszer általában az operációs rendszerben egyszerre nem csak egy adatlapnak megfelelő darabszámú blokkot foglal le, hanem többet. A több adatlaphoz tartozó, a lemezen folyamatosan elhelyezkedő blokkokat nevezzük együtt kiterjesztésnek (extent).
Alapvetően egy adatbázis-objektumhoz az adatbázis-kezelő rendszer először csak egy kiterjesztést használ. Később, ahogy az adatbázis-objektum nő, egyre több tárterületre van szüksége, ezért újabb és újabb kiterjesztéseket foglal le az adatbázis-kezelő rendszer. Ezeket az újonnan lefoglalt kiterjesztéseket nevezzük másodlagos kiterjesztéseknek.
A táblatér tulajdonságaként adható meg, hogy mekkora legyen a táblatérbe létrehozott adatbázis-objektumhoz az elsődleges és a másodlagos kiterjesztések mérete.

Állománykiterjesztés

Előfordulhat, hogy egy állomány eléri az adatbázis-kezelő rendszerben megadott maximális méretét, azonban az állományba még újabb adatokat szeretnénk beszúrni. Ekkor állománykiterjesztést hozhat létre az adatbázis-adminisztrátor. Ez egy új állomány, amely az eredeti állományhoz van kötve és csak az eredeti állománnyal együtt lehet használni.

Táblamódosítások

Az adatbázisbeli táblákban lévő adatokat a felhasználók bővíthetik, törölhetik vagy módosíthatják. Ami azt jelenti, hogy a lemezen lévő állományok tartalmát módosítja az adatbázis-kezelő rendszer.
A táblába egy új sor beszúrása nagyon egyszerű: az adatbázis-kezelő rendszer a táblát tartalmazó táblatérhez rendelt megfelelő állományban keres egy olyan adatlapot, amelyen van hely és oda teszi be az új rekordot. Ha nem talál ilyen adatlapot, akkor keres az állományba egy új adatlapot és oda szúrja be az új rekordot. Új adatlap vagy létezik az előző kiterjesztés lefoglalása miatt, vagy új kiterjesztésre van szükség.
Egy táblabeli sor törlése esetén az kézenfekvő lenne, hogy az adott hely egyszerűen felszabadul.

Azonban ez sem ilyen egyszerű, hiszen egy adatlapon több rekord van. Ha nem az utolsót töröljük, akkor ahhoz, hogy a szabad helyet a törlés után jól ki lehessen használni, az adatlapot újra kell szervezni vagy esetleg több adatlap összevonásával egy adatlapot meg lehet szüntetni.
A táblabeli módosítás esete pedig jelentheti azt, hogy az adatrekord mérete ugyanakkora marad, nő vagy éppen csökken. Ha ugyanakkora marad, akkor nincs vele gond, a megfelelő értékeket kicseréli az adatbázis-kezelő rendszer. A rekord mérete attól függően nő vagy csökken, hogy a rekordban lévő változó hosszúságú mezők hogyan változnak. Ha a rekord mérete nő, akkor keresni kell neki új helyet. Ha az adatbázis-kezelő rendszer nem talál ugyanazon az adatlapon szabad helyet, mint amelyiken a rekord van, akkor adatbázis-kezelő rendszertől függően vagy a teljes rekordot áthelyezi, vagy a rekordnak csak egy részét helyezi át új adatlapra. A második esetben mutatókat használ, hogy a rekord részeit összekapcsolja. Ha a rekord mérete csökken, akkor az ugyanolyan problémákat vet fel, mint egy sor törlése.

A táblaméret kiszámítása

Ha az adatbázis-adminisztrátor egy táblához a szükséges tárterületet számolja, akkor természetesen a rekordhosszúsággal és nem a sorhosszúsággal kell számolnia. A rekord hossza tartalmazza a fejlécet és az offset táblát is. A sor hossza a tábla oszlophosszainak az összege. Az adatbázis-adminisztrátornak ki kell tudnia számolni a pontos sorhosszat is, mivel a sorhossz az egyik része a rekord hosszának. Ez sem könnyű feladat, hiszen az oszlopok lehetnek változó hosszúságúak is.
A sor adatrészhosszának a kiszámításához az adatbázis-adminisztrátornak dokumentációra van szüksége, amely leírja az adatbázis-kezelő rendszer által támogatott adattípusok aktuális fizikai hosszának a meghatározását. Ezen adattípusok tárolásához szükséges hosszak adatbázis-kezelő rendszerről adatbázis-kezelő rendszerre változnak.
Ha az adatbázis-adminisztrátor ismeri az adatlap szerkezetét és ki tudja számolni a rekord hosszát, akkor egy tábla méretének a kiszámítása már könnyű. A rekordhossz kiszámítása után a következő lépés meghatározni, hogy mennyi sor fér el egy adatlapra. Egy adatlap mérete adatbázis-kezelő rendszerről adatbázis-kezelő rendszerre változik, és a legtöbb adatbázis-kezelő rendszer esetén az adatbázis-adminisztrátor határozhatja meg a különböző adatlapméreteket.
Például tegyük fel, hogy egy adatlap mérete 2 KB, de ebből 32 bájt szükséges az adatlap fejléc-információinak. Marad 2016 bájt, amelyen adatsorokat lehet tárolni. Ha elosztjuk a 2016 bájtot egy adatsorhoz tartozó rekord méretével, akkor megkapjuk azt, hogy mennyi sor fér rá egy adatlapra.
A táblaméretet úgy kaphatjuk meg, hogy kiszámoljuk, hogy mennyi adatlap szükséges a táblában tárolt sorok számához és megszorozzuk az adatlapmérettel. A példánk alapján: a sorok számát elosztjuk az adatlaponkénti sorok számával és ezt az értéket megszorozzuk 2 KB-tal, ami az adatlapnak a mérete.
Természetesen az adatbázis-adminisztrátornak számolnia kell a táblához meghatározott szabad területtel is, hiszen a tábla adatait frissíthetik, törölhetik, vagy lehet, hogy új adatok kerülnek a táblába. Ehhez azt is meg kell tudnia becsülnie, hogy a tábla tartalma milyen gyakran fog módosulni és azok milyen típusúak lesznek (frissítés, törlés, beszúrás). Azt se feledjük, hogy vannak olyan adatok, amelyek a táblán kívül, de az adatbázison belül tárolódnak, mint a szövegoszlopok, vagy a LOB oszlopok. Sőt megjelenhetnek adatok külső, operációs rendszerbeli állományokban is.

Az indexméret számítása

Egy index tárolásához szükséges hely kiszámításának első lépése az, hogy kiszámítjuk az index sorméretét. Ez a sorméret függ az index típusától. Azonban hasonlóan, mint a táblabeli sor méretének a kiszámításához nemcsak az indexnek a sorméretét, hanem az indexrekord méretét is ki kell számolni. A rekordok hossza a sorok hossza plusz a sor tárolásához szükséges egyéb hosszúságok. A rekordméret kiszámolásához szükség van az általunk használt adatbázis-kezelő

rendszerben az indexekhez szükséges egyéb hosszúságokra.
Ha kiszámoltuk a rekordméretet, mehetünk a következő lépésre, az indexhez szükséges tárhely összegének a kiszámítására. Amely teljesen ugyanaz, mint a táblánál alkalmazott módszer.
Természetesen idővel az index is változik, ezért az adatbázis-adminisztrátornak az indexeknek is kell olyan helyet fenntartani, ahová tud bővülni, ha szükséges.

Adatbázisnapló

A tranzakciónapló vagy adatbázisnapló az adatbázis egyik legfontosabb állománya. Az adatbázisnaplóban az adatbázisban végrehajtott tranzakciók bejegyzései vannak.
A tranzakció az adatbázis-műveletek végrehajtási egysége. A tranzakciónak van kezdőutasítása és befejezőutasítása, ez utóbbi általában a COMMIT vagy a ROLLBACK utasítás, vagy olyan művelet, amely ezeknek az utasításoknak a hatását váltja ki. A tranzakció alapvetően DML utasításokból áll: INSERT, UPDATE, DELETE. A tranzakcióknak a következő tulajdonsággal kell rendelkezniük (ACID-tulajdonságok):

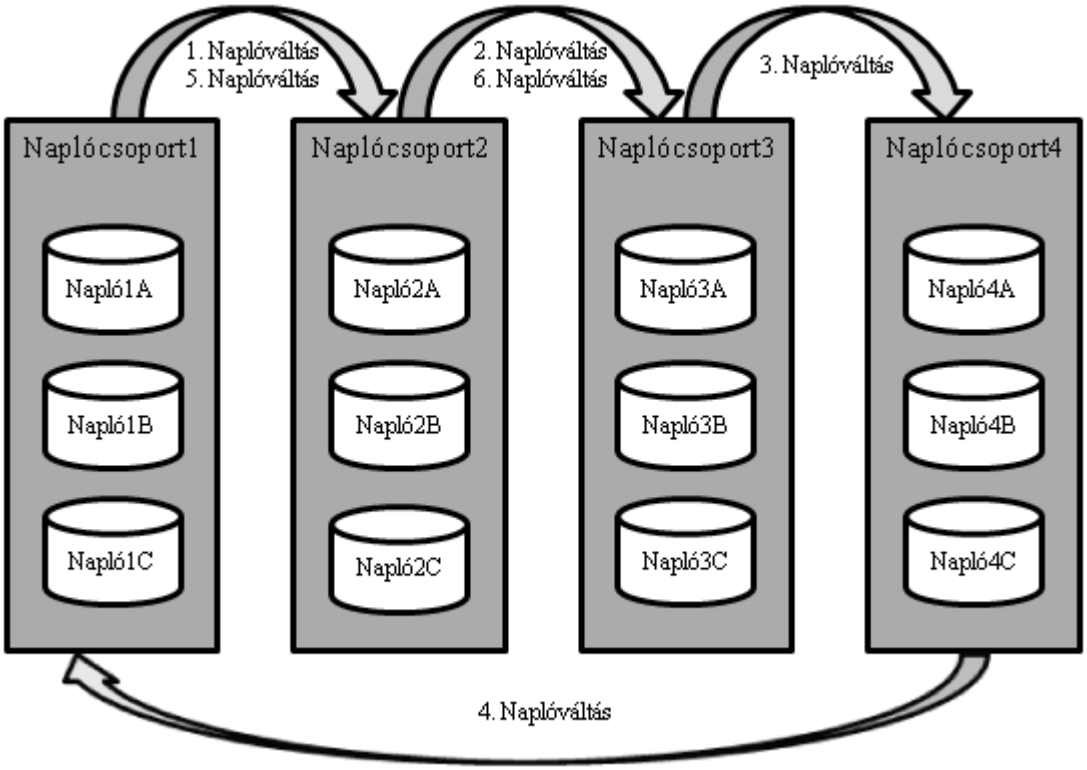
- *Atomosság* (**a**tomicity): a tranzakció „mindent vagy semmit” jellegű végrehajtása (vagy teljesen végrehajtjuk, vagy egyáltalán nem hajtjuk végre).
- *Konzisztenciamegőrzés* (**c**onsistency preservation): az a feltétel, hogy a tranzakció megőrizze az adatbázis konzisztenciáját, azaz a tranzakció végrehajtása után is teljesüljenek az adatbázisban előírt konzisztenciamegszorítások (integritási megszorítások), azaz az adatelemekre és a közöttük lévő kapcsolatokra vonatkozó elvárások.
- *Elkülönítés* (**i**solation): az a tény, hogy minden tranzakciónak látszólag úgy kell lefutnia, mintha ez alatt az idő alatt semmilyen másik tranzakciót sem hajtanánk végre.
- *Tartósság* (**d**urability): az a feltétel, hogy ha egyszer egy tranzakció befejeződött, akkor már soha többé nem veszhet el a tranzakciónak az adatbázison kifejtett hatása.

Ahhoz, hogy a tranzakciók ezen tulajdonságokat teljesíteni tudják, szükség van az adatbázisnaplóra.
Az adatbázisnaplóban naplóbejegyzések sorozata van. Naplóbejegyzések lehetnek a következőek:

- Az egyes tranzakciók kezdete és vége. A tranzakció vége nem feltétlenül csak a COMMIT vagy a ROLLBACK utasítás lehet. A tranzakció akkor is véget ér, ha áramszünet van, vagy ha a felhasználó munkamenete lezárul, mert kilépett a programból. A tranzakció kezdetét egyes adatbázis-kezelő rendszereknél külön jelezni kell, más adatbázis-kezelő rendszerek pedig automatikusan elkezdenek egy tranzakciót, amikor a felhasználó bejelentkezik vagy lezár egy tranzakciót.
- Az adat aktuális változásai, ennek az információnak elégnek kell lennie a módosítások visszavonásához az egyes tranzakciókon belül. Általában egy ilyen naplóbejegyzés tartalmazza, hogy melyik tranzakció, mikor, melyik adatbáziselemet milyen értékről milyen értékre módosította. Az adatbáziselem itt például egy tábla sorában egy mező, vagy egy teljes adatlap lehet.
- A COMMIT és ROLLBACK utasítások az egyes tranzakciónál.
- Egyéb bejegyzések.

Tudnunk kell azt is, hogy mielőtt bármilyen változás történik az adatbázisban, azt először az adatbázisnaplóba be kell jegyezni. Így, ha hiba történik az adatbázis-kezelő rendszerben, azaz leáll, akkor az adatbázisnapló bejegyzései alapján azoknak a tranzakciónak a hatása visszavonható, amelyek nem lettek véglegesítve, illetve ha a véglegesített tranzakciók műveletei nem kerültek az adatbázisba, akkor azokat az adatbázisnapló alapján újra végre lehet hajtani.
A naplóbejegyzések egy megnyitott adatbázisnapló-állományba kerülnek. Ez a naplóállomány az aktív napló. Azonban a naplóbejegyzések száma idővel egyre több lesz, amely egy idő után egy állományban kezelhetetlenül sok lenne. Ezért naplóváltásra kerül sor, amely azt jelenti, hogy az adatbázis-kezelő rendszer az aktuális aktív naplóállományt bezárja, és egy másik naplóállományba kezdi el a naplóbejegyzéseket írni. A naplóváltás történhet automatikusan és az adatbázis-adminisztrátor is

kikényszerítheti. Az automatikus naplóváltáshoz az adatbázis-kezelő rendszerek különböző opciókat biztosíthatnak. Ilyen lehet például, hogy megadható a naplóállományok mérete, így naplóváltás akkor történhet, amikor az adott állomány betelik. Másik lehetőség, hogy egy időintervallumot kell megadni, amely letelte után történik a naplóváltás.
Ha esetleg nincs adatbázismentés, és nincs szükség az adatbázisnapló archiválására, akkor felmerülhet a kérdés, hogy meddig szükséges őrizni a naplóbejegyzéseket. A régi naplóbejegyzésekre addig van szükség, amíg a hozzájuk tartozó változások az adatbázisban meg nem történtek, és az adott tranzakció véget nem ért. Kérdés, hogy ezt honnan lehet tudni. Az adatbázis-kezelő rendszer erre használja az ellenőrzőpontokat. Az adatbázisnaplóban található ellenőrzőpont-bejegyzés alapján az adatbázis-kezelő rendszer biztosan tudhatja, hogy az adatbázisnaplóban az az ellenőrzőpont előtti naplóbejegyzésekre már nincs szükség, ha nem szükséges az adatbázisnaplót archiválni. Ebben az esetben minden adatbázisbeli módosítás bekerült az adatbázisba is, és ezért az adatbázisnaplót nem szükséges az ellenőrzőpont-bejegyzés elé visszagörgetni. A naplóállományokat azonban csak addig szükséges őrizni, amíg szükség van rájuk.
Az adatbázis-kezelő rendszerek így rendelkeznek általában egy aktív naplóállománnyal, és legalább egy de inkább néhány inaktív naplóállománnyal. Naplóváltáskor egy inaktív naplóállományt nyit meg az adatadatbázis-kezelő rendszer. Természetesen a naplóállományok számát és helyét az adatbázis-adminisztrátornak előre meg kell mondani. Az adatbázis-kezelő rendszer a naplóállományokat ciklikusan használja. Legalább két naplóállományra van szükség.
Láthatjuk, hogy az adatbázisnapló állományai valóban nagyon fontosak az adatbázisunk életében. Az aktív naplóállományt nagyon kell védeni, mert ha elveszik, akkor az adatbázisunk nem lesz konzisztens, nem fogjuk tudni használni. Mivel ez egy megnyitott állomány, ezért menteni még nem lehet, de pontos másolatot készíteni róla igen. A legtöbb adatbázis-kezelő rendszer támogatja a többszörös redundáns naplókat. Azaz az adatbázis naplóállományai legalább két példányban két különböző lemezen léteznek. Így ha az egyik naplóállománnyal történik valami, és használhatatlanná válik, akkor a hiba kijavításáig a naplóállomány másik példányát lehet használni. Az adatadatbázis-adminisztrátornak kell eldöntenie, hogy hány példányban hozza létre az adatbázis naplóállományait. Ezeket a példányokat lehetőleg nem egy háttértárolón helyezi el az adatbázis-adminisztrátor. Az azonos tartalmú naplóállományok együttesét gyakran naplócsoportnak hívják.
A 4-6-os ábrán az adatbázisnapló-állományok közötti naplóváltást láthatjuk, redundáns naplózás megvalósítás esetén. Minden egyes naplóállományból 3 egyforma példány, azaz 3 naplócsoport van, és a naplóváltások miatt 4 különböző naplóállományt láthatunk az ábrán. Az ábra nem mutatja be a naplóarchiválás folyamatát.



4-6. ábra Naplóváltás

Az adatbázis-adminisztrátornak tehát először döntenie kell arról, hogy mennyi adatbázisnapló-állományt hoz létre, aztán dönteni kell arról, hogy hány példányban tárolja őket. Ha 5 naplóállományt használ 3-3 példányban, akkor máris 15 naplóállományról beszélünk. Ezeket a döntéseket nagyban befolyásolja az is, hogy az egyes naplóállományoknak mennyi lehet a maximális mérete. Ha nem adunk meg maximális méretet, akkor a napló esetleg végtelen nagyra megnőhet, betöltheti a teljes háttértárolót. Ha betölti, akkor a következő naplóbejegyzést sem fogja tudni hova írni, ami lelassuló folyamatokat vagy teljes leállást okozhat. Fontos kérdés tehát az, hogy a naplóállományoknak mekkora legyen a mérete. Az adatbázis-adminisztrátor az adatbázisban történő tranzakciós tevékenység mennyiségére alapozva állapíthatja meg a naplóállományok méretét. A legfontosabb cél azonban az, hogy az adatbázisnapló állományai a legforgalmasabb időszakban is ki tudják szolgálni a teljes adatbázisbeli tevékenységeket teljesítménycsökkenés nélkül.
Az adatbázisnapló tárolási igényével kapcsolatban van még egy fontos kérdés. Ha az adatbázisunkat mentjük, márpedig a legtöbb esetben ez történik, akkor a naplóállományokat menteni, archiválni kell, és az adatbázismentéssel együtt kell a továbbiakban kezelni. Naplóváltáskor a régi állományt, amely tovább már nem változik, azonnal lehet archiválni. Ez történhet automatikusan és kézzel is. Az archivált naplóállományoknak azonban ugyancsak szükségük van tárterületre. Célszerű őket először a lemezre archiválni, mert így az archiválás gyorsabb, majd minél hamarabb, de legkésőbb 24 óra múlva más tárolási területre, például szalagra menteni őket.

Növekedő tárhelyigény

Amikor egy adatbázistáblát módosítunk, az adatbázis mérete nő. Ne feledjük, hogy az adatbázis nem csak adatrészből, hanem napló részből is áll. Időnként és következetesen figyeljük az adatbázis

helyhasználatát. Ezt megtehetjük az adatbázis-kezelő rendszerrel járó eszközök és segédprogramok segítségével, vagy tároláskezelő-szoftverrel vagy más adatbáziseszközzel. Az adatbázis-adminisztrátornak a következőket kell figyelemmel kísérnie a tárigénnyel kapcsolatosan:

- a másodlagos kiterjesztések száma,
- állománykiterjesztések,
- eszköz töredezettsége,
- elérhető szabad helyek,
- a táblaterek mérete,
- a táblaterekben lévő táblák és indexek, egyéb adatbázis-objektumok darabszáma és mérete,
- a jelenleg nem használt fenntartott területek összmérete,
- objektumok, melyek helyhiánnyal küzdhetnek.

Készítsünk tervet az adatbázis növekedésének kezelésére. Első lépésben nyerjük vissza azokat a tárhelyeket, amelyeket olyan adatbázis-objektumok foglalnak, amelyeket már nem használunk. Egyszerűen dobjuk el őket. Vannak olyan objektumok, amelyeket létrehoztunk, aztán elfeledkeztünk róluk, és többé nem használjuk őket. A használaton kívüli adatbázis-objektumok eldobásával a helyük felszabadul.
Ha még így sincs elég helyünk, az adatbázis által elérhető tárat ki kell terjeszteni. Ezt megtehetjük egy olyan ALTER utasítással, ami további helyet határoz meg egy vagy több adatbázisbeli táblatér számára. Néhány adatbázis-kezelő rendszer esetén az ALTER utasításban a növekedés összegét, máshol az új tárolási méretet kell megadni a táblatérhez. Az adatbázis-adminisztrátornak lehet, hogy egy új tárolóeszközt is az adatbázis rendelkezésére kell bocsátania. Ekkor adatbázis-objektumokhoz tartozó állományok lemezek közötti mozgatására is sor kerülhet.

Állományok elhelyezése

Sok tárolási kérdést kell eldönteni, mielőtt az adatbázis-adminisztrátor egy adatbázist létrehozhat. Az egyik legfontosabb kérdés az, hogy mennyi helyet ad az adatbázisnak. A táblabeli adatokon kívül a táblaterekhez az indexeket is hozzá kell rendelni. Alapvetően gondolkodhatunk úgy, hogy egy táblatér egy állománynak felel meg. Azonban az adatokon és az indexeken kívül a tranzakciós naplókkal is számolni kell. A tranzakciós napló megfelelő tárolása az adatoktól és indexektől különböző állományokban történhet.
Miután meghatároztuk a szükséges tárigényt, az adatbázis-adminisztrátornak választania kell egy elegendő tárterülettel rendelkező, megfelelő háttértárat az adatbázis állományainak a tárolására. Az adatbázis-adminisztrátor a különböző állományokhoz több háttértárat választhat azért, hogy

- az állományok teljesítménykövetelményeit segítse a megfelelő lemezeszközökkel,
- elkülönítse az indexeket az adatoktól teljesítmény okokból,
- a naplóállományt elkülönítse egy különálló és nagyon gyors eszközre,
- az ideiglenes és munkaállományokat elkülönítse, hogy ha lemezhiba történik, akkor az ideiglenes állományokat lehessen törölni, és újradefiniálni mentés és visszaállítás nélkül,
- az adatokat több eszközre helyezze, hogy elősegítse a párhuzamos elérést.

Állományok elhelyezése a lemezen

Az adatbázis-adminisztrátornak meg kell határoznia az optimális állományelhelyezkedést a lemezen. Időnként az adatbázis-adminisztrátor teljesítménynövelést érhet el, ha az állományokat az egyik fizikai lemezről a másikra mozgatja. Általános technika, hogy a táblabeli adatokat tartalmazó állományt és az adott táblára létrehozott indexeket tartalmazó állományt különböző lemezeszközökre helyezi el az adatbázis-adminisztrátor. Az indexek és az indexelt adatok elkülönítésével az input/output műveleteket sokkal hatékonyabbá lehet tenni, mert a lemezeszköz fizikai író-olvasó fejének nem kell többször mozognia. Ugyanebből az okból ugyanazon műveletek

által elért adatok különböző fizikai lemezeszközökre helyezése egy másik általános állományelhelyezési technika és ugyanolyan előnyökkel jár, mint az indexek elkülönítése az adatoktól.
Modern tárolási eszközök (RAID) használatával a pontos állományelhelyezés elveszíti jelentőségét. Bármilyen tárolási eszközt használunk, biztosítsuk, hogy a tranzakciónapló a táblák adatait tartalmazó állományoktól mindig elkülönített eszközön legyen, és hogy a tranzakciónapló mentése független legyen az adatbázis mentésétől. Így növeljük a visszaállíthatóságot.
Minden adatbázis-kezelő rendszer más tárolási módszert alkalmaz. Ismerjük meg az általunk használt adatbázis-kezelő rendszer azon mechanizmusát, amely a tárolási alrendszerekkel és a lemezekkel kölcsönhatásban áll és az adatbázis-állományokat hozza létre. Nem megfelelően létrehozott adatbázis-állományok a gyenge teljesítmény jelentős okai lehetnek.
Az adatbázis-kezelő rendszerek szállítói ajánlanak olyan terméket, amelynek a segítségével a tárolást a rendszerre lehet bízni (system-managed storage, rendszer által kezelt tárolási megvalósítás, SMS). Az SMS-sel az adatbázis-adminisztrátor helyett a rendszer határozza meg az állományok aktuális helyét. Az új, hatékony lemeztechnológiákkal az SMS sokkal hasznosabb opció, mint régen volt.
Lehetséges, hogy a rendszerre csak az adatbázis-állományok nagy részének az elhelyezését bízzuk, azonban bizonyos teljesítményérzékeny állományokhoz (mint a tranzakciónapló) nem a rendszer által kezelt állományelhelyező technikát használunk. Így a rendszer megoldja a legtöbb adatbázis-állomány elhelyezését, és az adatbázis-adminisztrátornak csak a legszükségesebb adatbázis-állományok elhelyezésével kell foglalkoznia.

Raw partíció és állományrendszer

A raw partíció egy egyszerű lemezeszköz, amelyen nincs operációs rendszer vagy állományrendszer telepítve. Általában akkor lehet használni, ha az adatbázis UNIX operációs rendszerre került, de az újabb Windows verziók mellett is létezik.
Amikor az írásokat az operációs rendszer puffereli, az adatbázis-kezelő rendszer nem tudja, hogy az adat fizikailag a lemezre került-e. Az operációs rendszer helyett az adatbázis-kezelő rendszer írja az adatokat a gyorsítótárból a lemezre. Ha hiba történik, és nem raw partíciót használunk, az adatbázis adatai esetleg nem lesznek 100%-ig helyreállíthatóak. Ezt kerüljük el a raw partícióval.
Ha raw partíciót használunk, akkor az adatok az operációs rendszer közbeavatkozása nélkül kerülnek a lemezre. Raw partíció használata esetén az adatbázis-kezelő rendszer kezeli a raw partícióval létrehozott tárat, így azt is az adatbázis-kezelő rendszer biztosítja, hogy legyen elég elérhető hely. Ha raw partíciót használunk, az operációs rendszer nem foglalja le a szükséges helyet az állománynak. Ha más alkalmazás is ugyanazért a területért versenyzik, akkor az esetleg nem látja, hogy az adatbázis-kezelő rendszer használja a területet, ezért esetleg nem lesz annyi hely az adatbázis adatainak, mint amennyit gondoltunk. Ezt megelőzhetjük azzal, hogy az adatbázis-kezelő rendszer előre lefoglalja a szükséges területet, de nem tölti ki.
A raw eszköz hátránya, hogy nehéz nyomon követni az adatbázis-állományokat. Ha nem az operációs rendszer vagy az állományrendszer foglalta le az állományokat, akkor nem lehet operációs rendszerbeli vagy állomány rendszerbeli utasításokkal nyomon követni őket. Esetleg más szoftvertermékek segíthetnek a probléma megoldásában.
Az adatbázis-kezelő rendszerek újabb verzióihoz a szállítók nem javasolják. Nehezebb kezelni, és teljesítményben is csak kb. 5-10%-os javulást eredményezhet. A mai tárolást támogató eszközök ettől sokkal jobb teljesítményt és kezelhetőséget biztosítanak úgy, hogy közben operációs rendszerbeli állományokat használ az adatbázis-kezelő rendszer.

Ideiglenes vagy munkaállományok

Az adatbázis-kezelő rendszerek ideiglenes adatbázis-objektumokat hozhatnak létre, amelyek csak

addig léteznek, amíg egy bizonyos tranzakció hatásköre tart. Ezek az objektumok ideiglenes adatokat tartalmaznak, amelyek átmenetiek és nem szükséges őket hosszú ideig tárolni, csak amíg használják őket. Ilyen ideiglenes vagy munkaállományt használhat az adatbázis-kezelő rendszer például az adatok rendezéséhez.
Adatbázis-kezelő rendszertől függ, hogy az ideiglenes adatbázis-objektumok használatához az adatbázis-adminisztrátornak kell-e lemezeszközt és tárterületet az adatbázishoz rendelni.

Tömörítés

A legtöbb adatbázis-kezelő rendszer termék biztosít az adatok tömörítéséhez valamilyen eljárást. Ha mégsem, akkor lehet más terméket vásárolni. A tömörítéssel kevesebb hely szükséges ugyanahhoz az adatmennyiséghez. Egy tömörítő rutin algoritmikusan tömöríti az adatokat, amikor az adatbázisa kerülnek, és kitömöríti őket, amikor visszanyerjük őket. A tömörítés a processzort terheli, mert az adatelérés megköveteli az adatok tömörítését és kitömörítését. Az input/output műveletek sokkal hatékonyabbak tömörítés esetén, mert az adatok a lemezen kevesebb helyet foglalnak, és így egy input/output művelettel több adatot lehet olvasni.

Tervezés a jövőre nézve

A legtöbb adatbázis megvalósítása nem statikus. Folyamatosan fejlődik, lekérdezik az adatait, módosítják, kimentenek belőle, töltenek be új adatokat, törölnek adatokat, újraszervezik. Az adatbázis változásával a tárolási követelmények is változnak.
Az adatbázis-adminisztrátornak mindig gondolnia kell a jövőbeli növekedésre. Tudnia kell, hogy mennyi adat van az adatbázisban és mennyi felhasználó éri el az adatokat. Bármelyik növekedése jelentheti, hogy az adatbázis tárolási megvalósításán változtatni kell.

Kapacitástervezés

A kapacitástervezésben megmérjük a teljes rendszer kapacitását és összehasonlítjuk a követelményekkel. Az összehasonlítás célja a rendszer által elérhető erőforrások megfelelő beállítása. A kapacitástervezéshez meg kell érteni az újonnan felmerülő kezdeményezéseket és azok hatását a meglévő hardver- és szoftverinfrastruktúrára.
Eldönthetjük, hogy a létező infrastruktúra elegendő-e a jövőbeli munkaterhelésre, ha megmérjük az aktuális kapacitást, felbecsüljük a kapacitásnövekedést és számolunk az új vállalati és IT kezdeményezések követelményéhez járuló kapacitással. Ha a vázolt növekedés nagyobb, mint amit a rendszerünk képessége támogatni tud, akkor meg kell becsülni a változtatások költségét és a lehetőségekhez mérten bővíteni az infrastruktúrát.
Az új adatok és felhasználók támogatására nem elég új lemezt behelyezni és az adatbázis-kezelő rendszerhez rendelni. Más feladatok is vannak:

- alkalmazások újratervezése
- adatbázisok újratervezése
- adatbázis-kezelő rendszer paramétereinek módosítása
- hardverkomponensek újrakonfigurálása
- szoftverinterfészek módosítása
- licenszbővítés

A tárfogyasztást több oldalról is figyelhetjük. Az egyik lehetőség, ha összesítve figyeljük a felhasznált lemezterületek arányát. Másik lehetőség, ha szerverenként határozzuk meg a lemezterület felhasználásának a gyorsaságát. Az állományrendszer tárolási szükségleteit is monitorozhatjuk. Az adatbázis-adminisztrátor csak az adatbázis-kezelő rendszerrel kapcsolatos állományokra és tárigényre kíváncsi. Az adatbázis növekedésével a következő kérdésekre kell tudnunk válaszolni:

- Mikor lesz szükség több tárra?
- Mennyi új tár szükséges?
- Hol szükséges az új tár?
- Mit kell tenni, hogy az új tárat működésbe helyezzük?

Összefoglalás

A fejezetben megnéztük, hogy az adatbázis-kezelő rendszernek milyen háttértárigénye van. Megnéztük azt, hogy a háttértár kiválasztásához milyen szempontokat kell figyelembe venni. Megvizsgáltuk, hogyan lehet kiszámolni azt, hogy az adatbázis-kezelő rendszernek mekkora tárigénye lesz az adataink tárolásához. Ehhez megismerkedtünk a táblatér, az adatrekord, az index, az indexrekord, az adatlap, a kiterjesztés, állománykiterjesztés fogalmakkal és ezek kapcsolatával. Ezek közül részleteztük az adatrekord, az indexrekord és az adatlap szerkezetét is. Betekintettünk a tárolás mechanizmusába. Az adatbázis-adminisztrátornak meg kell becsülnie, hogy az adatbázisnak mekkora tárterületre van szüksége. Ennek a munkának egy része, hogy ki tudja azt számolni, hogy a táblák és az indexek az adminisztratív információkkal együtt mennyi tárhelyet foglalnak el a lemezen. A továbbiakban részleteztük az adatbázis-kezelő rendszerek egyik legfontosabb állománycsoportját, az adatbázisnapló jellemzőit. A naplózáshoz áttekintettük a tranzakciók fogalmát és az ACID tulajdonságait, és megvizsgáltuk, hogy az adatbázisnaplóba milyen típusú naplóbejegyzések kerülnek. Megismerkedtünk a naplóváltás és a redundáns naplózás fogalmakkal. Megnéztük azt, hogy az adatbázis növekedésével kapcsolatban az adatbázis-adminisztrátornak milyen feladatai vannak. Szempontokat vettünk át az állományok háttértáron való elhelyezésével kapcsolatban. Megnéztük a raw partíció és a tömörítés előnyeit és hátrányait. És végül megvizsgáltuk, hogy milyen munkával jár az adatbázis-adminisztrátor számára annak a kezelése, hogy az adatbázis folyamatos növekszik.

Ellenőrző kérdések

1. Milyen alapvető tényezőket kell figyelembe venni a megfelelő tárolási technológia kiválasztásához?
2. Milyen modern tárolási megoldásokról hallottunk?
3. A tárolási rendszer kialakításakor milyen célokat érdemes szem előtt tartani?
4. Mi az a táblatér?
5. Mi az az adatlap? Hogyan épül fel?
6. Mi az adatrekord? Hogyan épül fel?
7. Egy adatlapra hány darab adatrekordot lehet elhelyezni?
8. Mi az az index? Miért hasznos?
9. Hogyan épül fel az indexrekord?
10. Mi az a kiterjesztés? Mi a másodlagos kiterjesztés?
11. Mit jelent az állománykiterjesztés?
12. Táblamódosítások esetén hogyan változik a lemezen lévő adatlapok tartalma?
13. Hogyan lehet kiszámolni, hogy egy táblának mennyi tárterületre van szüksége?
14. Hogyan lehet kiszámolni, hogy egy indexnek mennyi tárterületre van szüksége?
15. Mi az a tranzakciónapló vagy adatbázisnapló?
16. Mik a tranzakciók ACID tulajdonságai?
17. Milyen naplóbejegyzések vannak az adatbázis-naplóban?
18. Mennyi állomány tartozik az adatbázisnaplóhoz?
19. Mit jelent a többszörös redundáns adatbázis-naplózás?
20. Mi az ellenőrzőpont?
21. Meddig szükséges őrizni a naplóbejegyzéseket?

22. Mi alapján választja meg az adatbázis-adminisztrátor a naplóállományok méretét?
23. Miért szükséges több háttértárat használni egy adatbázis tárolásához?
24. Milyen szempontokat vegyünk figyelembe, amikor az állományokat különböző lemezekre helyezzük el?
25. Mi az a raw partíció? Milyen előnyei és hátrányai vannak?
26. A tömörítésnek milyen előnyei és milyen hátrányai vannak?
27. A jövőbeli adatbázis-növekedéssel járó tárigény kezelésére az adatbázis-adminisztrátor hogyan tud felkészülni?

5. fejezet – Adatok mozgatása

Az adatok mindig mozgásban vannak. Ha egyszer az adat létrejött, akkor azt különböző célokból mozgatjuk. Így ugyanaz az adat megjelenhet különböző informatikai környezetekben futó, különböző adatbázis-kezelő rendszerekben, amelyeket különböző felhasználók különböző alkalmazások segítségével érnek el.
Általában a cégeknek nem elegendő az adatoknak egyetlen másolata. Az adatról a cégek másolatot készítenek, transzformálják, tisztítják, duplikálják és többször mentik. Ugyanazon adat különböző másolatait használják a tranzakciófeldolgozáshoz, az elemzések támogatására, tesztelésre, minőségbiztosításhoz, a napi műveletekhez, jelentésekhez, adattárházakhoz, adatpiacokhoz, adatbányászathoz, illetve elosztott adatbázisokhoz is. Az adatbázis-adminisztrátor feladata ezt a hatalmas mennyiségű adatfolyamot kontrollálni.
Az adatbázis-adminisztrátor különféle technikákat, és technológiákat használhat az adatok mozgatásának és elosztásának megkönnyítésére.

Adatbetöltés és adatkimentés

Az adatok mozgatásának egyik legegyszerűbb módja a LOAD és az UNLOAD segédprogramok használatával történik. Mindkét segédprogram az adatbázis-kezelő rendszer beépített eszköze. A LOAD segédprogrammal új adatokat vihetünk be egy táblába, míg az UNLOAD segédprogrammal egy tábla adatait vagy egy lekérdezés eredményéből származó adatokat írhatjuk egy adatállományba.

A LOAD segédprogram

A LOAD segédprogrammal nagy mennyiségű adathalmazt vihetünk be egy adatbázis táblájába. Ha a tábla nem üres, akkor egy opcióval adhatjuk meg, hogy a LOAD segédprogram a meglévő sorok megőrzésével adja hozzá a táblához az új sorokat, vagy az új betöltésből származó adatokkal írja felül a tábla tartalmát.
Amikor az adatbázis-adminisztrátor adatokat tölt be az adatbázisba, úgy kell megvalósítania a betöltést, hogy mérlegeli a következőket: a betöltés újraindíthatósága, triggerek, adatbázis-megszorítások, jogok.
A *betöltés újraindíthatósága*: A betöltés külső, a betöltő folyamaton kívüli hiba esetén leállhat. Ekkor a betöltést újra kell indítani. Ha az adatok nem megfelelőek, mert például nem olyan típusúak, mint amilyet a tábla vár, akkor a betöltése nem áll le, hanem a számára hibás adatokat nem tölti be, és valamilyen módon megjelöli őket. Ha a betöltő folyamaton kívüli hiba történik a betöltés során, fontos kérdés az, hogy a betöltést elölről kell-e kezdeni, vagy a betöltött adatokat megtartva újra lehet-e indítani a betöltést. Időigényesebb megvalósítani úgy a betöltést, hogy az újraindítható legyen. Ha a betöltés leáll, az újraindítással a betöltési idő lerövidül. Ahhoz, hogy a betöltő folyamat újraindítható legyen, az adatbázis-adminisztrátornak biztosítania kell, hogy a betöltésre használt állományok helye meg legyen határozva, és hogy a LOAD segédprogram onnan legyen folytatható, ahol abbamaradt. Ha a LOAD segédprogram nem indítható újra, és olyan hiba lép fel, amely megállítja a betöltési folyamatot, akkor az adatbázis-adminisztrátornak a következő két lehetőségből kell választania:

- meghatározza, hogy mely adatok lettek már betöltve, és törli ezeket az input adatok közül.
- a már betöltött adatokat törli, és az egész folyamatot újra kezdi. Ez a feladat sokkal bonyolultabb, ha már létező táblabeli adatokhoz szeretnénk újabb sorokat hozzáadni, mert szelektív törlést igényel.

Triggerek: A legtöbb adatbázisban a LOAD segédprogram nem indítja el a triggereket, ami adatintegritási problémákhoz vezethet. Ha az adatbázis és az alkalmazás olyan triggerekre épül,

amelyek adatokat módosítanak vagy megszorításokat kényszerítenek ki, akkor a triggerek futása nélküli adatbetöltés az adatbázis állapotában károkat okozhat. Ha rendszeres adatbetöltés történik egy ilyen adatbázisba, akkor az adatbázis-adminisztrátornak vagy az alkalmazás fejlesztőknek olyan fejlesztői programokat vagy szkripteket kell fejleszteniük, amelyek szimulálják a triggerek működését.
Adatbázis-megszorítások: adatbázis-kezelő rendszertől és a LOAD segédprogramtól függ, hogy mi történik, ha olyan adatokat próbálunk betölteni, amelyek nem teljesítik az egyediségi megszorításokat vagy az ellenőrző megszorításokat. Néhány LOAD segédprogram olyan opciókat ajánl fel, amellyel megadhatjuk, hogy a program a megszorításokat figyelembe vegye-e vagy ne. Azonban ha a LOAD nem kényszeríti ki a megszorításokat, akkor az adatbázis-adminisztrátor olyan adatot is betölthet, amely nem felel meg a megszorításnak és így adatintegritási problémákat okoz. Ekkor valamilyen más módon kell a megszorításoknak nem megfelelő adatokat kiszűrni, és megfelelően kezelni. Néhány LOAD segédprogram egyáltalán nem ad az adatbázis-adminisztrátornak választási lehetőséget – vagy minden adatot betölt, tekintet nélkül a megszorításokra; vagy pedig automatikusan eldobja a nem megfelelő adatokat. Általában mindig jobb, ha a segédprogram rugalmasan kezelhető, még akkor is, ha ez az adatbázis-adminisztrátor számára több munkával jár. Az adatbázis-adminisztrátornak támogatnia kellene a megszorítások kikényszerítését, amennyiben a LOAD segédprogram ezt lehetővé teszi. Különben a felhasználónak kell a betöltés után az adatokat kijavítani, vagy az adatintegritás sérülésével együtt élni. Sokkal hatékonyabb az adatokon egyszer, betöltés közben végigmenni, mint kétszer feldolgozni őket – egyszer betölteni azután kijavítani.
Jogok: A LOAD segédprogram hívása előtt, az adatbázis-adminisztrátornak biztosítania kell, hogy minden folyamat és felhasználó, ami, illetve aki a LOAD segédprogramot használni akarja, rendelkezzen a betöltés működéséhez szükséges jogosultságokkal. Néhány adatbázis-kezelő rendszer rendelkezik olyan jogosultsággal, amelyet speciálisan a LOAD segédprogram részére lehet adni. A legtöbb adatbázis-kezelő rendszer azonban csak az adattábla tulajdonosának, illetve az adminisztrátoroknak engedélyezi az adatbetöltést.

Az input állomány leírása

A LOAD segédprogramnak az adatok betöltéséhez szüksége van egy input állományra, amely a betöltendő adatokat tartalmazza. Az adatbázis-adminisztrátornak meg kell adnia az input állomány szerkezetét, hogy a LOAD segédprogram az input állomány soraiban található adatokat az adatbázisnak megfelelő formátumra konvertálja. Az input állomány szerkezetének a leírásában meg kell határozni az egyes oszlopok adattípusát, illetve az állományon belül az egyes sorokban lévő adatok kezdő és a befejező pozícióit. A LOAD segédprogramok általában tudnak más, speciális formátumú, például pontosvesszővel tagolt állományokat is betölteni. A LOAD segédprogram az UNLOAD segédprogrammal készített állományokat is be tudja tölteni, amelyekhez az UNLOAD segédprogram automatikusan elkészíti az adatállomány leírását is. Így megkönnyíti az adatbázis-adminisztrátornak a munkáját.
A LOAD segédprogramnak képesnek kell lenni a NULL értékek kezelésére, amelyet ugyancsak az input állomány leírásában szabályozhatunk.
A táblákba töltendő lebegőpontos adatok formátumáról a LOAD segédprogram általában valamilyen speciális információt vár.
A LOAD segédprogramok számos lehetőséget adnak az adatbázis-adminisztrátornak, hogy a betöltést vezérelhesse. Például lehet olyan záradékokat használni, amellyel megadhatjuk az adat alapértelmezett értékét, vagy amely olyan feltételeket tartalmaz, amely alapján bizonyos rekordokat a LOAD segédprogram nem tölt be az adatbázisba. Az adatbázis-adminisztrátornak a LOAD segédprogram minden használt paraméterét ismernie kell, hogy a betöltő folyamatot megfelelően vezérelni tudja.
Az 5-1. ábrán példát láthatunk a LOAD segédprogram input állományának a leírására.

Konkrétabban ez egy Oracle SQL*Loader vezérlő állományának a tartalma.

```
LOAD DATA
INFILE 'sample.dat'
BADFILE 'sample.bad'
DISCARDFILE 'sample.dsc'
APPEND
INTO TABLE emp
TRAILING NULLCOLS
(hiredate SYSDATE,
  deptno POSITION(1:2)  INTEGER EXTERNAL(2)
         NULLIF deptno=BLANKS,
   job   POSITION(7:14)  CHAR  TERMINATED BY WHITESPACE
         NULLIF job=BLANKS  "UPPER(:job)",
   mgr   POSITION(28:31) INTEGER EXTERNAL
         TERMINATED BY WHITESPACE, NULLIF mgr=BLANKS,
   ename POSITION(34:41) CHAR
         TERMINATED BY WHITESPACE  "UPPER(:ename)",
   empno POSITION(45) INTEGER EXTERNAL
         TERMINATED BY WHITESPACE,
   sal   POSITION(51) CHAR  TERMINATED BY WHITESPACE
         "TO_NUMBER(:sal,'$99,999.99')",
   comm  INTEGER EXTERNAL  ENCLOSED BY '(' AND '%'
         ":comm * 100"
)
```

5-1. ábra LOAD segédprogram input állományának a leírása

Hatékony betöltés

Az adatbázis-adminisztrátornak a betöltés hatékonyságánál figyelembe kell vennie az indexeket. Meg kell tudnia becsülni, hogy melyik lesz a hatékonyabb: az adatok betöltése előtt létrehozni az indexeket, így a betöltéssel egyszerre épül fel az index; vagy először betölteni az adatokat, és az indexeket csak azután felépíteni. Az, hogy melyik a hatékonyabb, függ az adatbázis-kezelő rendszertől, a tábla méretétől, az index típusától.
A LOAD segédprogram a betöltési folyamat alatt az adatokat egyik típusból a másikba konvertálhatja. A konverzió számolásigényes, amely leterheli a processzort, emiatt kerüljük el ezeket a konverziókat, ha lehet.
A LOAD segédprogram képes lehet egy tábla különböző partícióiba párhuzamosan, különböző input állományokból adatokat betölteni. Nagy mennyiségű adat betöltése esetén érdemes ezt a lehetőséget kihasználni. A párhuzamosan futó adatbetöltések növelhetik a processzorigényt, azonban csökkentik a betöltő folyamatok futására fordított összidőt.
Ha több táblába töltünk be adatokat, érdemes utánanézni, hogy a LOAD segédprogram milyen feltételek mellett futtatható párhuzamosan. A párhuzamos munkaterhelés ésszerű ütemezésével a LOAD segédprogram képes párhuzamosan működni. Például, különböző adattáblákba, melyek különböző adatbázisokban vagy állománycsoportokban vannak, párhuzamosan tölthetünk be adatokat. Az adatbázis-adminisztrátornak meg kell ismernie a LOAD segédprogram képességeit és korlátait, hogy ennek megfelelően tervezhesse a konkurens betöltéseket.
A LOAD segédprogram opciót biztosíthat a naplózás kikapcsolására. Ha a naplózást kikapcsoljuk, a betöltési folyamat felgyorsul, és minimalizáljuk a napló túlcsordulását. Viszont az adatbázis-adminisztrátornak ügyelnie kell a betöltött adatok helyreállíthatóságára, emiatt a betöltés befejezése után az adatokról valamilyen mentést kell készítenie.
Általában a LOAD segédprogrammal sokkal hatékonyabb lehet egy tábla kezdő adatait betölteni, mint többszörös SQL INSERT paranccsal vagy alkalmazói programmal végrehajtani ugyanazt a feladatot. Ez csökkenti az adatbázis-adminisztrátorok vagy a programozók munkáját, mert nem kell

megírni az INSERT utasításokat, vagy mert nem kell megírni, tesztelni, illetve a hibákat keresni az alkalmazói program logikájában. Helyette mindezt a jól megírt LOAD segédprogram tartalmazza, a feldolgozást pedig az adatbázis-adminisztrátor paraméterek és a segédprogram záradékainak segítségével tudja vezérelni. A LOAD segédprogram használatával kisebb lesz a hibázás lehetősége is.
Néhány adatbázis-kezelő rendszer esetén sokkal hatékonyabb a LOAD segédprogramot használni a tömeges törlés helyett. A tömeges törlés alatt az olyan SQL DELETE parancsot értünk, amelyhez nincs megadva WHERE feltétel. A LOAD segédprogram használata input állomány megadása nélkül ugyanezt eredményezi – minden sort töröl a megadott adattáblában. A LOAD nagy valószínűséggel hatékonyabb, főleg akkor, ha a naplózás ki van kapcsolva a betöltő folyamat alatt. Ebben az esetben azonban ne feledjük el, hogy a betöltés futása után – még akkor is, ha ez valójában törlés – mentést kell végezni.

Más alkalmazások futtatása a LOAD alatt

Néhány LOAD segédprogram a betöltött adatokat azonnal hozzáférhetővé teszi. Azaz a betöltés ugyan még nem fejeződött be, de a már betöltött adatokkal lehet dolgozni. Így az adatbázis-adminisztrátor már a betöltés folyamata alatt képmásolati mentést vagy adatbázis-statisztikát készíthet az aktuálisan betöltött adatokról. Az eljárás azért hatékony, mert az adatot egyszer olvassuk be a memóriába, de azt többször dolgozzuk fel (betöltés, képmásolat, statisztika stb.).

Az UNLOAD segédprogram

Az adattáblákban szereplő információkat gyakran kell mozgatni, vagy másolni különböző helyekre. Az UNLOAD segédprogram célja, hogy adatokat olvasson ki az adatbázisból és a kiolvasott adatokat egy output adatállományba írja.
Az UNLOAD segédprogram nélkül az adatbázis használóinak adatok kimentésére SQL SELECT utasításokat kellene írniuk egy interaktív SQL eszközben, vagy riportoló eszközben vagy alkalmazói programban. Ezeknél a módszereknél azonban sok a hibázási lehetőség és nagy mennyiségű adat esetén lassúak. Egy olyan alkalmazás fejlesztése, amely lekérdezésekhez output állományt hoz létre, túlságosan rugalmatlan és sok időt felemésztő feladat. A legtöbb adatbázis-kezelő rendszernek van UNLOAD segédprogramja, amely rugalmasan és gyorsan hajtja végre a nagy mennyiségű adatok mozgatását.
Az UNLOAD segédprogramot használhatjuk például,

- ha adatot viszünk át egy másik adatbázisba,
- ha egy adattáblából külső feldolgozás számára szekvenciális állományba mentünk,
- egy másik relációs adatbázis-kezelő rendszerbe vagy platformra mozgatjuk az adatot,
- tesztadatok generálására, amikor egy tábla sorainak a részhalmazát mentjük ki,
- objektumok újraszervezésénél, amikor az adatokat kimentjük, optimalizáljuk, majd újra betöltjük,
- adatbázis-objektumok törlésénél és újralétrehozásánál. Ezt adatbázissémák változása követelheti meg. Ha az objektumot töröljük, akkor a benne lévő adatok is elvesznek. Ennek megfelelően az adatokat ki kell menteni, mielőtt az adatbázis-objektum változik.

Az UNLOAD használata előtt az adatbázis-adminisztrátornak biztosítania kell, hogy minden folyamat és felhasználó, ami, illetve aki az UNLOAD segédprogramot futtatni akarja, rendelkezzen az ehhez megfelelő jogokkal. Ez általában azt a jogot jelenti, hogy az adatokat a felhasználó vagy folyamat olvashassa (SELECT) az adatbázisból. Azonban adatbázis-kezelő rendszertől és segédprogramtól függően esetleg további korlátozás is megkövetelhető.

Konkurencia

Az UNLOAD segédprogram paramétereitől és opcióitól függ, hogy azon a táblán, amelyből az

adatokat mentjük, végezhető-e párhuzamos tevékenység, és ha igen, milyen. Egyidejű olvasás lehetséges, mert az UNLOAD nem változtatja meg az adatokat. A legtöbb UNLOAD program esetén a felhasználó vezérelheti, hogy történhet-e párhuzamos adatmódosítás az adatok kimentése alatt. A kimentett állományba kerülő adatok konzisztenciája érdekében legjobb, ha nem engedjük a párhuzamos módosításokat az UNLOAD alatt.
Az adatok kimentése alatt szükséges lehet a párhuzamos segédprogramok futását letiltani. Például az UNLOAD alatt történő adatok betöltése, újraszervezése vagy helyreállítása megjósolhatatlan eredményeket okozna az UNLOAD segédprogram output állományában, mert ezek a segédprogramok változtatják az adatokat.

Adat kimentése egy képmásolati mentésből

Az UNLOAD segédprogramok képesek lehetnek képmásolati mentésből adatokat kinyerni. Ez a képesség növeli az egyidejű adatelérést. Ha képmásolati mentésből nyerünk ki adatok az UNLOAD segédprogrammal, az az élő adatra nincs hatással, nem kell zárolni az élő adatot. Mivel az UNLOAD segédprogram a képmásolati mentésből olvassa az adatokat, az élő adatokon futó alkalmazások teljesítményére és elérhetőségére nincs hatással a párhuzamos UNLOAD művelet.
Kérdéses viszont, hogy a kinyert adatok mennyire frissek. Ha az adattáblán a képmásolati mentés készítése után módosítás, beszúrás vagy törlés történt, ezek a módosítások nem jelennek meg a kinyert adatokon, mivel a képmásolati mentésen sem hajtódtak végre.

A LOAD paraméterek generálása

Az UNLOAD segédprogramnak általában van egy olyan opciója, amellyel olyan vezérlő utasításokat lehet generálni, amelyek a LOAD segédprogram használatához szükségesek. Azaz a LOAD segédprogram input állományának a leírását generálja. Így a kimentett adatokat könnyedén vissza lehet tölteni. Ezzel az opcióval sok idő takarítható meg, mert nem az adatbázis-adminisztrátornak kell a LOAD segédprogramhoz az input állomány leírását és az egyéb vezérlő utasításokat kézzel létrehoznia. Ha az adatokat egy másik adattáblába fogjuk betölteni, akkor is hasznos, ha legeneráljuk a LOAD segédprogramhoz a vezérlőutasításokat az UNLOAD művelet során. Egy legenerált LOAD vezérlőutasítás módosítása általában könnyebb, mint rögtönözni egy újat.

Adatkódolási séma

Az UNLOAD segédprogram használatánál általában meghatározhatjuk a kódolási sémát a kimenteni kívánt adathoz. Az adatbázis-kezelő rendszertől és az UNLOAD segédprogramtól függően különböző adatkódolási sémák közül válogathatunk a kimentett adatokhoz. A legtöbb általános kódolási sémát (EBCDIC, ASCII vagy UNICODE) azonban minden UNLOAD segédprogram ismeri. A kódolási sémát minden esetben javasolt megadni a konverziós problémák kezelése miatt.

Lebegőpontos adat

Mint ahogy a LOAD segédprogramnál is, itt is figyelemmel kell kísérni a lebegőpontos adatok kimentését. Lebegőpontos adatokat kimentésekor speciális paramétereket kell megadni a formátum meghatározásához, amelyben a lebegőpontos adatot menteni kívánjuk.

Az adatkimentés korlátozása

Néha az adatbázis-adminisztrátor csak az adatok egy részét menti ki, vagyis az UNLOAD segédprogram output állománya csak az adattábla sorainak egy részhalmazát fogja tartalmazni. Többféle oka lehet annak, ami miatt elég a sorok egy részhalmazának kimentése. Például tesztadatok generálása, vagy bizonyos sorok vizsgálata.

A legtöbb UNLOAD segédprogramnak van olyan opciója, amellyel megadhatunk valamilyen feltételt, hogy a tábla mely sorait kívánjuk menteni. Általában három lehetőség közül választhatunk: LIMIT, SAMPLE, és WHEN.
A LIMIT paraméter arra szolgál, hogy korlátozza az UNLOAD segédprogram által kimentendő sorok számát. Egy „LIMIT 200” paraméter megadása esetén az UNLOAD segédprogram a 200-adik kimentett sor után megáll.
A SAMPLE paraméter arra szolgál, hogy az adattáblából egy mintát nyerjünk ki. A SAMPLE paraméter után általában egy százalékos értéket kell megadni, amely a mintaként szolgáló sorok százalékát határozza meg. Például a következő paraméter azt mutatja, hogy a tábla sorainak 25,75%-a kimentésre vár: SAMPLE 25.75.
A WHEN záradék azt határozza meg, hogy csak bizonyos, a WHEN kulcsszó után szereplő feltételnek megfelelő adatok szerepeljenek a kimentett adatok között. Például a következő feltétel azt határozza meg, hogy csak azok a sorok kerülnek kimentésre, melyeknél a SALARY nagyobb, mint 50000: WHEN (SALARY>50000).
Általában az UNLOAD segédprogram tudja rendezi az adatokat, ha megadjuk az ORDER BY záradékot. Így a kimentett adatok rendezve kerülnek az állományba.
Ezeknek az opcióknak a pontos szintaktikája adatbázis-kezelő rendszerről adatbázis-kezelő rendszerre és alkalmazásról alkalmazásra változhat, de az általános koncepció ugyanaz. Az adatbázis-adminisztrátor nagyon rugalmasan készíthet külső adatállományokat az UNLOAD segédprogram segítségével.

Adatkimentés nézetekből

A legtöbb UNLOAD segédprogram nem csak adattáblából, hanem nézetből is tud adatkimentést végezni. Nézet létrehozásával lehetőségünk van egyszerre több tábla adatait egy állományba kimenteni.

Alkalmazás-tesztadatok kezelése

Alkalmazói programok teszteléséhez adatok konzisztens együttesét kezelhetjük a LOAD és az UNLOAD segédprogramok segítségével. Olyan adatállomány létrehozásával, amelyet a programozók az egyes programtesztek előtt betölthetnek, a fejlesztők biztosak lehetnek abban, hogy az egyes programfutáskor ugyanazokat az adatokat használják. Az, hogy a programok ugyanazon az input adatokon fussanak, kritikus a hibák felfedezésében, nyomon követésében és abban, hogy megbizonyosodjunk a programkód helyességéről. Az adatbázis-adminisztrátor elkészíti a megfelelő LOAD és UNLOAD szkripteket, és aztán átadja használatra az alkalmazásfejlesztő csoportnak. Az UNLOAD segédprogrammal az adatbázis-adminisztrátor vagy a programozók a betölthető adatállományokat hozhatják létre valamilyen más, nem a tesztelésre szolgáló adatbázisból történő adatkimentéssel. A LOAD segédprogrammal, pedig betölthetik az adatokat a tesztadatbázis tábláiba.

Export és import

A LOAD és UNLOAD segédprogramok egyes adatbázis-kezelő rendszerekben export és import néven is megjelenhetnek. Több adatbázis-kezelő rendszer rendelkezik mindkét segédprogrammal. Ekkor a különbség általában az, hogy a LOAD és az UNLOAD szöveges, az export és az import segédprogramok bináris formában kezelik az adatokat.

Nagy mennyiségű adat mozgatása

A LOAD és az UNLOAD segédprogramok kombinációja a legelterjedtebb módszer, amit az adatbázis-adminisztrátorok nagy mennyiségű adat mozgatására használnak. De vannak más módszerek is nagy tömegű adat mozgatására.

ETL szoftver

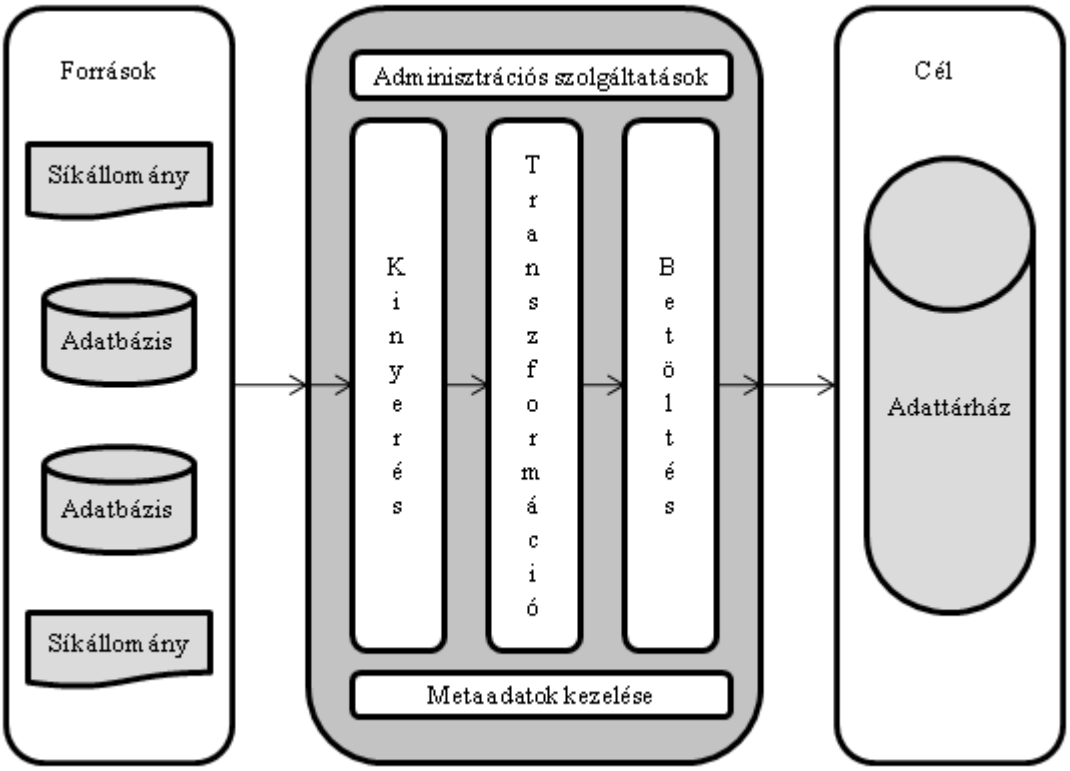
Az ETL (extract, transform, load) egy olyan szoftvertípus, amely ugyancsak nagy mennyiségű adat mozgatására szolgál. Részei: kinyerés (extract), transzformálás (transform) és a betöltés (load). Az ETL szoftver általában az adatokat adatforrásokból vagy adatbázisokból adattárházakba vagy adatpiacokba tölti.

Az ETL használatával az adatbázis-adminisztrátor automatizálhatja a különféle heterogén forrásokból származó adatok kinyerését. Az ETL szoftvert be lehet úgy állítani, hogy felismerje és kinyerje az adatokat különböző forrásokból, mint például síkállományokból (szöveges állományok, csv állományok – vesszővel tagolt mezőket tartalmazó szöveges állomány, xml állományok), relációs adatbázisokból, mint az ORACLE, MS SQL Server, és a DB2 adatbázisok, melyek különböző platformokon futnak, vagy egyéb külső adat forrásokból.

Ha az ETL szoftver kinyerte az adatokat, akkor a legtöbb esetben az adatot transzformálni kell, mielőtt a céladatbázisba kerül. Az ETL szoftver könnyedén automatizálja ezeket a transzformációkat. Előfordulhat, hogy kódolt adatot szeretnénk transzformálni, mivel a kódok helyett értelmes értékeket szeretnénk a céladatbázisban látni, mint például az „1”, „2”, „3” értékek helyett a „Házas”, „Egyedülálló” és „Elvált”. Az ETL sokféle transzformációra képes, az egyszerűtől (az adattípusok megváltoztatása) a bonyolultig (numerikus adatok aggregálása).

Miután az adatokat az ETL szoftver transzformálta, betölti őket a céladattárházba. Az ETL szoftver sokkal rugalmasabban kezeli az ilyen típusú, nagy mennyiségű adatmozgatást, mint az egyszerű LOAD és UNLOAD segédprogramok.

Az 5-2. ábra az ETL szoftver munkáját szemlélteti.



5-2. ábra ETL szoftver

Replikáció és propagáció

Az adatot replikáljuk, ha egy adattárat másolunk egy vagy több adattárba. Az adattárak lehetnek ugyanazon a gépen, vagy más gépen is. A replikáció során másolhatunk teljes adatbázist, táblákat, vagy táblák sorainak és oszlopainak a részhalmazát is. A replikációt be lehet állítani úgy, hogy rendszeres időközönként automatikusan frissüljenek a másolt adatok.
A propagáció csak a megváltozott adatokat migrálja. A propagációt meg lehet valósítani úgy, hogy a tranzakciós naplót vizsgáljuk és az adatváltozásokat alkalmazzuk a másik adattárban. Egy adattárház kezdeti betöltése végrehajtható replikációval, majd az azt követő változások megvalósíthatóak replikációval (ha az adatok nagyon dinamikusak) vagy propagációval.

Üzenetküldő szoftverek

Az üzenetküldő szoftverek, vagy más néven üzenetsorba-állító szoftverek egy másik lehetőségét adják az adatmozgatásnak. Egy üzenetsor használatával az adatokat egy alkalmazás vagy folyamat behelyezi a sorba; és az adatokat egy másik alkalmazás vagy folyamat olvassa a sorból.
Az üzenetküldő szoftverek olyan API biztosításával dolgoznak, melyeknek a segítségével lehet a sorból olvasni és a sorba írni a formázott üzeneteket. Egy alkalmazás bármilyen, a szoftver által támogatott platformról képes a sorból üzeneteket olvasni, illetve a sorba írni.
Az üzenetküldő szoftvereknek az adatmozgatást tekintve abban van a legnagyobb előnye, hogy egyszerű, heterogén, bármihez hozzácsatlakoztatható kapcsolatokat hozhatunk létre velük. Az üzenetküldő szoftver segítségével a cég aszinkron módon tud az elszeparált adatszigeteket között kapcsolatot létesíteni. Az aszinkron azt jelent, hogy az alkalmazások akkor is kommunikálhatnak egymással, ha nem egy időben futnak.

Más módszerek

Sok más módszer is létezik az adatok mozgatására – a legegyszerűbbtől, amikor egy táblaszerkesztő eszközzel kijelölünk és kimásolunk adatokat; a összetettebbekig, mint olyan alkalmazás fejlesztése, amely olvassa az adatbázist, és külső állományba vagy közvetlenül egy másik adatbázisba ír.
Néhány adatbázis-kezelő rendszer további beépített módszereket is biztosít az adatok másolására és mozgatására.

Összefoglalás

A fejezetben megnéztük, hogyan lehet hatékony eszközökkel nagy mennyiségű adatot az adatbázisba betölteni vagy az adatbázisból kimenteni.
Elsőként a LOAD segédprogrammal ismerkedtünk meg, amellyel az adatbázisba betölteni lehet adatokat. Megnéztük, hogy a betöltéshez az adatbázis-adminisztrátornak milyen tényezőket kell mérlegelnie: a betöltés újraindíthatóságát, az adatok rendezését, a triggereket, az adatbázis-megszorításokat, és a jogokat. A betöltéshez nem csak az input állományra van szüksége a LOAD segédprogramnak, hanem az input állomány leírására is. Megnéztük, hogy ez az állomány milyen információkat tartalmazhat. Megnéztük, hogy a betöltést hogyan tehetjük hatékonyabbá, azaz azt, hogy figyelembe kell venni az indexeket, az adatkonverziókat, a párhuzamos betöltés ugyanazon tábla más partícióiba vagy más táblákba, a naplózást. Az LOAD segédprogram hatékonyabban valósíthatja meg a kezdő adatok beszúrását egy táblába, vagy a tömeges törlést egy táblából. Megvizsgáltuk azt is, hogy a LOAD segédprogram és más alkalmazások egyidejű futása milyen hatással lehet a másikra.
Az UNLOAD segédprogrammal adatokat menthetünk ki az adatbázisból egy output állományba. Példákat hoztunk rá, hogy mely esetekben érdemes használni az UNLOAD segédprogramot. Megvizsgáltuk, hogy azon a táblán, amelyből az UNLOAD segédprogram éppen adatokat ment ki, végezhető-e más párhuzamos művelet, illetve azok hatását vizsgáltuk. Adatokat nem csak a működő

adatbázisból lehet kimenteni, hanem képmásolati mentésből is. Megvizsgáltuk ennek az előnyeit és a hátrányait. Az UNLOAD segédprogram nem csak az output állományt képes előállítani, hanem annak a leírását is. Ez azért hasznos, mert a LOAD segédprogram ezt a leírást azonnal tudja használni, mint az input állomány leírását. Beszéltünk arról a lehetőségről, hogy különböző adatkódolási sémákat határozhatunk meg a menteni kívánt adatokhoz, illetve a lebegőpontos adatok kimentésének a módjáról. Az UNLOAD segédprogram nem csak egy adattábla teljes tartalmát tudja egy output állományba menteni, hanem egy nézet tartalmát is, illetve a kimentett sorokra tehetünk különböző megszorításokat, mint LIMIT, SAMPLE, WHEN.
A LOAD és az UNLOAD segédprogramok segítségével könnyedén előállíthatja az adatbázis-adminisztrátor az alkalmazások teszteléséhez szükséges adatokat.
Nagy mennyiségű adatok mozgatására más technikákkal is megismerkedtünk, mint az ETL szoftver, replikáció és propagáció, illetve az üzenetküldő szoftverek.

Ellenőrző kérdések

1. Mire szolgál a LOAD segédprogram?
2. Mi történik, ha olyan táblába töltünk be adatokat, amely nem üres?
3. Az adatbázis-adminisztrátornak milyen tényezőket kell figyelembe vennie a betöltés megvalósításakor?
4. Milyen állományokra van szükség a betöltéshez?
5. Mit tartalmaz az input állomány leírása?
6. Hogyan lehet hatékonyabbá tenni a betöltést?
7. Lehet-e más alkalmazásokat futtatni a betöltés alatt?
8. Mire szolgál az UNLOAD segédprogram?
9. Az adatkimentéssel párhuzamosan végezhető-e valamilyen tevékenység azon a táblán, amelyből adatot mentünk ki? Ha igen, milyen? Ha nem, miért nem?
10. Mi az előnye és a hátránya annak, ha a képmásolati mentéseket arra használjuk, hogy például egy tábla adatait egy másik állományba mentsük?
11. Mit jelentenek adatkimentésnél a LIMIT, SAMPLE, WHEN opciók?
12. Tesztelésnél hogyan lehet használni a LOAD és az UNLOAD segédprogramot?
13. Mi az az ETL szoftver?
14. Mit jelent a replikáció és a propagáció?
15. Mi az az üzenetküldő szoftver?

6. fejezet – Adatok elosztása

Sok esetben nem elég az, ha az adatokat az egyik helyről a másikra csak egyszerűen átmozgatjuk. Létezhetnek olyan cégek, amelyeknek több különböző telephelyük van, és ezeken a telephelyeken különböző adatbázisokban tárolják az adatokat. A telephelyek adatait azonban nem csak a helyi felhasználók szeretnék használni, hanem más telephelyek alkalmazottai is. Az adatok egyszerű átmozgatása már nem biztos, hogy elegendő, mert például az adatok túl gyakran változnak vagy több telephelyről is módosítani szeretnék. Az ilyen esetekben elosztott-adatbázis megoldásra van szükség.

Elosztott adatbázisok

Az elosztott-adatbázis megoldások teszik lehetővé, hogy egy logikai egységet alkotó adatok fizikailag teljesen különböző helyeken, különböző adatbázisokban legyenek tárolva, amelyeket talán különböző adatbázis-kezelő rendszerek kezelnek. Így állandó elérést lehet biztosítani a logikailag összefüggő, de különböző csomópontokon tárolt, fizikailag elosztott adatokhoz. Például egy cég a kiskereskedelmi hálózatát megvalósíthatja elosztott adatbázissal. Minden kiskereskedéshez saját adatbázis tartozik, és a központban kap helyet a központi adatbázis. Ha a gépek hálózati kapcsolatban vannak és kihasználjuk az adatbázis-kezelő rendszernek az elosztott adatok elérését támogató képességeit, akkor bármely helyen lévő adat bárhonnan elérhető, és módosítható. Meghatározhatjuk, hogy melyik helyről frissíthetik, olvashatják, érhetik el az adott adatbázist.

Nem nevezzük elosztott adatbázisnak azt, amikor egymástól függetlenül működő adatbázisokat átmenetileg, adatmozgatás céljából összekapcsolunk, mert ezek az összekapcsolt adatbázisok nem alkotnak logikai egységet. Az a szituáció sem tekinthető elosztott adatbázisnak, amikor csak a metaadatokat osztjuk el. Maga az adatbázis ettől még egységes egészként kezelendő.

Az elosztott adatbázis esetén lehet, hogy az egyes csomópontok egyenrangúak, vagy lehet, hogy van egy kitüntetett központi adatbázis-szerver, amelynek a kommunikáció koordinálásában, a jogosultságok kezelésében és az adatok megfelelő elosztásában van szerepe.

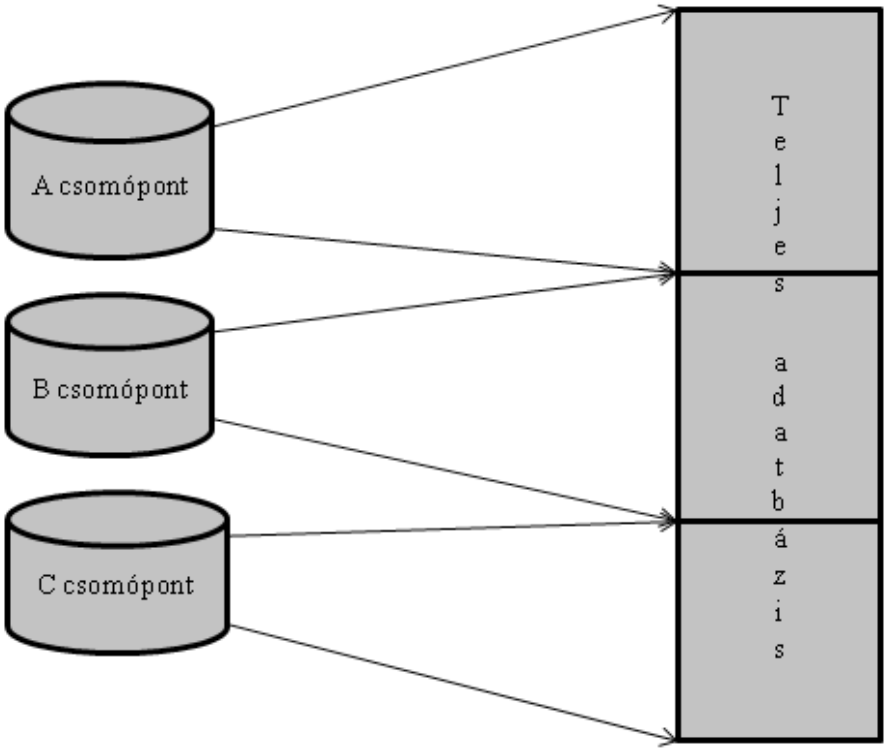
Egy elosztott adatbázis létrehozható egyetlen adatbázisként, vagy több adatbázisként is. Egyetlen adatbázis felállításakor az egyetlen adatbázis-kezelő rendszernek a folyamatai vezérelnek minden adatelérést. Az egyes csomópontokba a különböző adatállományok vannak elhelyezve. Több adatbázis felállításánál azonban, több független adatbázis-kezelő rendszerbeli folyamat van, amelyek a saját adataik elérését vezérlik.

Az elosztott adatbázisok lehetnek heterogének vagy homogének. Heterogén elosztott adatbázis esetén különböző adatbázis-kezelő rendszerek kezelik a különböző csomópontokban lévő adatokat. Ekkor egy olyan szoftvereszközre van szükség, amely segíti azt, hogy a különböző típusú adatbázis-kezelő rendszerek kommunikálni tudjanak egymással. A homogén elosztott adatbázis esetén minden csomópontban ugyanaz az adatbázis-kezelő rendszer. Az adatbázis-kezelő rendszerek általában támogatják a különböző csomópontokban lévő adatbázisok közti kommunikációt.

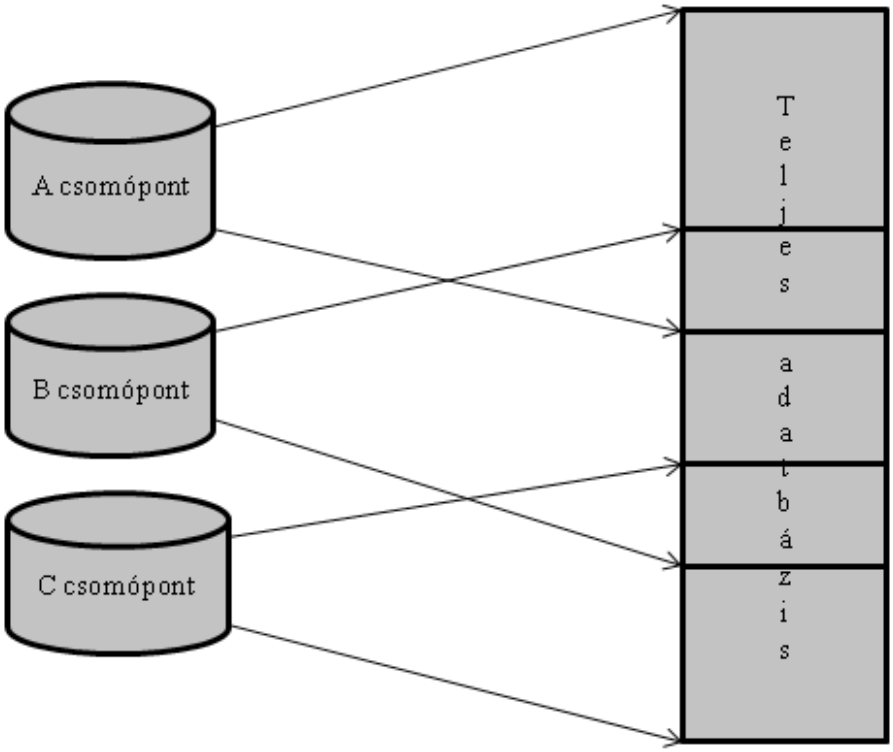
Az elosztott rendszerek esetében fontos kérdés az, hogy az egyes csomópontok adatai más csomópontokban vannak-e replikálva. Vagy fogalmazhatunk másképp: az adatokat szétválasztották valamilyen szempont szerint, vagy az egyes csomópontok adatai átfedik egymást. A szétválasztott adatok esetén az egyes csomópontokban csak annyi redundancia a megengedett, amennyi feltétlenül szükséges, például az egyik csomópont elsődleges kulcsértékeit a másik csomópont külső kulcsként használja. Ha az egyes csomópontok adatai átfedik egymást, az az jelenti, hogy ettől sokkal nagyobb mértékű redundancia is megengedett, mint amikor egy tábla teljes tartalmát két csomóponton tároljuk.

A 6-1-es ábrán olyan elosztott adatbázist láthatunk, amelyben az adatok szét vannak választva, míg

a 6-2-es ábrán olyat, amelyben az egyes csomópontok adatai átfedik egymást.



6-1. ábra Szétválasztott adatok



6-2. ábra Átfedő adatok

Ha az elosztott adatbázisunkban az adatokat szétválasztottuk, azaz nincs redundancia, és ha az egyik csomópont esetén hiba történik, akkor annak a csomópontnak az adatai nem lesznek elérhetőek. Ha átfedő adataink vannak, és egy csomópont minden adatát meg lehet találni egy másik csomóponton is, és ha egy csomópontban hiba történik, akkor az elérhetőség nem sérül, hiszen az adatokkal lehet tovább dolgozni. Ahhoz, hogy az adatokkal úgy lehessen tovább dolgozni, mintha nem történt volna meg a hiba, szükségesek más technikák is, amit itt nem részletezünk. A klaszterezés (vagy fürtözés) az elosztott adatbázisoknak ezt a tulajdonságát használja ki, amikor a 99,999%-os elérhetőséget kell biztosítania.
Az elosztott adatbázisokat a következőkkel szokták jellemezni:

- Autonómia: azt határozza meg, hogy a különböző csomópontokon lévő elosztott adatbázis-implementációk milyen mértékben működhetnek egymástól függetlenül. Egy csomópont autonóm, ha az ott tárolt adatokat a többi csomóponttól függetlenül lehet kezelni, adminisztrálni.
- Átlátszóság: arra utal, hogy az adatok helye a felhasználóktól, illetve az alkalmazásoktól mennyire vannak védve. Azt mondjuk, hogy az adatbázis átlátszó, ha az adatok használója az adatbázist egy egységként látja, és nem kell tudnia, hogy a számára szükséges adatok hol vannak elhelyezve. Átlátszó elosztott rendszerben a fejlesztőknek nem kell ismerniük az adatok helyét. Nem átlátszó rendszerben a fejlesztőnek explicit módon kell megadnia egy kapcsolatot, ha egy adatbázis-csomóponton lévő adatokat szeretné elérni. Az adatbázis-adminisztrátor feladata, hogy a fejlesztőknek megadja, hogyan érhetik el az adatokat a nem átlátszó elosztott adatbázisban. A mai technológiák mellett az átlátszóság gyakran a köztes réteg feladata. Ilyenkor az adatbázis-adminisztrátor felelős az adatbázis és a köztes réteg munkájának az összehangolásáért.

Az adatbázis-adminisztrátor elosztott adatbázisok esetén különböző szintű autonómiát és átlátszóságot tud megvalósítani egy adatbázis-kezelő rendszer segítségével. Természetesen a megvalósítás függ az adatbázis-kezelő rendszer képességeitől is. Ahhoz, hogy az adatbázis-adminisztrátor ezeknek a jellemzőknek, illetve más tényezőknek megfelelően alakítsa ki az elosztott adatbázist, pontosan ismernie kell az adatbázis-kezelő rendszer képességeit.

Elosztott rendszer tervezése és létrehozása

Az adatbázis-adminisztrátor számára az egyik legbonyolultabb feladat egy elosztott rendszer implementációban a tervezés és a létrehozás. Az adatbázis-adminisztrátor első feladata megismerni az adatbázis-kezelő rendszer lehetőségeit az elosztott rendszerekkel kapcsolatban. Olyan kérdéseket kell feltennie, mint: milyen más szoftver szükséges az elosztott rendszer támogatásához, támogatja-e az adatbázis-kezelő rendszer a kétfázisú COMMIT-ot, milyen hálózati protokollok szükségesek, stb.
Az adatbázis-adminisztrátornak tudnia kell, hogy az elosztott adatbázis minél több csomópontra van bontva, annál nehezebb lesz a kezelése és az adminisztrálása. Karbantartás szempontjából több kisebb gépen futó adatbázis-kezelő rendszer adminisztrációja nagyon bonyolult. Egy tényleges elosztott adatbázis-implementáció esetén az adat gyakran több szerveren tárolódik, és néha kliens csomópontokon is. Karbantartás szempontjából az is fontos, hogy az egyes csomópontok mennyire lesznek függetlenek. Az autonóm csomópontok karbantartása könnyebben megoldható, mint az egymástól nagy mértékben függő csomópontok karbantartása.
Az adatbázis-adminisztrátornak együtt kell dolgoznia a hálózati adminisztrátorokkal, hogy a cég hálózata megfelelően legyen konfigurálva az elosztott adatbázis támogatásához. Az adatbázis-kezelő rendszerhez be kell állítani a konfigurációs paramétereket, amelyek lehetővé teszik az elosztást, és megadják a távoli adatbázisok hálózati helyeit.
Az adatbázis-adminisztrátornak meg kell terveznie, hogy melyik adat hova kerül. A tervezéshez

néhány irányelvet az adatbázis-kezelő rendszer lehetőségei határoznak meg, de vannak adatbázis-kezelő rendszertől független irányelvek is. A legfontosabb ilyen irányelv az, hogy az adatot mindig azon a csomóponton tároljuk, ahol a legtöbbet használják. Azaz a költséges hálózati forgalmat a lehető legnagyobb mértékben minimalizálni kell. Cél, hogy az adatokat gyorsan el lehessen érni, könnyen lehessen kezelni, és az, hogy az adatintegritást magas fokon fenn lehessen tartani. Az adatintegritás fenntartásában az egyik kihívás a redundáns adatok kezelése, a másik kihívás pedig az elosztott tranzakciók megfelelő lebonyolítása.
Az adatbázis-adminisztrátornak az alkalmazásfejlesztést támogatnia kell. Az alkalmazások tervezésénél ugyancsak figyelembe kell venni, hogy a hálózati forgalom minimalizálásának érdekében lehetőleg mindig azzal az adattal dolgozzunk, amelyik a felhasználóhoz a legközelebb van. Az adatbázis-adminisztrátornak útmutatást kell adnia az alkalmazásfejlesztő csoport számára az elosztott adatbázishoz, amelyben leírja, hogy az adat hol van tárolva, mi az elosztott adat elérésének teljesítménybeli hatása és hogyan optimalizálható az elosztott elérés. Az alkalmazásfejlesztőknek pedig törekedniük kell arra, hogy a hálózati forgalmat azzal is minimalizálják, hogy csak a valóban szükséges adatokkal dolgoznak. Azaz az adatbázis kérések ne tartalmazzanak több sort és több oszlopot, mint amelyekre az alkalmazásoknak szüksége van.

Adatelosztási szabványok

Két általános szabvány létezik az elosztott adatbázisokhoz, melyeket a legtöbb adatbázis-kezelő rendszer támogat: a DRDA és az RDA. Céljaikat tekintve a két szabvány hasonló.
Elosztott Relációs Adatbázis Architektúra (Distributed Relational Database Architecture), vagy DRDA, az IBM protokollja. A DRDA az elosztott csomópontok közötti kommunikáció koordinálására ad módszereket. A módszerek segítségével az alkalmazások több csomóponton lévő több távoli táblákat érnek el úgy, hogy a végfelhasználó számára úgy tűnik, mintha azok egyetlen logikai egységet alkotnának.
Az architektúra és az implementáció között különbséget kell tenni. A DRDA az architektúrát írja le az elosztott adatokhoz, és nem többet. Szabályokat definiál az elosztott adatok eléréséhez, de nem biztosítja az aktuális API-t, hogy az elérést megvalósítsa. A DRDA nem egy program, hanem szabványok gyűjteménye.
Távoli Adatbázis Elérés (Remote Database Access), vagy RDA, egy kommunikációs protokoll távoli adatbázisok eléréséhez. Az ISO és az ANSI fejlesztett ki. Az RDA úgy készült, hogy az SQL olyan részhalmazával dolgozzon, amely a legtöbb adatbázis-kezelő rendszerhez elérhető.
Az RDA egy távoli kapcsolatot hoz létre egy szerver és egy kliens között. Az alkalmazói program helyett tevékenykedő kliens egy olyan szerver-oldali folyamattal érintkezik, amely kontrollálja az adatátvitelt az adatbázisba, illetve az adatbázisból. Az RDA célja, hogy lehetővé tegye, hogy az alkalmazások heterogén adatbázisokkal és környezetekkel kapcsolódjanak össze.
Elosztott adatok elérésére a DRDA helyett az RDA egy jó alternatívát ad, ha az RDA alapján megvalósított átjáró (gateway) alkalmazást használunk. Egy ilyen alkalmazás legalább két komponenst tartalmaz – egyet a helyi oldalon, és minden távoli ponthoz egyet – amik egymással kommunikálva lehetővé teszik az adatelérést.

Elosztott adatok elérése

A távoli oldalon lévő adatok lekérdezésére, módosítására több különböző elérési mód létezik. Az egyes adatbázis-kezelő rendszerek ezek közül nem biztos, hogy mindegyiket támogatják. Ha az elosztott rendszerünk heterogén, akkor további korlátozások merülhetnek fel.
A legegyszerűbb módja egy elosztott adatbázis elérésének a távoli kérés (remote request). A távoli kérés egy tranzakción (munkaegységen) belül egyetlen távoli csomópontról egyetlen kérést tartalmaz. A távoli kérés lehetővé teszi a fejlesztő számára, hogy az egyik adatbázis-kezelő rendszerben hajtson végre olyan műveletet, amelynek egy másik adatbázis-kezelő rendszerben lesz

hatása. Ez a legegyszerűbb és a legkevésbé rugalmas módszere az elosztott elérés kódolásának.
Összetettebb típusú elosztott elérés a távoli munkaegység (remote unit of work). Ez akkor alkalmazható, ha egy alkalmazói programnak több helyről van szüksége adatra, de azokat nem kell egy tranzakción belül kérni. A programozónak tudnia kell, hol van az adat, és egy tranzakciót úgy kell megírnia, hogy az csak egy csomópontról kérjen adatokat. Azaz minden kérés egyetlen csomópontot érinthet, minden tranzakció (munkaegység) több SQL parancsot tartalmazhat, de minden tranzakció csak egyetlen csomópontról érhet el adatokat. Ily módon több SQL utasítás is megengedett egy tranzakcióban, de egy tranzakció csak egy adatbázis-kezelő rendszerből dolgozhat.
A következő típusú elosztott adatelérést elosztott munkaegységnek nevezzük (distributed unit of work). Ennél az elérésnél a tranzakciók (munkaegység) több csomóponton érhetnek el adatokat. Azaz több adatbázis-kezelő rendszer is elérhető egy tranzakción belül. Több SQL utasítás olvashatja és/vagy módosíthatja az adatokat több adatbázis-kezelő rendszerben, egyetlen tranzakción belül. Azonban egy SQL utasítás egyszerre csak egy csomóponton dolgozhat.
Az utolsó, és egyben legerősebb az elosztott elérési típusok közül az elosztott kérés (distributed request), ahol egyetlen SQL utasítás több csomóponton lévő adatokhoz egy időben férhet hozzá. Így egy SQL utasítás képes több adatbázis-kezelő rendszer elérésére. Egyetlen munkaegység több SQL utasítást tartalmazhat, akár elosztott, akár nem. Ez a legrugalmasabb szintje az elosztott adatbázisok elérésének.
A 6-3. ábra összefoglalja az adatok elérési módjait.

Elosztott elérés	SQL utasítások száma tranzakciónként	DBMS-ek száma tranzakciónként	DBMS-ek száma SQL utasításonként
távolikérés	1	1	1
távolimunkaegység	>1	1	1
elosztottmunkaegység	>1	>1	1
elosztottkérés	>1	>1	>1

6-3. ábra Az elosztott adatok elérésének szintjei

Kétfázisú COMMIT

Ha egy tranzakción belül különböző csomópontokon módosítunk adatokat, akkor az adatbázis-kezelő rendszernek biztosítania kell, hogy a módosítások egyetlen műveletként legyenek kezelve. Minden tranzakció véget ér vagy egy COMMIT utasítással vagy valamilyen ok miatt visszagörgetéssel. A COMMIT utasítás hatására a véglegesítés lehet sikeres vagy sikertelen. Elosztott adatbázisok esetén ennek ugyanígy kell lennie, azaz a tranzakcióban lévő minden műveletnek az eredményét alkalmazni kell az összes adatbázison, vagy a tranzakció minden műveletét vissza kell vonni, függetlenül attól, hogy melyik csomópont melyik adatbázisában történt a változás. Az elosztott rendszerekben a véglegesítést kétfázisú COMMIT segítségével valósíthatjuk meg, ahol az első fázis az előkészítés, a második pedig maga a véglegesítés.
Elosztott kétfázisú COMMIT segítségével egy alkalmazói program egyetlen tranzakción belül több adatbázis-kezelő rendszerben lévő adatot tud frissíteni. A kétfázisú COMMIT koordinálja a

különböző csomópontokon végbemenő COMMIT-okat. A kétfázisú COMMIT konzisztens kimenetet biztosít, garantálva az adatintegritás megőrzését a csomópontok között kommunikációs vagy rendszerhibák esetén is. Az egyik adatbázis-kezelő rendszer koordinálja a kétfázisú COMMIT-ot, míg a másik résztvevőként van jelen.
Az előkészítő szakaszban minden résztvevő felkészül egy COMMIT-ra. Minden résztvevő informálja a koordinátort, ha a naplórekordok sikeresen kiíródtak és jelzi, hogy kész a COMMIT végrehajtására. Ha minden résztvevő kész a COMMIT végrehajtására, a következő fázis, azaz a COMMIT végrehajtása megkezdődik. Ez a fázis úgy valósul meg, hogy a koordinátor és az alárendelt résztvevők sorozatosan kommunikálnak egymással. A COMMIT fázis alatt a siker még rendszerhiba esetén is feltételezett. Ha minden résztvevő a COMMIT folytatását választja, a siker az adatintegritás sérülése nélkül feltételezhető. Azonban ha egyetlen résztvevő nem képes a COMMIT végrehajtására, akkor a koordinátornak vissza kell állítania a változásokat az összes résztvevőnél.
Ha egy alkalmazás egy tranzakción belül olyan módosítást végez, amelynek több csomópontra van hatása, kétfázisú COMMIT-ot kell használni az adatintegritás biztosítása érdekében.

Elosztott rendszerek teljesítmény problémái

Egy elosztott adatbázis-környezetben a teljesítmény különösen problémás lehet. Mint a legtöbb olyan rendszernél, amely hálózatokra épül, itt is a hálózati adatforgalom teljesítménye jelenti a legnagyobb veszélyt a teljesítményre nézve. Minél több adatot szeretnénk a hálózaton átjuttatni, annál több probléma merül fel a hálózat teljesítményével.
A teljesítmény témakörével a Teljesítménykezelés, Rendszerteljesítmény és az Adatbázis-teljesítmény című fejezetekben foglalkozunk. A Teljesítménykezelés fejezetben meghatározzuk a teljesítményt befolyásoló öt tényezőt: munkaterhelés, áteresztőképesség, erőforrás, optimalizáció, versengés. Elosztott környezetben ezek közül külön kell foglalkoznunk az áteresztőképességgel. A Teljesítménykezelés című fejezetben részletesen kifejtjük, hogy az áteresztőképesség definiálja a számítógép teljes képességét az adatfeldolgozásra, vagyis az áteresztőképesség az az adott mennyiségű munka, ami egy időegység alatt elvégezhető. Áll az input/output műveletek sebességéből, a processzor sebességből, a számítógép párhuzamos képességeiből és az operációs rendszer, illetve a rendszerszoftver hatékonyságából.
Elosztott rendszerekben a kérést küldő és a kérést kiszolgáló csomópontok teljesítményét külön-külön is vizsgálni kell. A kérést kiszolgáló csomópontnak több kérést is ki kell egyszerre szolgálnia. Minél több kérést képes kiszolgálni, annál jobb az áteresztőképessége, annál jobb a teljesítménye. A kérést küldő oldal számára a reakcióidő a fontos. A reakcióidő az az időmennyiség, ami egy előre definiált munkafolyamat végrehajtásához szükséges. A reakcióidő sokkal hasznosabb jelző a végfelhasználó számára, mert ő az, aki az eredményre vár – és minél hosszabb a reakcióidő, annál tovább tart, míg a felhasználó elvégzi a feladatát.
Egy elosztott adatbáziskérés áteresztőképességének analizálásához, az adatbázis-adminisztrátornak az egész kérést teljesítő áteresztő láncot meg kell vizsgálnia. Ha csak egy komponens teljesítménye nem megfelelő, akkor az teljesítménybeli romláshoz vezethet. Egy kéréshez tartozó áteresztő láncba beletartozik minden egyes olyan hardver és szoftver, illetve minden olyan konfiguráció, amely a szolgáltatást szállítja és amelyen a kérésnek át kell haladni végfelhasználóhoz. Az áteresztő láncban a következők általában szerepelnek:

- A kérést küldő oldalon a számítógép hardvere, a helyi operációs rendszer, hálózati rendszer, és a helyi adatbázis-kezelő rendszer.
- A kérést küldő és a kiszolgáló oldal között lévő hálózati hardverek, kábelezés, gateway-ek, routerek, és a hubok.
- Bármilyen köztes termék vagy tranzakció feldolgozó rendszer, amit a kérést küldő vagy a kiszolgáló oldal használ.

- A kiszolgáló oldalon a számítógép hardvere, a helyi operációs rendszer, hálózati rendszer, és a helyi adatbázis-kezelő rendszer.
- A lemezes tároló eszköz és a tárolást kezelő szoftver.

Az áteresztő lánc minden láncszeme egy adott feladatot teljesít. Egy adott konfiguráció legjobb áteresztőképességét mindig a láncban szereplő leglassabb összetevő határozza meg. Az elosztott adatbázis teljesítményének növeléséhez az adatbázis-adminisztrátornak a láncban szereplő gyenge pontokon kell javítania.

Összefoglalás

A fejezetben az elosztott adatbázisok fogalmával ismerkedtünk meg. A lehetséges felépítései és jellemzői után a tervezésbeli és a létrehozásbeli problémákkal foglalkoztunk. Megismertük az elosztott adatbázisokhoz kapcsolódó szabványokat: a DRDA-t és az RDA-t. A távoli oldalon lévő adatok lekérdezésére és módosítására áttekintettük a különböző elérési módokat: távoli kérés, távoli munkaegység, elosztott munkaegység, elosztott kérés; illetve megvizsgáltuk a közöttük lévő különbségeket. A tranzakciókat az elosztott adatbázisokban visszagörgeti vagy véglegesíti a rendszer, ez utóbbihoz kétfázisú COMMIT-ra van szükség. Megnéztük a megvalósításának a fázisait. Végül az elosztott rendszerek teljesítményproblémáit tekintettük át, ezen belül az áteresztőlánc egyes elemeit vizsgáltuk meg.

Ellenőrző kérdések

1. Mi az elosztott adatbázis?
2. Van-e kitüntetett csomópont az elosztott adatbázisban?
3. Összesen hány adatbázis-kezelő rendszert kell telepíteni egy elosztott adatbázis megvalósításhoz?
4. Egy elosztott adatbázisban csak egyforma típusú adatbázis-kezelő rendszerek kapcsolódhatnak?
5. Elosztott adatbázisban az adatok redundánsan vannak tárolva?
6. Milyen jellemzőket használnak elosztott adatbázisok esetén?
7. Milyen tényezőket kell figyelembe venni az elosztott adatbázisok tervezése és létrehozása esetén?
8. Mitől függ az, hogy melyik csomópontra helyezzük el az adatokat?
9. Elosztott adatbázisra fejlesztett alkalmazások esetén a fejlesztőket milyen információkkal segíti az adatbázis-adminisztrátor?
10. Elosztott rendszerekkel kapcsolatosan milyen szabványokról hallottunk?
11. A távoli oldalon lévő adatok lekérdezésére és módosítására milyen elérési módok léteznek? Mi a különbség közöttük?
12. Mi a kétfázisú COMMIT? Mire szolgál? Hogyan megy végbe?
13. Elosztott rendszerek esetén milyen extra teljesítményproblémával kell az adatbázis-adminisztrátornak szembe néznie?
14. Mi az az áteresztő lánc? Milyen komponensek vesznek benne részt?

7. fejezet – Adatbázis-biztonság

Az adatbázis-kezelő rendszer felügyeli az adatbázis minden erőforrását. Egy felhasználó csak akkor végezhet el egy adatbázis-műveletet, ha engedélyezték számára a művelet végrehajtását, vagy pedig a művelet minden felhasználónak engedélyezve van. Nincs alapértelmezett feljogosítás, amelyet minden felhasználó eleve megkap, amikor belép az adatbázisba.
Az adatbázis-kezelő rendszer biztonsági eszközeit használva az adatbázis-adminisztrátor alakítja ki, hogy mely felhasználók, illetve mely alkalmazások milyen műveleteket végezhetnek el egy adott adatbázisban. Ha a felhasználók több szerver több adatbázisát szeretnék elérni, akkor az adatbázis biztonság adminisztrálása sokkal bonyolultabbá válik. A különböző adatbázisokra vonatkozó feljogosítást külön-külön meg kell adni, azaz a parancsokat minden adatbázisban meg kell ismételni. Általában nincs olyan központi repository, amely segítené azt, hogy a felhasználók biztonsági beállításainak a módosítását vagy a törlését több adatbázisban egyidejűleg végre lehessen hajtani.
Általában az adatbázis-adminisztrátor felelős az adatbázis-biztonság adminisztrálásáért. Azonban az is előfordulhat, hogy a cégünk ezt a feladatot egy elkülönített biztonsági adminisztrációra hárítja, amely a cég teljes IT biztonságát felügyeli. Léteznek olyan cégek is, amelyeknek van külön biztonsági adminisztrációja, ám az adatbázis-biztonságot mégis másképp kezelik, mint egy tipikus IT feljogosítást.
Ha a cégnél a biztonsági politikát biztonsági adminisztrációs csoport kezeli, akkor ez a csoport gyakran egy biztonsági szoftvert használ. A biztonsági szoftvertermékekkel a biztonság kezelése és felügyelete automatizált, így nem szükséges az, hogy az adminisztrátornak magasabb privilégiumai legyenek a biztonsági politika kezeléséhez.

Az adatbázis-biztonság alapjai

A szigorú hitelesítés minden adatbázis megvalósítási terve esetén nagyon fontos. Lehetetlen feljogosítást felügyelni és nyomon követni nélküle.
Először egy felhasználói nevet kell létrehozni az adatbázis-kezelő rendszer felhasználójához. A felhasználói névhez jelszót kell rendelni, amelyet csak az tud, aki a felhasználói nevet használhatja. Néhány adatbázis-kezelő rendszer az operációs rendszerbeli felhasználói nevet és jelszót használja az adatbázis-kezelő rendszerbe való bejelentkezéshez is, mások külön az adott adatbázis eléréséhez és biztonságához létrehozott felhasználói nevet és jelszót használnak. Ha az adatbázis-kezelő rendszer felügyeli a felhasználói neveket, akkor az adatbázis-kezelő rendszernek még a következő információkra is szüksége van, vagy szüksége lehet:

- jelszó,
- alapértelmezett adatbázis: amelyhez a felhasználó csatlakozni fog,
- alapértelmezett nyelv: ha az adatbázis-kezelő rendszer több nyelvet támogat, a felhasználói névhez rendelendő nyelv,
- további információk a felhasználóról, akihez a felhasználói név tartozik: e-mail, telefonszám, hivatali hely, üzleti egység, stb. Ez dokumentációs célból hasznos.

A jelszót szabályos időközönként meg kell változtatni, hogy a betörőknek nehezebb legyen elérni az adatbázist. A legtöbb adatbázis-kezelő rendszerben van olyan eszköz, amely arra kényszeríti a felhasználót, hogy bizonyos időközönként változtassa meg a jelszavát, és amíg ezt nem teszi meg, addig nem tud bejelentkezni. Azokat a felhasználókat, akik nem változtatják meg a jelszavukat, az adatbázis-adminisztrátor letilthatja. A letiltások visszaállítása csak külön adminisztrációval történhet.
Ha az adatbázis-kezelő rendszer egy felhasználójának már nem szükséges az adatbázishoz hozzáférni, mert például elhagyja a céget, akkor az adatbázis-adminisztrátornak törölnie kell a

felhasználóhoz tartozó felhasználói nevet a rendszerből, amilyen hamar csak lehet. Probléma lehet, ha a felhasználói nevet nem lehet eldobni, mert a felhasználó aktuálisan be van jelentkezve az adatbázis-kezelő rendszerbe vagy a felhasználónak sok adatbázis-objektuma van. Ez is az egyik oka annak, hogy általában az adatbázis-adminisztrátoron kívül más felhasználó nem kaphat jogot arra, hogy adatbázis-objektumot létrehozhasson.
Az adatbázis-kezelő rendszernek lehet olyan eszköze, amellyel az adatbázis-adminisztrátor a felhasználói nevet zárolja. A felhasználói név zárolása megakadályozza, hogy a felhasználó elérje az adatbázis-kezelő rendszert, azonban a felhasználói nevet nem dobja el a rendszerből. A felhasználói nevet a zárolás alól fel lehet oldani, így újra engedélyezhető az adatbázis-kezelő rendszer elérése. Ez az eszköz abban az esetben is hasznos, amikor azokat a felhasználóknak, akik régen változtatták meg a jelszavukat, ideiglenesen nem engedjük, hogy elérjék az adatbázis-kezelő rendszert.
Néhány adatbázis-kezelő rendszer további eszközöket és paramétereket biztosít a felhasználói nevekhez és a jelszavakhoz. Például a következők korlátozására találhatunk eszközöket:

- hibás bejelentkezési próbálkozások száma, mielőtt a felhasználói név zárolásra kerülne,
- napok száma, amíg a jelszó érvényes,
- a jelszó újrahasználhatósága (a napok száma, mielőtt egy jelszó újrahasználható és a maximum szám, ahányszor a jelszó újrahasználható).

Ha az adatbázis-kezelő rendszer képes kikényszeríteni az időszakos jelszóváltást, akkor gyakran olyan jelszószabályokat is ki tud kényszeríteni, amelyek csökkentik a jelszó kitalálásának az esélyét. Ilyen szabály például a minimális hosszúság, vagy az alfanumerikus követelmény.

Jogok adása és visszavonása

Az adatbázis-erőforrások használatának a feljogosítását felhasználói névhez lehet rendelni.
Az adatbázis-adminisztrátor az adatvezérlő nyelv (Data Control Language, DCL) segítségével felügyeli az adatbázis-biztonságot. A DCL utasításokkal lehet jogot adni a felhasználóknak az adatbázis-objektumok és parancsok használatára. A DCL nyelv két alap utasítást tartalmaz a jogosultságok kezelésére:

- GRANT: a jogokat ad a felhasználónak
- REVOKE: visszavonja a jogokat a felhasználótól

A GRANT utasítás két listából áll össze: az egyikben a jogosultságok vannak, a másikban a felhasználók. A két listát kell összerendelni. A GRANT utasítást az a felhasználó tudja kiadni, aki magas szintű csoportfeljogosítással rendelkezik (pl. DBA), vagy WITH GRANT OPTION záradékkal kapott valamilyen jogot, vagy az adatbázis-objektum tulajdonosa.
A WITH GRANT OPTION záradékkal kapott jogosultságot a felhasználó továbbadhatja más felhasználónak. Ez alapján megkülönböztethetünk:

- Központi adminisztrációt: ekkor a központi adminisztrátor feladata a jogokat kiosztani. Rajta kívül más nem adhat jogosultságokat. A központi adminisztrációt általában könnyebb adminisztrálni, de az adatbázis-adminisztrátorra sokkal több feladat hárul.
- Nem központi adminisztrációt: A felhasználók jogokat adhatnak, illetve továbbadhatnak más felhasználók számára. A nem központi adminisztrációt általában könnyebb létrehozni, de sokkal nehezebb felügyelni. Minél több felhasználó adhat tovább jogot, annál nehezebb lesz a jogosultságokat kezelni, nyomon követni.

A kétféle megközelítésből a központi adminisztráció a megfelelőbb. Természetesen a megfelelő dokumentáció itt sem maradhat el.
Az adatbázisjogok kiadására tervezett alkalmazói programokat annak a felhasználónak kell futtatni, akinek van megfelelő jogosultsága kiadni az alkalmazói programba kódolt GRANT és REVOKE utasításokat. Az olyan alkalmazói programokba, amelyeket nem kimondottan adatbázisjogok kiadására terveztek, kerüljük a GRANT és REVOKE utasítások használatát. Egyébként az adatbázisbeli feljogosítások követhetetlenek lesznek.

A jogok típusai

Minden adatbázis-kezelő rendszer biztosít alapvető jogtípusokat, mint jogosultság adatok elérésére, adatbázis-objektumok létrehozására, vagy különböző rendszerfunkciók teljesítésére. A különböző adatbázis-kezelő rendszereknek azonban lehetnek további jogtípusai is.
A következő jogtípusokat általában minden adatbázis-kezelő rendszer biztosítja:

- táblajogok: annak a felügyeletére, hogy ki tudja lekérdezni és módosítani a táblában lévő adatokat
- adatbázisobjektum-jogok: felügyelni, hogy ki hozhat létre új adatbázis-objektumokat és ki dobhat el létező adatbázis-objektumokat
- tárolt-eljárás jogok vagy programjogok: felügyelni, hogy ki futtathatja a tárolt függvényt, vagy a tárolt eljárást, illetve az adatbázisbeli programokat
- rendszerjogok: felügyelni, hogy ki hajthat végre adott rendszerszintű tevékenységet

Táblajogok

A táblajogok felügyelik, hogy a felhasználók mely táblákat, nézeteket és táblabeli oszlopokat érhetnek el, illetve módosíthatnak. A következő jogosultságokat lehet a táblákhoz és a nézetekhez adni:

- SELECT: a felhasználó lekérdezhet a táblából vagy a nézetből
- INSERT: a felhasználó sorokat szúrhat be a táblába vagy a nézetbe
- UPDATE: a felhasználó módosíthatja a táblát vagy a nézetet
- DELETE: a felhasználó sorokat törölhet a táblából vagy a nézetből
- ALL: a felhasználó lekérdezhet, beszúrhat, módosíthat és törölhet a táblán vagy nézeten

Például a következő utasítás lehetővé teszi, hogy a FELHASZNALO7 nevű felhasználó a CIMEK táblából töröljön:
GRANT DELETE ON CIMEK TO FELHASZNALO7;
Néhány táblaprivilégium oszlop szinten is megadható. Ekkor a felhasználó a táblának csak a megadott oszlopát tudja módosítani, más oszlopokat nem. Például a következő utasítás a FELHASZNALO7 nevű felhasználónak azt a jogot adja, hogy a CIMEK táblának csak az IRANYITOSZAM nevű oszlopát módosítsa:
GRANT UPDATE ON CIMEK (IRANYITOSZAM) TO FELHASZNALO7;
A táblajogokat az adatbázis-adminisztrátor általában a fejlesztői és a tesztkörnyezetben a fejlesztőknek adja fejlesztési céllal.
A termelési rendszerben az adatok elérésének és módosításának a felügyeletét általában nem úgy oldják meg, hogy a különböző felhasználókhoz táblajogot rendelnek. Helyette inkább a felhasználók által használt alkalmazások, illetve a tárolt eljárások biztosítják az adatok elérésének a felügyeletét. A programozók és a végfelhasználók csak bizonyos esetekben kaphatnak táblajogokat a termelési környezetben.

Adatbázisobjektum-jogok

Az adatbázisobjektum-jogok felügyelik, hogy mely felhasználóknak van joguk létrehozni adatbázis-objektumokat, mint például indexeket, táblákat, nézeteket, tárolt eljárásokat, triggereket. Azt, hogy konkrétan milyen jogosultságokat lehet kiosztani, függ az adatbázis-kezelő rendszertől és az adott adatbázis-objektumoktól. A legtöbb adatbázis-kezelő rendszer esetén van CREATE jog, amelyet olyan adatbázis-objektumokra vonatkozóan lehet kiadni, mint például tábla, táblatér, index, trigger, vagy felhasználó által definiált típus. Például a következő utasítás jogot ad arra, hogy a FELHASZNALO5 és a FELHASZNALO9 nevű felhasználók táblákat és indexeket hozzanak létre:
GRANT CREATE INDEX, CREATE TABLE TO FELHASZNALO5, FELHASZNALO9;
Az adatbázis-adminisztrátor gyakran nem engedi, hogy rajta kívül más is hozhasson létre adatbázis-

objektumokat az adatbázisban. Ennek az az oka, hogy ha más felhasználók is kapnak jogot valamilyen adatbázis-objektum létrehozására, akkor nehéz lenne felügyelni az adatbázis-objektumok számát, illetve azt, hogy mely objektumokat használják valójában és melyek azok, amelyekről elfeledkeztek. Továbbá ha tábla, index vagy táblatér létrehozásának a jogáról beszélünk, akkor az adatbázis-adminisztrátor nehezen tudná meghatározni a pontos tárigényt és tervezni az adatbázis számára szükséges tárterületet.
Azonban a fejlesztőknek gyakran kell a munkájuk miatt tárolt eljárásokat, tárolt függvényeket, triggereket létrehozni, amelyekhez a jogot természetesen a legtöbb esetben meg is kapják. A fejlesztők ezeket a jogokat gyakran egy szerepkör segítségével kapják meg.

Tárolt-eljárás jogok vagy programjogok

Az EXECUTE jogosultság segítségével adhatunk a felhasználóknak engedélyt egy program vagy tárolt eljárás futására. Például a következő parancs a FELHASZNALO1 nevű felhasználónak és FELHASZNALO9 nevű felhasználónak a PROC1 nevű tárolt eljárásra ad futtatási jogok:
GRANT EXECUTE ON PROC1 TO FELHASZNALO1, FELHASZNALO9;
A programjogokat és a tárolt-eljárás jogokat gyakran egyszerűbb felügyelni, mint a táblajogokat. Emiatt lehet az hasznos, ha a termelési adatokat csak a programokon vagy tárolt eljárásokon keresztül érjük el. A programban és az tárolt eljárásban lévő programlogika kezeli azt, hogy mely táblák és oszlopok módosíthatóak. Így az adatbázis-adminisztrátornak könnyebb lesz a termelési adatok integritását karbantartani.

Rendszerjogok

A rendszerjogok felügyelik, hogy a felhasználók melyik adatbázis-kezelő rendszerbeli eszközöket használhatják és milyen adatbázis-kezelő rendszerbeli parancsokat futtathatnak. A rendszer-privilégiumok elérhetősége adatbázis-kezelő rendszerről adatbázis-kezelő rendszerre változhat, de a legtöbb tartalmazza a következőket: az archív adatbázis-naplózás, az adatbázisszerver leállítása és újraindítása, nyomkövetés indítása a monitorozáshoz, a tárolás kezelése és a gyorsítótárak kezelése.
A következő példa a FELHASZNALO6 nevű felhasználónak ad nyomkövetési jogot:
GRANT TRACE TO FELHASZNALO6;
A rendszerjogokat óvatosan kell kezelni, és általában az adatbázis-adminisztrátor és a rendszer-adminisztrátor számára kell fenntartani.

Jogok adása PUBLIC-nak

Az adatbázis-kezelő rendszereknek általában létezik egy speciális „felhasználója”: a PUBLIC. Valójában a PUBLIC nem felhasználó, de jogot adhatunk neki, és jogok elvehetünk tőle. Ha a PUBLIC-nak adunk jogot, akkor a hozzárendelt jogot mindenki megkapja, aki az adatbázisba be tud jelentkezni. A PUBLIC-nak adott jogokat nem lehet WITH GRANT OPTION-nal adni, hiszen azt úgyis mindenki megkapja.
Például a következő utasítás minden felhasználó számára lehetővé teszi, hogy sorokat töröljön a CIMEK nevű táblából:
GRANT DELETE ON CIMEK TO PUBLIC;
Ha jogokat adunk a PUBLIC-nak, az adatbázis-adminisztrátor elveszítheti a kontrollt az adott adatbázis-objektum vagy erőforrás felett, mert mindenki elérheti vagy használhatja azt az objektumot vagy erőforrást, melyet a GRANT utasításban megadtunk. Bonyolultabb feladat, ha a jogokat felhasználónként osztjuk ki, de így az adatbázis-adminisztrátor az adatbázis-objektumok és erőforrások elérhetőségét nyomon tudja követni.

Jogok visszavonása

A REVOKE utasítást használjuk az olyan jogosultságok visszavonására, amelyeket előzőleg a

GRANT utasítással odaadtunk. A REVOKE szintaktikája hasonló a GRANT szintaktikájához: megadjuk, hogy mely felhasználóktól milyen jogokat vonunk vissza. A privilégiumokat az adatbázis-kezelő rendszer automatikusan visszavonja, ha egy adatbázis-objektumot eldobnak.
Például a következő utasítás visszavonja a CIMEK tábla IRANYITOSZAM oszlopának módosítási jogosultságát a FELHASZNALO7 nevű felhasználótól:
REVOKE UPDATE ON CIMEK (IRANYITOSZAM) FROM FELHASZNALO7;
Ha egy felhasználó egy jogot kétféleképpen kapott meg: egyszer a saját jogán, egyszer pedig azért, mert a PUBLIC megkapta, és ha a PUBLIC-tól visszavonjuk a jogot, akkor természetesen a felhasználó a saját jogán továbbra is rendelkezik az adott jogosultsággal.

Kaszkádolt visszavonás

Ha egy jogot az adatbázis-adminisztrátor visszavon, akkor az adatbázis-kezelő rendszernek el kell döntenie, hogy szükség van-e további jogosultság visszavonására valamely felhasználótól. Ha egy jog visszavonása arra készteti az adatbázis-kezelő rendszert, hogy további jogokat vonjon vissza, azt kaszkádolt visszavonásnak nevezzük.
Ha Jánosnak X joga van és joga van továbbítani azt, és az X jogot továbbadta Károlynak és Gézának, és ha Jánostól elveszik az X jogot, akkor Károlynak és Gézának sem lesz a továbbiakban X joga.
Azért, hogy a kaszkádolt visszavonás hatását minimalizáljuk, kerüljük a WITH GRANT OPTION-nal adott jogokat. Minél kevesebb az olyan felhasználó, aki jogokat tud adni, annál könnyebb kezelni és adminisztrálni az adatbázis-kezelő rendszer biztonsági struktúráját.

Biztonsági riportok

Az adatbázis-adminisztrátornak monitorozni és riportolni kell a felhasználók jogait. Az adatbázis-biztonság a rendszerkatalógusban van karbantartva. Az adatbázis-adminisztrátor SQL utasítások segítségével kinyerheti a szükséges információkat a megfelelő rendszerkatalógus-táblákból. Néhány adatbázis-kezelő rendszer nézeteket és tárolt eljárásokat biztosít, amelyek leegyszerűsítik az adatbázis-biztonsági információk kinyerését.
Legyünk biztosak, hogy a rendszerkatalógus megfelelő védelmet kap a termelési rendszerben. Csak az adatbázis-adminisztrátornak, rendszer-adminisztrátornak és a biztonsági adminisztrátornak kell elérnie a rendszerkatalógusban tárolt, adatbázisra vonatkozó biztonsági információkat.
A felhasználói biztonsági követelmények és elvárások az idővel alakulnak. Ha új alkalmazásokra van szükség az üzleti követelmények változása miatt, az adatbázis-biztonságnak is változnia kell. Az adatbázis-adminisztrátornak szabályos időközönként biztonsági ellenőrzést kell végrehajtania, hogy biztosítsa azt, hogy a megvalósított adatbázis-biztonság megegyezik a jelenlegi felhasználói követelményekkel. A rendszerkatalógus tábláiból készült riportok lehetnek egy ilyen felülvizsgálat inputjai.

Szerepkörök és csoportok feljogosítása

A legtöbb adatbázis-kezelő rendszerben használhatunk szerepköröket az adatbázis-biztonság kezelésének megkönnyítésére. Illetve a legtöbb adatbázis-kezelő rendszernek vannak speciális beépített felhasználói csoportjai, amelyek elsősorban az adatbázis-adminisztrációs feladatok elvégzéséhez szükséges jogosultságokat tartalmazzák.

Szerepkörök

A szerepkör jogosultságok gyűjteménye. A szerepkörök segítségével egy vagy több felhasználónak egyszerre több jogot adhatunk. Később, ha ugyanezeknek a felhasználóknak új jogot szeretnénk adni, vagy elvenni tőlük egyet, elég csak a szerepkörnek odaadni, vagy elvenni azt, és a

felhasználók jogosultsága automatikusan megváltozik. Az adatbázis-biztonság adminisztrációja a szerepkörök segítségével egyszerűsödik. Például tekintsük a következő utasítássorozatot:

```
CREATE ROLE VEZETOK;
GRANT SELECT, INSERT, UPDATE, DELETE ON DOLGOZOK TO VEZETOK;
GRANT SELECT, INSERT, UPDATE, DELETE ON RESZLEGEK TO VEZETOK;
GRANT EXECUTE ON BERLISTA TO VEZETOK;
GRANT VEZETOK TO FELHASZNALO1, FELHASZNALO2;
GRANT SELECT ON MUNKAKOROK TO VEZETOK;
```

Definiáltunk egy új szerepkört VEZETOK néven, jogokat adtunk bizonyos táblákhoz és eljárásokhoz a szerepkörnek, és összerendeltük a VEZETOK szerepkört a FELHASZNALO1 és a FELHASZNALO2 nevű felhasználóval. Az utolsó utasításban egy új jogot rendeltünk a VEZETOK szerepkörhöz, így az új jogot a FELHASZNALO1 és a FELHASZNALO2 is megkapta.

Egy harmadik felhasználót is összerendelhetünk a VEZETOK szerepkörrel. Az adatbázis-adminisztrátornak nem kell újra kiadnia minden GRANT utasítást, mert azokat a VEZETOK szerepkörhöz rendelte. Elég csak a VEZETOK szerepkört odaadni a harmadik felhasználónak is.

Csoportok

A csoport szintű feljogosítás hasonló a szerepkörökhöz. Ezeket a beépített csoportokat az egyes adatbázis-kezelő rendszerek biztosítják, és nem lehet őket változtatni. Általában adminisztrációs feladatok elvégzésének a feljogosítását rendelték hozzájuk. Az adatbázis-kezelő rendszerek a csoport szintű adatbázis-biztonságot különböző módokon valósítják meg. Egyes adatbázis-kezelő rendszereknél szerepkörökkel valósítják meg, más adatbázis-kezelő rendszereknél ez egyfajta jogosultság, és majd a bejelentkezésnél kell megadni, hogy milyen csoport tagjaként kívánunk dolgozni az adatbázisban, és persze más lehetőségek is létezhetnek. A csoportnevek és jogok a különböző adatbázis-kezelő rendszerekben mások-mások lehetnek. Az adatbázis-kezelő rendszerek között azért sok hasonlóság van. A következő csoportok általánosak a fő adatbázis-kezelő rendszerekben:

- Rendszer-adminisztrátor (a rövidítése SA vagy SYSADM): A rendszer-adminisztrációs csoport a legerősebb az adatbázis-kezelő rendszerben. Egy felhasználó rendszeradminisztrátor-szintű feljogosítással általában az adatbázis-kezelő rendszerben minden parancsot futtathat, minden adatbázist és adatbázisbeli vagy az adatbázis-kezelő rendszerhez kapcsolódó objektumot elér. A rendszer-adminisztrátor az adatbázis-kezelő rendszer telepítéséért felelős és úgy tekintünk rá, mint a rendszererőforrások és a rendszerkatalógus-táblák tulajdonosára. Csak azok az adatbázis-adminisztrátorok vagy rendszerprogramozók kaphatják meg a feljogosítás ezen szintjét, akik a cégnél lévő összes adatbázis-kezelő rendszert elérhetik. A végfelhasználóknak, menedzsereknek és alkalmazás fejlesztőknek nincs szükségük rendszer-adminisztrátori feljogosításra a munkájukhoz.
- Adatbázis-adminisztrátor (a rövidítése DBADM vagy DBA): Az adatbázis-adminisztrátori csoport a cég egy adott adatbázis-kezelő rendszerében minden joggal rendelkezik. Az adatbázisadminisztrátor-szintű feljogosítással rendelkező felhasználó eldobhat és törölhet minden objektumot az adatbázisban (táblatér, tábla, és index).
- Adatbázis-karbantartó (a rövidítése DBMAINT): az adatbázis-karbantartó csoport egy bizonyos adatbázis-objektumainak karbantartásához kap jogokat. Például újraszervezhet táblákat, vagy egy táblatérhez új adatállományt adhat hozzá. Úgy, mint az adatbázis-adminisztrátori szinten, az adatbázis-karbantartói szinten is a jogok egy-egy adatbázis-kezelő rendszerhez tartoznak.
- Biztonsági adminisztrátor (a rövidítése SSO): Az adatbázis-kezelő rendszeren belül a jogosultságok adásához és visszavonásához szükséges feljogosítást kapja meg. A biztonsági adminisztrátor minden, az adatbázis-biztonságához kapcsolódó tevékenységet végrehajthat,

beleértve a felhasználói nevek létrehozását, a jelszavak adminisztrációját, az auditot, a biztonsági konfigurációkat, és GRANT és REVOKE utasítások kiadását.
- Műveletek (üzemeltetés) felügyelője (rövidítéssel OPER vagy SYSOPER): a műveletek felügyelőjének joga van az üzemeltetési adatbázis-feladatok elvégzésére, mint például a mentés vagy a helyreállítás.

Más adatbázis-biztonsági mechanizmusok

Nézetek használata a biztonsághoz

A legtöbb adatbázis-biztonságot az adatbázis-kezelő rendszer belső biztonságának használatával valósítja meg az adatbázis-adminisztrátor. Lehetséges egyszerűsíteni az adatbázis-biztonságot, úgy hogy az adatok védelmére nézeteket hozunk létre.
Ha a cégünknek van egy DOLGOZO nevű táblája, amely a dolgozók minden adatát tartalmazza, és ha egy felhasználónak lekérdezési jogot adunk rá, akkor a felhasználó természetesen nem csak a saját magára vonatkozó adatokat érheti el, hanem a tábla összes adatát.
Azonban létrehozhatunk egy olyan nézetet, amely a DOLGOZO nevű tábla kényes információit kihagyja. Ha a tábla helyett a nézetre adunk a felhasználóknak lekérdezési jogot, akkor így a felhasználók a táblának csak a nézetben meghatározott részét kérdezhetik le. Például:

```
CREATE VIEW MINDEN_DOLGOZO AS
SELECT VEZETEKNEV, KERESZTNEV, IRANYITOSZAM, HELYSEG, UTCA
FROM DOLGOZO;
```

Ez az egyszerű példa egy nézetet mutat be, amely az alaptábla csak bizonyos oszlopait kérdezi le. Ha a felhasználó csak a nézetre kapja meg a lekérdezési jogot, akkor csak a nézetben meghatározott információkat nyerheti ki. Vertikális megszorításnak nevezzük, ha egy nézet kihagyja az alaptábla bizonyos oszlopait.
Horizontális megszorításnak nevezzük, amikor a nézet az alaptáblának csak bizonyos sorait kérdezi le. Ezt úgy adhatjuk meg, hogy a nézet definíciójában használt lekérdezésben WHERE feltételt adunk meg. Például:

```
CREATE VIEW DOLGOZO_RESZLEG20 AS
SELECT VEZETEKNEV, KERESZTNEV, IRANYITOSZAM, HELYSEG, UTCA
FROM DOLGOZO
WHERE RESZLEG_AZON=20
WITH CHECK OPTION;
```

Ha a felhasználók lekérdezik a nézetet, csak a feltételnek megfelelő sorokat kapják meg.
Ha a WITH CHECK OPTION záradék nincs megadva és a nézeten frissítést vagy törlést hajtunk végre, akkor olyan értékek is változhatnak, amelyek a nézetben nem látszanak. Például ha olyan dolgozókat törlünk az adatbázisól akiknek a vezetékneve 'A' betűvel kezdődik, akkor nem csak a 20-as azonosítójú részleg megfelelő dolgozóit törölnénk, hanem a táblában található összes dolgozó közül választanánk ki az 'A' betűvel kezdődőeket.
Ha viszont megadjuk a WITH CHECK OPTION záradékot, akkor frissítéskor és törléskor biztosak lehetünk abban, hogy azok az adatok, amelyek a nézet WHERE feltételét nem elégítik ki, nem fognak módosulni vagy törlődni.
A nézet segítségével biztosítani tudjuk, hogy egy felhasználó pontosan azokhoz (és csak azokhoz) az adatokhoz férjen hozzá, amelyekhez jogosultsága van, anélkül, hogy ismerné a mögöttes üzleti logikát és táblakezelést.

Tárolt eljárások használata a biztonsághoz

Egy tárolt eljárás futtatásának a jogát explicit módon kell megadni a felhasználónak, illetve

visszavonni tőle. A tárolt eljárás futtatásához a felhasználónak nincs szüksége más jogra. Azaz, ha a tárolt eljárás egy tábla tartalmát törli, akkor a felhasználónak nincs szüksége olyan jogosultságra, amellyel az adott tábla tartalmát törölheti. A tárolt eljárásnak kell olyan felhasználó nevében futnia, amelynek van joga törölni a tábla tartalmát.
A tárolt eljárásokat meg lehet úgy valósítani, hogy egy tábla adatainak csak sor vagy oszlopszintű részhalmazát érjék el. A tárolt eljárás futtatásának a jogát a felhasználókhoz lehet rendelni úgy, hogy az a tárolt eljárás tulajdonosának a jogosultságaival fusson. Ha a felhasználónak nincs joga a tárolt eljárás által módosított táblának a frissítésére, a felhasználó a tárolt eljáráson keresztül akkor is képes elérni az adatokat. Ezzel a módszerrel a szükséges biztonság is biztosítva van és ha megfelelően van kódolva, akkor akár jobb teljesítményt is biztosíthat.
Egy cég életében előfordulhat olyan eset is, amikor a felhasználó a napnak csak egy bizonyos részében használhatja egy tábla adatait, más időszakban viszont nem tekintheti meg. Az ilyen és ehhez hasonló esetekben a biztonságot tárolt eljárással lehet megoldani. A tárolt eljárás elindításakor az eljárás ellenőrzi, hogy az eljárást futtató felhasználónak van-e joga az adatokkal dolgozni a napnak annak az időszakában.

Audit

Az audit egy olyan adatbázis-kezelő rendszerbeli eszköz, amely lehetővé teszi az adatbázis-adminisztrátornak, hogy kövesse az adatbázis erőforrásainak a használatát. Ha az audit engedélyezve van, az adatbázis-kezelő rendszer az adatbázis-műveletekhez auditsort fog termelni. Minden követett művelet információk egy auditsorát eredményezi, amely magában foglalja, hogy az adatbázisbeli művelet milyen adatbázis-objektumokra volt hatással, ki hajtotta végre a műveletet és mikor. Fontos nyomon követni, hogy ki, milyen műveletet, milyen adaton hajtott végre. A legtöbb „betörést” a cég olyan jelenlegi vagy régi dolgozója követi el, akinek érvényes felhasználói neve és jelszava van. Az audit ezeket a betörőket segít megtalálni, így lehetővé teszi a biztonság megsértésének felfedezését.
Az audit utólagos tevékenység, azaz nem tesz semmit az adatbázisbeli objektum elérésének megakadályozására. Ahhoz, hogy az adatok elérését megakadályozzuk, a jogosultságokat kell megfelelően kiosztani az adatbázis felhasználóinak.
Egy tipikus auditeszköz különböző szintű auditot tesz lehetővé. Megkülönböztethetünk például adatbázis, adatbázis-objektum és felhasználói szinteket.
Az adatbázis-kezelő rendszerek auditeszközeinek általában az egyik legnagyobb problémája a teljesítménydegradáció. Az auditsorok tartalmazzák az adatbázis-változás előtti és utáni képeket, azaz az audit nagyon sok információt gyűjt össze. A sok információ összegyűjtése egy leterhelt rendszerben teljesítmény csökkenést okozhat. A másik probléma az auditeszközökkel, hogy az auditsorokat valahol tárolni kell. Minél több változás történik, annál több auditsort fog termelődni, amit tárolni kell. A legtöbb auditeszköz segíti az auditrekordok szelektív létrehozását, így minimalizálja a teljesítmény és tárolási problémákat.
A különböző adatbázis-kezelő rendszerek különböző auditopciókat ajánlanak. Tekintsünk át néhány általános opciót:

- a be- és kijelentkezési próbálkozások (sikeres, és sikertelen)
- adatbázis-kezelő rendszer újraindítása
- a rendszer-adminisztrátori jogosultsággal rendelkező felhasználók által kiadott parancsok
- integritás megsértésének próbája (ha a változtatott vagy beszúrt adat nem egyezik egy hivatkozott egyediségi vagy ellenőrzési megszorítással)
- SELECT, INSERT, UPDATE, DELETE műveletek
- tárolt eljárások futtatása
- sikertelen próbálkozás egy adatbázis vagy tábla elérésére (jogosultság megsértése)
- rendszerkatalógus-táblák változtatása

- sorszintű műveletek

Ha az adatbázis-kezelő rendszer nem támogatja a számunkra szükséges auditszintet vagy opciót, akkor egy másik cégtől lehet naplóelemző eszközt vásárolni.
Ha az adatbázis-kezelő rendszertől auditot kérünk, vegyük figyelembe a következő tanácsokat:

- Az audit a rendszererőforrások nagy fogyasztója. Ha az auditsor tele van, a feladatok, melyek auditrekordokat generálnak, várni fognak addig, amíg az auditfeladatot folytatni nem lehet.
- Egy elkülönített inaktív lemezre helyezzünk el rendszerkatalógus-táblákat, melyek a biztonsághoz kapcsolódó információkat tárolják. Ez növeli az auditteljesítményt az író-olvasó fejek mozgatásának a csökkentésével.
- Biztosítsuk, hogy az az adathalmaz vagy tábla nem telik meg, amely az auditadatok tárolására szolgál. Ha az adathalmaz megtelik, az audit megbénul. Ekkor a felhasználói feladatok, melyek az auditsornak adatokat próbálnak küldeni, hibaüzenettel le fognak állni.

Külső biztonság

Az adatbázis-adminisztrátornak biztosítania kell, hogy az adatbázis-kezelő rendszer által használt erőforrások védve legyenek, akkor is, ha ezek az adatbázis-kezelő rendszer felügyeletén kívül vannak. Ha az adatbázis erőforrásai nem csak az adatbázis-kezelő rendszerbeli parancsok és SQL utasítások használatával érhetők el, akkor a biztonsági mechanizmusokat nem lehet csak az adatbázis-kezelő rendszerbeli felhasználói hitelesítésre bízni. Kerüljük azt a szituációt, hogy az adatbázis felhasználói a futtató operációs rendszerhez kapjanak operációs-rendszer szintű jogokat.
Ha külső biztonsági mechanizmust használunk az adatbázishoz kapcsolódó erőforrások védelmére, akkor az adatbázis-adminisztrátornak elsődlegesen az adatbázis-kezelő rendszer által használt állományokra kell figyelni. Operációs rendszer vagy állományrendszer szinten a következő állományokat kell védeni:

- rendszerkatalógus adatállományai
- aktív és archív naplóállományok
- felhasználói adathalmazokat tartalmazó állományok
- a felhasználói adathalmazokhoz tartozó indexeket tartalmazó állományok
- audithoz tartozó adatállományok
- teljesítmény-követő állományok
- program és szkript állományok (forrás és futtatható kód)

Ha a cég adatbázis-kezelő rendszereihez tartozó állományoknak nincs megfelelő védelme, akkor a leleményes betörők kitalálhatják az állományok felépítését és jogosulatlanul elérhetik az adatokat.
Fontoljuk meg az adattitkosító szoftver használatát az adatbázis-állományokra. Az adattitkosítás egy olyan biztonsági technika, amely kódolja az olvasható adatokat az állományban.

Feladatütemezés és biztonság

A legtöbb cég ütemezi a feladatokat, azaz megadja, hogy az adott feladatok mikor fussanak. Ha ezeknek a feladatoknak az adatbázisban lévő adatokkal kell dolgozniuk, akkor az adatbázis-adminisztrátornak a feladatütemezőhöz kell a megfelelő jogosultságokat rendelni. Előfordulhat, hogy az ütemezést egy másik cég terméke hajtja végre. Az adatbázis-adminisztrátornak meg kell találnia a legjobb megoldást arra, hogy az ütemező szoftverhez a megfelelő adatbázisbeli jogosultságokat rendelje.
Ne adjunk az ütemezőnek rendszer-adminisztrátori vagy adatbázis-adminisztrátori csoportfeljogosítást. Az minden lehetséges, adatbázis-kezelő rendszerhez kapcsolódó feladat elvégzését engedélyezné, amely esetleg biztonsági problémákhoz vezet. Helyette adjunk egyedi jogosultságokat a különböző feladatokhoz az ütemező szoftver és az adatbázis-kezelő rendszer

eszközeinek használatával. Sok feladatütemezőt be lehet úgy állítani, hogy a különböző feladatokat különböző felhasználók nevében lássa el. A különböző felhasználókhoz lehet rendelni a feladatnak megfelelő jogosultságokat az ütemezett műveletek elvégzésére.
Egy másik általános biztonsági hiba, amikor a jelszavakat az adatbázis segédprogramokba és szkriptekbe ágyazzák. Ha a jelszó le van írva a kódban, bárki elolvashatja és a rendszeren kívül használhatja. Ez nem védi az adatainkat.

Az adatbázis-adminisztrátor operációs-rendszer szintű jogai

Az adatbázis-adminisztrátornak a cég adatainak és adatbázisainak a kezelésével és adminisztrálásával járó feladatok elvégzésére magas szintű operációs rendszerbeli feljogosításra van szüksége. Telepítés esetén ez rendszergazdai jogosultságot jelenthet, azaz például UNIX-on az adatbázis-adminisztrátornak root jelszóra lehet szüksége. Ha az adatbázis-adminisztrátor nem kaphat ilyen magas feljogosítást az operációs rendszerhez, akkor a rendszer-adminisztrátorhoz kell fordulnia, hogy a feladatait elvégezze. Megoldás lehet, hogyha az adatbázis-adminisztrátor és a rendszer-adminisztrátor együtt egy hatékony operációs-rendszer biztonságot dolgoz ki, amely lehetővé teszi az adatbázis-adminisztrátornak a feladatait elvégezni, de védi a platform biztonságát és integritását.

Összefoglalás

A fejezetben az adatbázis-kezelő rendszerhez szükséges biztonságot tekintettük át. Azzal kezdtük, hogy az adatbázisba belépni csak felhasználói névvel és jelszóval lehet. Áttekintettük a felhasználói névnek és a jelszónak a tulajdonságait is, és azt, hogy az adatbázis-kezelő rendszerektől milyen támogatást várhatunk el a felhasználói névvel és a jelszóval kapcsolatban.
Ha van felhasználói nevünk, tudunk hozzá jogosultságot rendelni. Megnéztük a jogokat adó, és a jogokat visszavonó utasításokat, illetve azt, hogy milyen típusú jogokat lehet velük adni: táblajogok, adatbázisobjektum-jogok, tárolt-eljárás jogok vagy programjogok, rendszerjogok. Ezeket a jogtípusokat részleteztük, illetve hoztunk rájuk példát. Beszéltünk még a kaszkádolt visszavonásról és a biztonsági riportokról.
Megismerkedtünk a szerepkör fogalmával, és használatával. Felsoroltuk azokat az általános felhasználói csoportokat, amelyeket a különböző adatbázis-kezelő rendszerek adatbázis-adminisztrációs feladatok elvégzéséhez szükséges jogosultságok kiosztására használnak. Megnéztük, hogy a nézeteket és a tárolt-eljárásokat hogyan lehet az adatbázis-biztonsághoz használni. Megismerkedtünk az audit fogalmával, áttekintettük a vele járó problémákat, illetve azt, hogy az adatbázis-kezelő rendszerek általában milyen lehetőségeket kínálnak vele kapcsolatban. Felsoroltuk még néhány tanácsot az auditfeladatok ellátásához.
A fejezet végén pedig az olyan biztonsággal foglalkoztunk, amelyet már nem az adatbázis-kezelő rendszer felügyel, mint az operációs rendszerbeli állományok biztonsága, a feladatütemező jogosultsági kérdései, és az adatbázis-adminisztrátor jogai az operációs rendszerben.

Ellenőrző kérdések

1. Ha a céghez új fejlesztő érkezik, akkor az adatbázis-adminisztrátornak milyen biztonsági feladatai vannak az új alkalmazottal?
2. Az adatbázis-kezelő rendszerben milyen információkat szükséges/lehet egy felhasználói névhez rendelni?
3. Mit jelent az, ha az adatbázis-kezelő rendszerben egy felhasználói nevet zárolunk? Miért jó ez?
4. Miért szükséges bizonyos időközönként megváltoztatni a felhasználónak a jelszavát? Az adatbázis-adminisztrátor milyen adatbázis-kezelő rendszerbeli eszközöket használhat a

jelszóváltoztatás kikényszerítésére?

5. Milyen utasítással lehet jogosultságokat adni, illetve visszavonni?
6. Mit jelent a WITH GRANT OPTION záradék?
7. Hogyan csoportosítottuk a jogokat? Melyik csoport milyen jogokat tartalmaz? Hozzunk példát az egyes típusokra!
8. Ki az a PUBLIC?
9. Mit jelent a kaszkádolt jogosultság visszavonás?
10. Mik azok a biztonsági riportok? Miért hasznosak?
11. Mik azok a szerepkörök? Miért hasznosak? Hogyan lehet őket használni?
12. Az adminisztrációs feladatok elvégzésére milyen csoportokat különböztettünk meg?
13. Hogyan használhatjuk a nézeteket az adatbázis-biztonsághoz?
14. Hogyan használhatjuk a tárolt eljárásokat az adatbázis-biztonsághoz?
15. Mi az az audit?
16. Milyen problémák merülnek fel az audittal kapcsolatban?
17. Az adatbázis-kezelő rendszerek milyen auditopciókat ajánlanak fel?
18. Milyen tanácsokkal találkoztunk az audittal kapcsolatban?
19. Operációs rendszer szinten milyen állományok védelmére kell odafigyelnie az adatbázis-adminisztrátornak?
20. A feladatütemező szoftverek számára milyen jogosultságot rendeljen az adatbázis-adminisztrátor?
21. Az adatbázis-adminisztrátornak az operációs rendszerhez milyen jogosultságra van szüksége?

8. fejezet – Adatbázismentés

Egy új adatbázis átadásakor általában a rendszer megfelelően működik. Azonban idővel a dolgok változnak. Egyre több lesz a felhasználó, egyre több lesz az adat, új szoftverek és hardverek kerülnek a rendszerbe, stb. És az eddig jól működő adatbázisunkban egyszer valamilyen hiba történik. Az adatbázismentés történhet jogszabályi és termelési előírás alapján is, ahol előírják, hogy egy adott állapotot hosszú távra archiválni kell.
Az adatbázis-adminisztrátornak fel kell készülnie a rendszerben bekövetkező hibák kezelésére. A fellépő hiba hatással lehet az elérhetőségre, az integritásra vagy az adatbázis használhatóságára. A bekövetkező hibákra az adatbázis-adminisztrátornak minél hamarabb reagálnia kell. Az, hogy az adatbázis-adminisztrátor milyen gyorsan és mennyire hatékonyan tud egy fellépő hibára reagálni, sokszor attól függ, hogy az adatbázismentés megfelelően meg van-e szervezve és van-e megfelelő adatbázis-helyreállítási terv. Azaz van-e a cégnek megfelelő mentési-helyreállítási stratégiája.

Lehetséges problémák feltérképezése

A rendszerhibáknak nagyon sok oka lehet. Az adatbázis mentési-helyreállítási stratégiájának a tervezésénél vegyük figyelembe az összes olyan lehetőséget, amely az adatbázis elérhetőségét és integritását fenyegeti. Az olyan technikák, mint UPS rendszer, tükrözött lemezek, stb. csökkenthetik a mentés szükségességét.
Az adatbázishibákat, amelyek mentést igényelhetnek, 3 kategóriába sorolhatjuk:

- példányhiba: az adatbázis-kezelő rendszerben egy belső kivétel eredménye, egy operációs rendszer hiba vagy más szoftver hiba. Néhány esetben a példányhiba adathibát is eredményezhet, amelyhez helyreállítás szükséges, de általában ezek a hibák nem okoznak az adatokkal kapcsolatos sérülést, ezért az adatbázis-kezelő rendszert egyszerűen újra kell indítani és helyre kell állítani a normál működést.
- alkalmazáshiba(vagy tranzakcióhiba): akkor következik be, ha egy program vagy szkript rossz időben fut, rossz inputtal vagy rossz sorrendben. Egy alkalmazáshiba általában hibás adatokat eredményez, amely adatbázis-helyreállítást igényel. Minél hamarabb azonosítjuk és javítjuk az alkalmazáshibát, annál kisebb kár keletkezik az adatbázisban.
- médiahiba: lemezhiba, állomány-rendszerbeli hiba, szalagkárosodás, vagy törölt adatállományok. Károsítják az adatainkat. A memóriachip hibái is okozhatnak adathibát. Egy médiahiba után az adatbázis olyan állapotba kerül, ahol az érvényes adatok olvashatatlanok, érvénytelen adatok olvashatóak vagy a hivatkozási integritás sérül. A médiahibák különböző lemeztechnológiákkal (mint pl. RAID) megelőzhetőek.

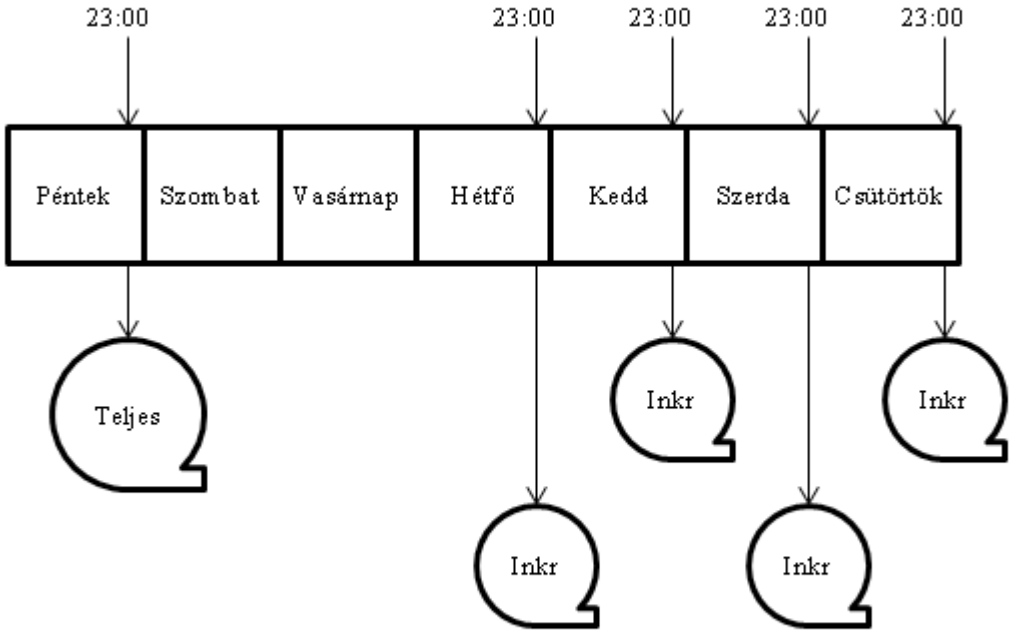
Képmásolati mentés

A képmásolati mentés a mentési és helyreállítási terv alapkomponense. Ha hiba történik, és sérül az adatintegritás, akkor az adatok mentési képmásolatát lehet használni az adatbázis helyreállításához.
Az adatbázis mentése az adatok egy konzisztens másolatát jelenti, amelyet a COPY segédprogram outputja ad. A segédprogram adatbázis-kezelő rendszerenként más és más, lehet BACKUP, COPY, DUMP, és EXPORT. Néhány adatbázis-kezelő rendszer natív operációs rendszerbeli állományrendszer-parancsot használ az adatok mentésére.
Csak naprakész és pontos képmásolati mentésekből lehet az adatbázist helyreállítani. Ehhez az adatbázis-adminisztrátornak a mentések tervezésénél két alapvető dologról kell döntenie: milyen gyakran végezzen képmásolati mentést, és hány mentési generációt tartson meg.
Az adatbázis mentésének a gyakoriságát két fontos tényező befolyásolja. Az egyik az, hogy az adatbázis-adminisztrátornak úgy kell meghatároznia a mentések gyakoriságát, hogy a mentési

folyamatok ne akadályozzák a napi üzleti munkát. A másik szempont pedig az, hogy ha hiba történik a rendszerben, akkor a helyreállításra fordítandó idő minél kevesebb legyen. Hiszen amíg a helyreállítás nem történik meg, addig az adatbázist nem lehet használni. Ekkor pedig a cégünk nem tudja ellátni a tevékenységét.
Az adatbázis-adminisztrátornak a mentési generációkból legalább kettőt meg kell tartania. Ennek az az oka, hogy ha az egyik mentési generáció megsérül és használhatatlanná válik, akkor még az adatbázisunk egy másik, esetleg régebbi mentési generáció alapján helyre lehessen állítani. Ha régebbi mentési generációt használunk a helyreállításra, akkor szükség lesz az azóta bekövetkező módosításokat tartalmazó naplóra is, hogy a helyreállított adatbázis a lehető legutóbbi időpillanatbeli állapotot tartalmazza.
Az adatbázis konzisztens, naprakész helyreállításához a legtöbb esetben az adatbázisnaplóra is szükség van. Az adatbázis-adminisztrátornak a mentés tervezése során nem szabad elfeledkeznie az adatbázisnapló megfelelő mentéséről sem.

Teljes vagy inkrementális mentés

A teljes mentés azt jelenti, hogy az adatbázis minden objektumát, és azoknak a teljes adattartalmát képmásolati állományba vagy állományokba menti az adatbázis-kezelő rendszer. Az inkrementális mentés ezzel szemben csak azokat az adatváltozásokat tartalmazza, amelyek az utolsó mentés óta történtek.
Az inkrementális mentés előnye, hogy sokszor gyorsabb lehet, és kevesebb helyet foglalhat a lemezen vagy a szalagon. A hátránya, hogy a helyreállítás az inkrementális mentés alapján sokkal lassabb, mivel egy sor akár többször is változhatott mielőtt az utolsó érték előállt.
Néhány adatbázis-kezelő rendszernek van olyan eszköze, amelynek a segítségével elemezni lehet, hogy melyik mentést érdemes választani az adott esetben: a teljes vagy az inkrementális mentést. Ha ez az opció nem érhető el, akkor az adatbázis-adminisztrátor csak a saját tapasztalataira és ismereteire hagyatkozhat.
Általában az inkrementális mentés akkor javasolható, ha az adatbázisban kevés változás történt vagy ha az adatbázis rendelkezésre állása elsődleges az üzlet szempontjából. Azonban ha az adatblokkoknak a 30-40%-a vagy több módosult, akkor már teljes mentés javasolt.
Kis adatbázisok esetén a teljes mentés a javasolt. Egy adatbázis kicsinek tekinthető, ha még nem érte el a 100GB méretet. Azonban ez az érték a technika fejlődése miatt az idővel növekszik.
A 8-1-es ábrán egy adatbázis mentéseit láthatjuk. Pénteken végeztek egy teljes mentést, majd a munkával töltött napok után inkrementális mentéseket végeztek az adatbázison.



8-1. ábra Adatbázismentés

Egyesített inkrementális másolatok

Ha az adatbázis-kezelő rendszer támogatja az inkrementális mentést, akkor lehet, hogy az inkrementális mentések egyesítést is támogatja. Az egyesítés során a rendszer a teljes mentést egyesíti az inkrementális mentésekkel, és létrehoz egy új teljes mentést.
Az egyesítés nincs hatással az adatbázis teljesítményére, hiszen elvégezhető az adatbázis működésétől függetlenül.
Az egyesített teljes mentés létrehozásával a helyreállítás időtartamát rövidíthetjük le.

Adatbázis-objektumok és mentések

A képmásolati mentés adatbázis, táblatér- vagy táblaszinten történik. Azaz olyan objektumokat mentünk, amelyek adatokat tárolnak. A támogatott szint az adatbázis-kezelő rendszertől függ. Általában az adatbázis-kezelő rendszer minél szemcsézettebb vezérlést biztosít az adatbázis-objektumok mentéséhez, annál könnyebb hatékonyan megvalósítani egy használható mentési-helyreállítási stratégiát. Finomabb szemcsézettség esetén könnyebb különböző adatbázis-objektumokhoz különböző mentési ütemezést megvalósítani. A ritkán vagy egyáltalán nem változó adatainkat ritkán kell menteni, esetleg csak hetente vagy havonta. Azonban a dinamikus, gyakran változó adatokat akár naponta vagy óránként is menthetjük.

Indexek mentése

Az indexeket az adatok alapján az adatbázis-kezelő rendszer újra fel tudja építeni. Ezért kézenfekvő a kérdés, hogy kell-e az indexeket menteni.
A legtöbb adatbázis-kezelő rendszer támogatja az indexek mentését, sőt van olyan adatbázis-kezelő

rendszer, amelynél az indexeket is menteni kell. Más adatbázis-kezelő rendszereknél viszont az adatbázis-adminisztrátor döntheti el, hogy szükséges-e egy indexet menteni.
Ha az index is mentve van, akkor az adatbázis-adminisztrátor a helyreállításnál választhat, hogy visszaállítja azt, vagy újraépíti a visszaállított adatok alapján.
Azt, hogy az adatbázis-adminisztrátor hogyan dönt egy index mentéséről, és a későbbi helyreállításáról, befolyásolhatja, hogy a mentett indexnek mennyi tárterületre lesz szüksége, illetve ha az index nincs mentve, akkor mennyi időbe telik újragenerálni azt. A nagyobb táblák indexeinek az újragenerálása olyan sok időt vehet igénybe, hogy inkább megéri menteni, és helyreállítani, annak ellenére, hogy sok helyet foglalnak. Ugyanez a helyzet a multimédiás adatok indexelésével is.
Ha az indexek mentését és helyreállítását végezzük, figyeljünk oda, hogy a mentett adatoknak a mentett indexek megfeleljenek. Ha nem felelnek meg egymásnak, akkor helyreállítás után elérhetetlenek lehetnek az adataink vagy a lekérdezések hibás vagy érvénytelen eredményeket adhatnak.

Párhuzamos elérés

Az adatbázis mentésének az a legegyszerűbb módja, hogy leállítjuk az adatbázist, lementjük, majd újra elindítjuk. Azonban ebben az esetben a felhasználók számára az adatbázis nem érhető el.
Azonban ma már a legtöbb adatbázis-kezelő rendszer ismeri azt a mentési technikát, amellyel az adatbázis mentése alatt az adatbázis elérhető marad. Ebben az esetben a mentésben az adatbázisnaplónak lesz nagy szerepe. Hiszen mentés közben a mentés alatt lévő objektumok adattartalma módosulhat, ami azt eredményezi, hogy a kapott mentés nem lesz konzisztens. A konzisztencia biztosításához az adatbázisnaplóra lesz szükség. A naplóban megtalálhatjuk a mentés elkezdése óta történt változásokat. A mentés után az adatbázis-adminisztrátornak gondoskodnia kell arról, hogy a szükséges adatbázisnapló is mentve legyen.
Néhány adatbázis-kezelő rendszer használja a forró mentés és hideg mentés kifejezéseket, amely a mentés alatti párhuzamos elérésekre utal. A forrónál az adatbázis online marad, a hidegnél le kell állítani a releváns állományokat.
Az adatbázis-kezelő rendszer kapacitásától függően a forró mentés problémásabb lehet, mert:

- sokkal komplexebb megvalósítani
- nagyobb a processzor és az input/output műveletek igénye a mentés alatt
- archiválni kell a naplót
- az adatbázis-adminisztrátornak esetleg szkripteket kell írnia
- alapos tesztelést igényel, hogy biztosak lehessünk, hogy a mentés valóban alkalmas a helyreállításra

A forró mentésnek a helyreállításnál is van hátránya, mert az adatbázis-kezelő rendszernek az adatbázisnaplóban található bejegyzéseket újra végre kell hajtania, ezáltal a helyreállítás időtartama megnő.
Néhány adatbázis-kezelő rendszernek a mentési-helyreállítási segédprogramja képes arra, hogy egy naprakész, konzisztens képmásolati mentést készítsen a meglévő képmásolati mentés és az adatbázisnapló egyesítésével. Hasonlóan, mint az inkrementális képmásolatok egyesítésénél.
Létezik olyan mentési technika is, amely a felhasználóknak csak azt engedi meg, hogy a mentés alatt lévő adatbázis-objektumokat olvassák. Ez gyorsabb mentést biztosít, mint azok a technikák, ahol a párhuzamos írás is engedélyezve van, és a mentésünk a napló nélkül is konzisztens lesz.
Az adatbázis-adminisztrátornak ismernie kell az adatbázis-kezelő rendszer mentési eszközének működését. A párhuzamos elérés tekintetében a megfelelő mentési stratégia kidolgozásához a következőket kell figyelembe venni:

- A párhuzamos elérések és módosítások szükségessége a mentési folyamat alatt.
- A mentési folyamatra szánható idő mennyisége és a párhuzamos elérések hatása az adatok

mentésének gyorsaságára.
- A helyreállítási segédprogram sebessége.
- Helyreállításkor az adatbázisnaplók elérésének szükségessége.

Mentési konzisztencia

A mentési terv elkészítésekor bizonyosodjunk meg arról, hogy a mentési terv alapján az adatbázis-objektumok mentésekor konzisztens helyreállítási pontot hozunk létre, és, hogy minden, az adatbázis-objektumokkal kapcsolatban álló objektum mentésre kerül. Ez jelenti az alkalmazásokhoz szükséges kapcsolatokat, a hivatkozási megszorításokat és a triggereket is. Helyreállításkor minden objektumot ugyanahhoz a mentési időponthoz tartozó pontra kell visszaállítani. Ha egy korábbi mentést használunk, akkor minden objektumnak az adott időpontbeli verziója szükséges.
Az adatbázis-kezelő rendszer QUIESCE nevű segédprogramot biztosíthat, amelyet arra használhatunk, hogy egy adatobjektumnak az adott időpillanatban lévő konzisztens állapotát menthetjük ki. A QUIESCE segédprogram megállítja az adatbázis-objektumokra vonatkozó módosítási kéréseket, hogy biztosítsa a konzisztenciát és mentse az adatbázisnaplóba a konzisztenciapontot.
Ha az adatbázis-kezelő rendszer nem támogatja a QUIESCE opciót, más módon kell biztosítani a konzisztens pontot a helyreállításhoz. Pl. tehetjük az adatbázis-objektumot csak olvasható állapotba, vagy offline állapotba.

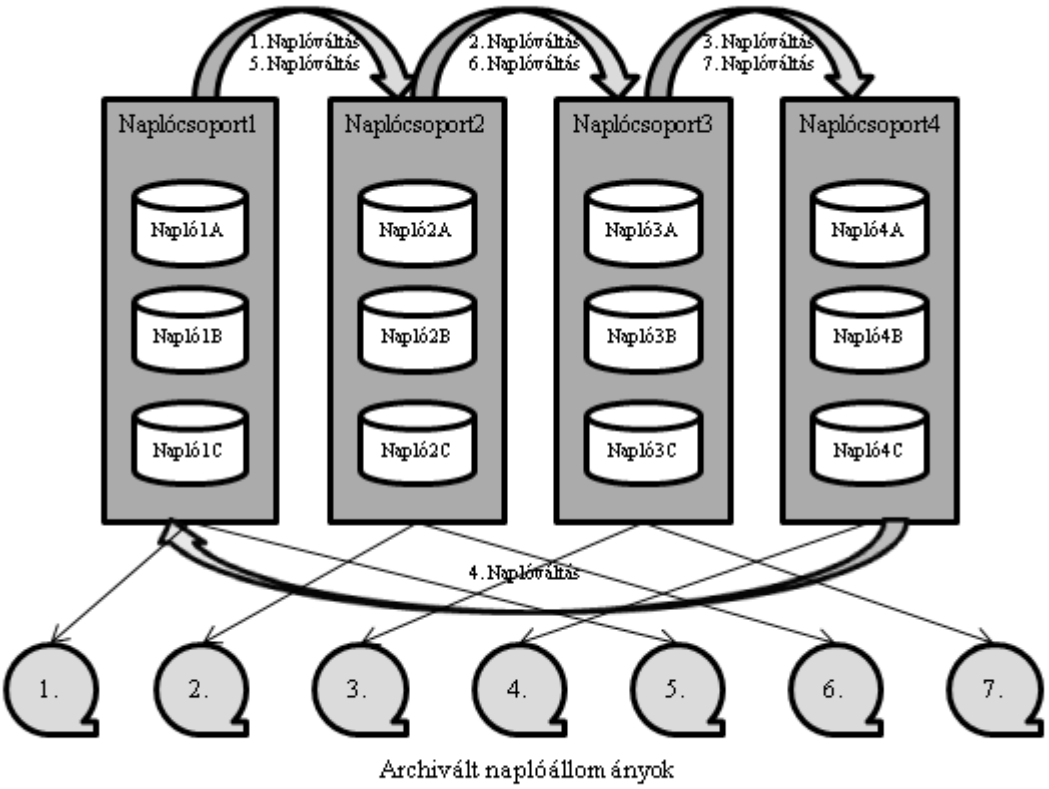
Mikor készítsünk konzisztenciapontot

Az adatbázis-adminisztrátornak a lehető leggyakrabban konzisztenciapontot kell készítenie. De a következő esetekben mindenképp ajánlott ezt megtenni:
- Az aktív napló archiválása előtt: Ha valamikor elveszítenénk az aktív naplót és az archív naplót kellene a helyreállításkor használni, akkor az archív naplót csak az utolsó konzisztenciapontig használhatnánk. Ha ez után használnánk, inkonzisztens adatokat kapnánk.
- Egymáshoz kapcsolódó adatbázis-objektumok mentése előtt, azaz például kapcsolódó adatbázistáblák mentése előtt. Ez biztosítja, hogy a képmásolati mentések minden kapcsolódó adatbázis-objektuma konzisztens egymással.
- Rögtön, miután egy képmásolati mentés elkészült. Így a mentés elkészültével egy jó helyreállítási pontot hozunk létre. Akkor érdemes, ha a mentés az online adatbázis működése közben történt.
- Azonnal, mielőtt súlyos adatbázisbeli módosítást hajtanánk végre. Ha a módosítás közben hiba történik, a helyreállítás a konzisztens állapotig, az előző konzisztenciapontig történhet. Így elkerülhető a képmásolati mentés a módosítás előtt.
- Csöndes időszakban: sok tevékenység esetén a konzisztenciapont létrehozása akadályozza a tevékenységeket. Ha az adatbázis-objektumokat nem módosítják, könnyebben lehet konzisztenciapontot létrehozni.

Naplóarchiválás és mentés

Az adatbázis-kezelő rendszer minden adatbázis-változást egy naplóállományba naplóz, amelyet hívnak tranzakciónaplónak vagy adatbázisnaplónak is. A naplórekordok minden SQL DML utasításhoz elkészülnek. Az adatbázisnapló alapján vissza lehet keresni, újra végre lehet hajtani, illetve vissza lehet vonni az adatbázison történt változások hatását.
Az idővel és az adatbázis-változások növekedésével az adatbázisnapló méretben növekedni fog. Azt az adatbázisnapló-állományt, amelyet az adatbázis-kezelő rendszer aktuálisan ír, úgy is hívjuk, hogy aktív napló. Ha az aktív adatbázisnapló betelik, naplóváltás történik, azaz egy új naplóállományt kezd el írni az adatbázis-kezelő rendszer. Ekkor a régi, betelt naplóállományt az adatbázis-kezelő

rendszer archiválja. A naplóállományokról bővebben az Adat- és tároláskezelés című fejezetben olvashatunk.
Egy adatbázisnapló-állomány archiválása naplóváltáskor történik, amikor az aktív naplót bezárjuk. A régi aktív naplóállomány tartalma nem törlődni fog, hanem az adatbázis-kezelő rendszer a tartalmát az archív naplóállományba másolja. Közben pedig az adatbázisban történő változások az új aktív naplóállományba kerülnek.
A 8-2-es ábrán a naplóváltáshoz kapcsolódó naplóarchiválási folyamatot láthatjuk.



8-2. ábra Naplóarchiválás

A napló archiválása esetén előfordulhat olyan eset, hogy hamarabb lesz szükség az adott naplóállományra, mint ahogy az archiválva lett. Azaz az adatbázis-kezelő rendszer nem tud írni aktív naplóállományt, mert a ciklusban soron következő naplóállomány még nincs archiválva és nem lehet törölni a tartalmát. Ebben az esetben az adatbázis-kezelő rendszer nem hajtja végre az adatbázisbeli módosításokat, amíg a soron következő naplóállomány törölve nem lesz.
Az adatbázis-adminisztrátor a napló archiválásának automatikus megvalósítását az adatbázis-kezelő rendszer paramétereivel vezérelheti. A legtöbb adatbázis-kezelő rendszer esetén az adatbázis-adminisztrátor egy parancs segítségével kézzel is kérheti a napló archiválását.
A helyreállítást, illetve a hibák miatti leállás megelőzését az is támogatja, hogy ha az aktív adatbázisnapló-állományból több, teljesen egyforma példány létezik, amelyeket az adatbázis-kezelő rendszer párhuzamosan ír. Ezeket a naplópéldányokat az adatbázis-adminisztrátor több lemezen helyezi el. Az adatbázis elérhetetlenné válik, ha nem tudja írni a naplót, hiszen az adatbázisbeli módosításokat nem tudja hova írni. Azonban több naplópéldány esetén, ha az egyik lemez sérül, akkor az adatbázis még elérhető marad, amíg a több példányból még egyet tud írni az adatbázis-kezelő rendszer. Az aktív adatbázisnaplóból mindig legalább két példányt kell biztosítani.

A mentési ütemezés meghatározása

Egyensúlyozni kell két szempont között: elég gyakran kell képmásolati mentéseket készíteni, de a cég napi üzleti munkáját ez nem zavarhatja. Az adatbázis-adminisztrátornak ezt az adatbázis-kezelő rendszer kapacitására és használati feltételeire kell alapoznia.
A mentés ütemezéséhez elemezni kell az adatbázisunkat, a következő kérdésekkel:

- Mennyire használják az adatot egy nap?
- Milyen gyakran változik az adat?
- Mennyire kritikus az adat a cég életében?
- Könnyen újra létre lehet hozni az adatot?
- Milyen típusú elérés szükséges az adatokhoz? 24/7 órás elérés szükséges?
- Mi a költsége annak, ha az adat nem érhető el?
- Mennyi a pénzértéke annak, ha a rendszer egy percig áll?

Általánosságban a kritikus adatokat gyakrabban kell menteni, mint a nem kritikusakat, és a dinamikus, gyakran változó adatokat gyakrabban kell menteni, mint a statikusakat. De mindez függ az adott adatbázistól és az adott cégtől.
A gyakori képmásolati mentés azért szükséges, hogy ha az adatbázisban hiba lép fel, akkor a helyreállításhoz szükséges idő a lehető legrövidebb legyen. Azaz hiba esetén a lehető legkevesebb ideig legyen elérhetetlen a rendszer.
A helyreállításhoz szükséges időtartamnak az egyik legmeghatározóbb tényezője az, hogy a helyreállításhoz szükséges naplót mennyi idő alatt lehet feldolgozni.
A helyreállításhoz szükséges időtartamot a következő tényezők határozzák meg:

- a szükséges mentési állományok megkeresése, beleérte azt is, hogy ha szalagos egységen vannak tárolva, akkor a szükséges szalagokat csatolni kell
- a teljes képmásolati mentések feldolgozása
- ha vannak, akkor az inkrementális mentés feldolgozása
- majd az archivált és az aktív adatbázis-naplórekordok feldolgozása

Ezek közül a tényezők közül a legidőigényesebb az adatbázis-naplórekordok feldolgozása lehet. Minél több idő telt el az utolsó mentés óta, annál több változás történt, annál több ideig tart ezeket a változásokat újra végrehajtani az adatbázisban. Ha régi mentésünk van, előfordulhat, hogy a feldolgozandó naplóállományok összmérete nagyobb lesz, mint a teljes adatbázisunk mérete. Ennek elkerülésére szükséges a gyakori mentés.
Ezért általában elmondhatjuk, hogy minél gyakrabban készítünk képmásolati mentést, annál rövidebb ideig tart a helyreállítás. A képmásolati mentés készítéséhez szükséges idő összegét ki kell egyensúlyozni a mentési folyamat alatt történő párhuzamos feldolgozás szükségességével.

Adatbázis-objektum definícióinak a mentése

Az adatbázis-objektumok SQL kódját is illik menteni. Idővel ezek is változnak, ezért a változások után szükséges a mentés.

Adatbázis-kezelő rendszer állományainak a mentése

Nagyon ritkán kell az adatbázis-kezelő rendszerhez tartozó állományokat menteni. Ezek az állományok jelentik a rendszerkatalógust, könyvtár-objektumokat, konfigurációs állományokat, rendszerkönyvtárakat, szalagkezelő könyvtárakat, programokhoz forrás és futtatható állományok könyvtárát.
Akkor lehet rá szükség, amikor eszköz vagy emberi hiba történik, vagy frissítéskor fellépő problémák megoldásakor.
Az adatbázis-kezelő rendszer állományainak a megfelelő mentését és helyreállítását is tervezni kell.

Az adatbázismentés más megközelítései

UNLOAD használata

Az adatbázis-kezelő rendszer UNLOAD segédprogramját használhatjuk logikai mentés készítésére. A következő esetekben lehet hasznos, ha van logikai mentésünk:

- adatbázis-objektum vagy táblabeli sor helyreállítása esetén: ha egy táblát valaki eldob, vagy egy pár sort töröl egy adatbázistáblából, akkor ezeket a fizikai mentésből nehéz visszaállítani. Egy logikai mentéssel a hiányzó adatokat egyszerűen visszatölthetjük a táblába.
- az adatbázis-kezelő rendszer verzió frissítése esetén: lehet, hogy egyszerűbb logikai mentést végezni és azt visszatölteni az új verzióba, mint a meglévő adatstruktúrát konvertálni.
- heterogén adatbázis migráció esetén: A különböző platformokon a fizikai struktúrák különbözhetnek. Pl. Oracle Linux-on vagy Unix-on. Egyszerűbb logikai mentést végezni és azt betölteni az új rendszerbe.
- adatmozgatás esetén: ha adatokat mozgatunk egy adatbázisból más adatbázisba, vagy a cégen belül több adatbázisba.

A logikai mentés az adatbázis online állapotában elvégezhető. A párhuzamos adatelérés a teljesítményre hatással lehet. A logikai mentésnél az adatbázis-adminisztrátornak az adatintegritásra figyelnie kell.

Tároláskezelő szoftver használata mentési másolat készítésére

Ebben az esetben az adatbázis-kezelő rendszer vezérlésén kívül történik a mentés. A mentés előtt bizonyosodjunk meg arról, hogy a menteni kívánt objektumok közül egyiket sem lehet írni, vagy állítsuk le az objektumot esetleg tegyük csak olvasható állapotba. A mentés után indítsuk el vagy tegyük írható-olvasható állapotba.
Az eszköz használata előtt ismerjük meg az eszköz működését, mert pl. van olyan tároláskezelő mentési mód, amelyik megnyitott állományt nem ment.
Helyreállításkor mindig azt a tároláskezelőt kell használni, amelyikkel mentettünk.

A mentési-helyreállítási stratégia dokumentálása

Ha egyszer a mentési-helyreállítási stratégiát megalkottuk, és a mentés ennek megfelelően van megvalósítva, akkor a mentési rendszer sokáig fut az adatbázis-adminisztrátor beavatkozás nélkül. Idővel a dolgokat azonban elfelejtjük, az adatbázis-adminisztrátori személyzet változik, és ha az adatbázis-adminisztrátornak az adatbázist helyre kell állítani, akkor tudnia kell, hogy ezt hogyan tegye. Ezért az adatbázis-adminisztrátornak megfelelően dokumentálnia és tesztelnie kell a mentési-helyreállítási stratégiát, és annak a megvalósítását, az eljárásokat.
Nagyon fontos, hogy a helyreállítás tesztelve legyen minden lehetséges hiba esetén: médiahiba, példányhiba, alkalmazáshiba, stb. Legyünk biztosak, hogy mindent helyre tudunk állítani.

Vezérelvek a helyreállítható környezet biztosításához

A mentési-helyreállítási stratégia elkészítésénél vegyük figyelembe a következő vezérelveket, hogy biztosítani tudjuk a helyreállítható környezetet:

- Legyen legalább két másolatunk minden képmásolati mentésből, hogy elkerüljük a helyreállíthatatlan állapotot, amelyet például egy médiahiba okozhat. A két másolatot két különböző helyen tároljuk.
- Koordináljuk a helyi mentési-helyreállítási stratégiánkat a katasztrófa mentési-helyreállítási stratégiánkkal. Sok mentési segédprogram párhuzamosan készíti el a helyi és a távoli mentést.

- Tartsunk meg legalább kettő generációt a képmásolati mentésekből az egyes adatbázis-objektumokhoz. Ha a legújabb képmásolat hibás, akkor visszatérhetünk egy előző másolathoz.
- A képmásolati mentést készítsük lemezre, majd migráljuk szalagra, amely felgyorsíthatja a mentés folyamatát. Nemcsak azért, mert a lemez gyorsabb, mint a szalag, hanem mert nem kell arra várni, hogy a szalagot kézzel kicseréljék.
- A mentéseket tömöríthetjük is, hogy csökkentsük a mentéshez szükséges szalagok számát.
- Legyünk biztosak, hogy a mentési-helyreállítási tervünk tartalmazza a rendszer-katalógusbeli adatbázis-objektumokat is.
- Biztosítsuk, hogy a mentési folyamat újraindítható legyen. Például ha a mentési folyamat 3 órát vesz igénybe és 2 óra 30 percnél megszakad, akkor az újraindításnál már csak fél óra kell, hogy befejezze, egyébként ha nincs újraindítás, akkor újabb 3 óra.
- Miután a mentés elkészült egy adatbázis-kezelő rendszerbeli eszközzel bizonyosodjunk meg a mentés helyességéről, azaz hogy a képmásolatok pontosak, érvényesek.
- Az alkalmazások által használt, de nem az adatbázisban tárolt adatokat ugyancsak ugyanabban az időben kell menteni.
- Biztosítani kell, hogy helyreállítási célra az adatbázisnapló is mentve legyen, illetve az aktív napló elérhető legyen.

Egyéb, a mentéssel kapcsolatos tanácsok:

- Ha a rendszerünket átszervezzük, érdemes utána teljes mentést végezni. Egyébként helyreállítás esetén az átszervezéskor keletkezett rengeteg adatbázisbeli változást a napló alapján újra végre kell hajtani, ami sok időt és erőforrást emésztene fel.
- Ha adatbázis-naplózás nélkül töltünk adatokat az adatbázisba, akkor utána mentést kell végezni. Ha nem végzünk mentést, hiba esetén a betöltött adataink el fognak veszni, hiszen adatbázis-napló nem készült róluk.
- Point-in-time helyreállítás után is végezzünk teljes mentést.
- Általában is elmondható, hogy ha az adatbázis-kezelő rendszerben naplózás nélküli változások történnek, akkor mindig a teljes mentés javasolt.

Alternatívák a mentésre és a helyreállításra

Tartalék (Standby) adatbázisok

Az online adatbázissal azonos másolat, amely majdnem naprakész az adattartalmat tekintve. Nem 100%-osan naprakész, mert az online oldal frissítéseinek hatása később jelenik meg a másolatban. Ha hiba történik a vezérlés átkerül a tartalék adatbázisba, és a normál működés ott folytatódik.
A tartalék adatbázis nem helyettesíti a normál mentést. Ha alkalmazásbeli hiba történik, akkor az a tartalék adatbázisban is bekövetkezik.
A tartalék adatbázist könnyű megvalósítani, és gyors helyreállítást biztosít bizonyos hibák, pl. katasztrófák esetén.

Másolatok (Replication)

Az adatmásolat azt jelenti, hogy az adatbázisban egy elszeparált adatbázis-objektumban redundáns adatokat tárolunk és tartunk karban. A másolat az eredeti adatbázis adatainak egy részhalmaza is lehet. Megvalósítható egyszerű táblamásolással. Néhány adatbázis-kezelő rendszer automatikus replikációs eszközt biztosít.
Két alaptechnika létezik: szimmetrikus másolat és pillanatfelvétel-másolat (snapshot).
A pillanatfelvétel-másolat adatbázistáblák másolata a forrásadatbázis egy lekérdezésén alapszik. A lekérdezés eredménye egy pillanatfelvétel-táblába kerül. Az egyes másolatokat az adatbázis-

adminisztrátornak kell naprakészen kell tartani. Ha több másolat létezik, az adatbázis-adminisztrátornak kell megoldania, hogy minden másolat ugyanabban a pillanatban frissüljön. Ha a másolatok különböző ütemben frissülnek, nehézzé válhat nyomon követni az egyes másolatok állapotát.
A pillanatfelvétel-másolat előnye a könnyű megvalósítás. A pillanatfelvétel-másolatok gyorsan elveszíthetik a naprakészségüket, és a frissítésük adminisztratív és teljesítménybeli problémákat okozhat.
A szimmetrikus másolat sokkal robusztusabb megvalósítás, mert a másolatokat automatikusan naprakészen tartja. A szimmetrikus másolatokat úgy lehet létrehozni, hogy a tranzakciók csak akkor legyenek teljesen véglegesítve, ha a módosítások a másolatokon is megtörténtek. Lehetséges a másolatokat aszinkron módon is frissíteni, hogy a frissítés ne akadályozza a helyi módosításokat. A frissítések később történnek, miután a fő adatbázisban a véglegesítés befejeződött.
A legnagyobb előnye a szimmetrikus másolatnak a pillanatfelvétel-másolattal szemben az, hogy a módosításokat automatikusan szinkronizálja. A hátránya a szimmetrikus másolatnak a pillanatfelvétel-másolattal szemben a nehezebb adminisztrálhatóság, és az, hogy a frissítések teljesítményproblémákat okozhatnak.
A másolatok nem helyettesítik a mentést és helyreállítást, de sok szituációban segíthetnek.

Lemeztükrözés

Az elsődleges lemezeszközről egy másodlagos eszközre másolat készül. Ha az elsődleges eszközben hiba történik, a másodlagos eszköz átveszi a szerepét.
Ez a másolás nem az adatbázis szintjén jön létre, hanem lemezszinten.
A mentés és helyreállítást nem helyettesíti, de a médiahiba kiküszöbölhető vele.

Összefoglalás

A fejezetben az adatbázis mentéséhez kapcsolódó ismereteket tekintettük át. Ehhez először megnéztük, hogy mik azok a hibák, amelyek miatt szükség lehet az adatbázist helyreállítani, és az elkészített mentést felhasználni. A képmásolati mentés a legfőbb alapfogalom az adatbázis mentése során. A képmásolati mentéshez kapcsolódóan megkülönböztettünk teljes és inkrementális mentést, megvizsgáltuk, hogy az adatbázist milyen szemcsézettséggel lehet menteni, illetve, hogy a felhasználók használhatják-e az adatbázist a mentés készítésével párhuzamosan. Felhívtuk a figyelmet arra, hogy forró mentés esetén a mentési konzisztenciára oda kell figyelni, melynek a biztosítását az adatbázisnaplóba kerülő konzisztenciapont segíti. A képmásolati mentéshez kapcsolódóan az adatbázisnaplót is menteni kell, egyrészt, hogy a konzisztenciát biztosítani lehessen, másrészt, hogy a mentés után bekövetkezett változásokat helyreállítás esetén újra alkalmazni lehessen az adatbázison. A mentési ütemezés meghatározásához különböző szempontokat soroltunk fel. Felhívtuk a figyelmet arra, hogy néha az adatbázis-kezelő rendszer állományainak a mentésére is szükség van. Az adatbázis mentése nem csak az adatbázis-kezelő rendszer mentési segédprogramjával végezhető el, hanem a megfelelő mentési stratégia támogatására használhatjuk az UNLOAD segédprogramot, vagy állomány-rendszerbeli mentést is végezhetünk. A mentést dokumentálni kell, amelyet az adatbázis mentési-helyreállítási stratégiájának a dokumentálásával tehetünk meg. Ehhez kapcsolódóan soroltunk fel vezérelveket és tanácsokat.
A fejezet végén alternatívákat soroltunk fel az adatbázis mentésére és helyreállítására, mint a tartalék adatbázisok, a másolatok, és a lemeztükrözés.

Ellenőrző kérdések

1. Hogyan csoportosítottuk az adatbázishibákat?

2. Mi az a képmásolati mentés?
3. A képmásolati mentés készítésének gyakoriságához milyen két legfontosabb tényezőt kell figyelembe venni?
4. Mit jelent a teljes és az inkrementális mentés? Mi a különbség közöttük?
5. Milyen szemcsézettséggel menthetjük az adatbázis-objektumait?
6. Szükséges-e menteni az indexeket? Miért?
7. El lehet-e érni az adatbázist a mentés alatt?
8. Mit jelentenek a forró mentés és a hideg mentés kifejezések?
9. Miért problémásabb a forró mentés?
10. Miért szükséges és hogyan tudunk konzisztens helyreállítási pontot létrehozni?
11. Mikor készítsünk konzisztenciapontot?
12. Szükséges-e a naplót menteni? Miért? Hogyan?
13. Hogyan lehet az adatbázis-naplózás segítségével támogatni a helyreállítást?
14. A mentés ütemezéséhez milyen kérdéseket kell feltenni a cégünknél?
15. Mi befolyásolja azt, hogy milyen gyakran kell képmásolati mentést készítenünk? Hogyan?
16. A táblabeli adatokon kívül mi az amit még mentenünk kell? Miért?
17. Az adatbázis-kezelő rendszer mentési segédprogramján kívül még milyen más eszközökkel tudunk adatbázismentést készíteni?
18. Miért szükséges dokumentálni az adatbázis mentési-helyreállítási stratégiáját?
19. Milyen vezérelvek vannak az adatbázis mentési-helyreállítási stratégiájának elkészítéséhez?
20. Milyen alternatívák léteznek az adatbázis mentésére és helyreállítására?

9. fejezet - Adatbázis-helyreállítás

Ha probléma történik az adatbázisban, és az adatbázis-adminisztrátor nem tudja gyorsan megoldani, akkor a cég adatai lehet, hogy elérhetetlenné válnak, ami a cégnek anyagi veszteséget okoz.
Ahhoz, hogy az anyagi veszteség elkerülhető legyen, az adatbázis-adminisztrátornak az adatbázisban történő problémára nagyon gyorsan kell tudnia reagálni. Így biztosítani tudja, hogy a cég üzleti folyamatai folyamatosan elérjék az adatbázist.
Azonban az adatbázis-adminisztrátor reagálási képessége nem csak azon múlik, hogy milyen gyorsan tud helyreállítani. A nehezebb feladat az, hogy a problémát fel kell tudni ismerni, és meg kell rá találni a megfelelő megoldást. Így az adatbázis-helyreállítás nagyon összetett feladat lehet, amely sok hibalehetőséget rejt magában és nehéz megfelelően megvalósítani.
A helyreállítás sokkal többet foglal magában, mint egyszerűen helyreállítani az adatok egy olyan képét, amely egy korábbi időpillanatban előállt. Az adatbázis-helyreállítás azt jelenti, hogy egy probléma előtti időben fennálló állapotba állítjuk az adatokat vissza. Gyakran a helyreállítás az adatok visszatöltése mellett azt is magában foglalja, hogy az adatbázisban történt, helyes változtatásokat újraalkalmazzuk a megfelelő sorrendben. Az adatbázis-adminisztrátor ehhez képmásolati mentéseket és az adatbázisnapló-állományokat használhatja.
A sikeres helyreállítás egy olyan előző állapot elérése, amely a múltban fennállt, és amelyet el akarunk érni. Ez az állapot lehetett akár a múlt héten, a múlt hónapban. Ha megfelelően megterveztük a mentési stratégiánkat, akkor képeseknek kell lennünk a helyreállításra minden lehetséges hiba esetén.

A helyreállítási opciók meghatározása

Ha hiba történik, az adatbázis-adminisztrátornak először meg kell tudnia, hogy szükséges-e helyreállítás. Ha szükség van helyreállításra, akkor meg kell határozni, hogy milyen erőforrások, mint mentések, adatbázisnaplók érhetőek el és a helyreállítást hogyan lehet a legjobban végrehajtani. Fogalmazzunk meg néhány kérdést, amelyek megválaszolásával megtudhatjuk a hiba okát és terjedelmét. A válaszok adják azokat a lépéseket, amelyek a rendszer helyreállításához szükségesek.

- Milyen hibatípus történt: média, tranzakció, vagy adatbázispéldány?
- Mi a hiba oka?
- Hogyan állt le az adatbázis? Megszakítással, vagy normál módon, esetleg összeomlott?
- Történt operációs rendszerbeli hiba?
- A szerver újraindult?
- Vannak hibák az alert log-ban?
- Milyen hibadiagnosztikai állomány készült?
- Generálódott nyomkövetési állomány?
- Mennyire kritikusak az elveszett adatok?
- Történt már eddig próbálkozás a helyreállításra? Ha igen, milyen lépések történtek?
- Milyen típusú mentés létezik: teljes, inkrementális, mindkettő?
- Mit kell helyreállítani: a teljes adatbázist, egy táblateret, egy táblát, egy indexet vagy ezek kombinációját?
- A mentési stratégiánk támogatja a szükséges helyreállítás típusát?
- Ha hideg mentésünk van, hogy állt le az adatbázis, amikor a hideg mentés készült?
- Minden archivált adatbázisnapló-állomány érhető el a helyreállításhoz?
- Van logikai mentésünk?
- Milyen párhuzamos tevékenységek voltak, amikor az adatbázis összeomlott?

- Az adatbázis-kezelő rendszert újra lehet indítani?
- Elérjük az adatbázis-objektumokat?
- Milyenek a rendszer elérhetőségi követelményei?
- Mennyi adatot kell helyreállítani?
- Raw állományokat használunk?

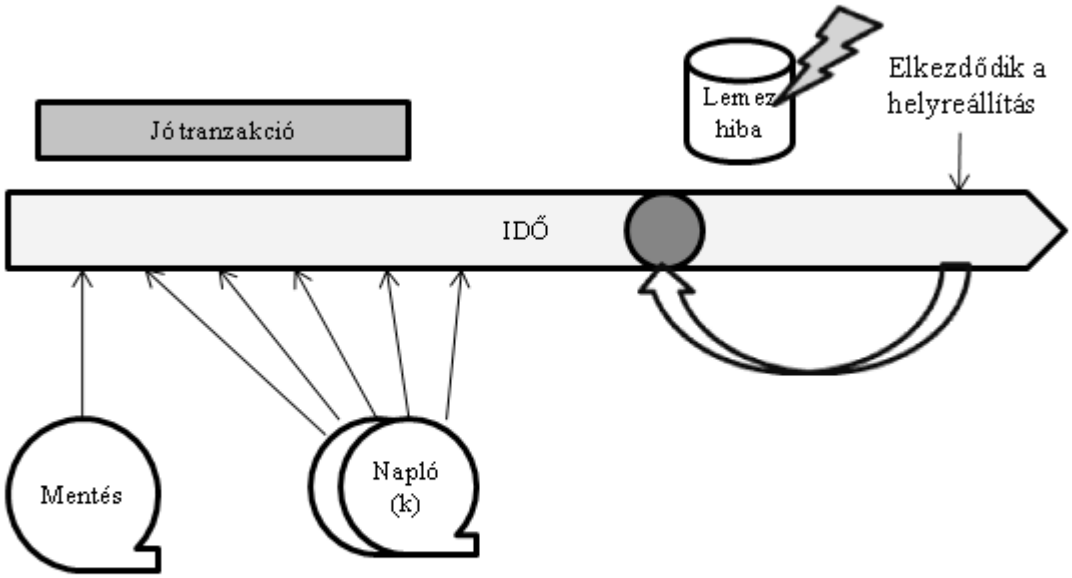
A verziómigráció hatással lehet a helyreállíthatóságra. Például ha 5-ös verzióban mentettünk egy táblát, majd migráltuk 6-os verzióba, és ha a hiba ekkor történik, akkor az adatbázis-kezelő rendszertől és az új verziótól függően a táblát esetleg nem lehet helyreállítani. Néha az adatbázis-kezelő rendszer szállítója módosít a mentési állományok felépítésén, vagy a naplóállományok felépítésén, így a régi formátumok az új verzióból elérhetetlenné válhatnak. A migrálás után mentést kell végezni, hogy az adatbázis helyreállítható legyen.

Általános lépések az adatbázis-objektum helyreállításához

1. Azonosítsuk a hibát: az adatbázis vagy nem válaszol vagy az adatbázis-kezelő rendszer bizonyos típusú hibaüzenetek ad. Néhány hiba nagyon alattomos, mint pl. a hibás vezérlőállomány. Ezt sokkal nehezebb azonosítani.
2. Elemezzük a helyzetet: Az adatbázis-adminisztrátornak elemeznie kell a hibát, hogy meghatározza a hiba okát, a típusát és a hatáskörét. Az elemzés eredményére alapozva az adatbázis-adminisztrátornak helyreállítási módot kell választania. A helyreállítási feladatban a legtöbb időt erre kell szánni.
3. Meg kell határozni, hogy mit kell helyreállítani: Az adatbázis-adminisztrátornak meg kell határoznia, mely adatbázis-objektumok (és talán más komponensek, mint például az adatbázisnapló) hibásak és elő kell készítenie a helyreállítási szkriptet, amely megfelelő az egyes komponensekhez. Ez a feladat is sok időt emészthet fel, például akkor ha egy nagy rendszerünk van.
4. Azonosítani kell a függőségeket a helyreállítandó adatbázis-objektumok között: Egy adatbázis-objektum hibája hatással lehet más adatbázis-objektumokra (indexek, hivatkozott táblák). Az adatvesztés és adott időpontra történő helyreállítás hatással lehet a kapcsolódó adatbázis-objektumokra is.
5. A szükséges képmásolati mentések előkeresése: minél közelebb van a képmásolati mentés a helyreállításban kért időponthoz, annál gyorsabban megy a helyreállítás. A megfelelő szalag megtalálása is időbe telik.
6. Helyreállítás a képmásolati mentésekből: az adatbázis-helyreállító segédprogramját vagy az állományrendszer parancsát használhatjuk.
7. Előregörgetés az adatbázisnapló alapján: az utolsó időpontra történő helyreállításhoz, vagy egy adott időpontra történő helyreállításhoz a képmásolati mentésből való helyreállítás után az adatbázisnaplót kell feldolgozni.

A helyreállítás típusai

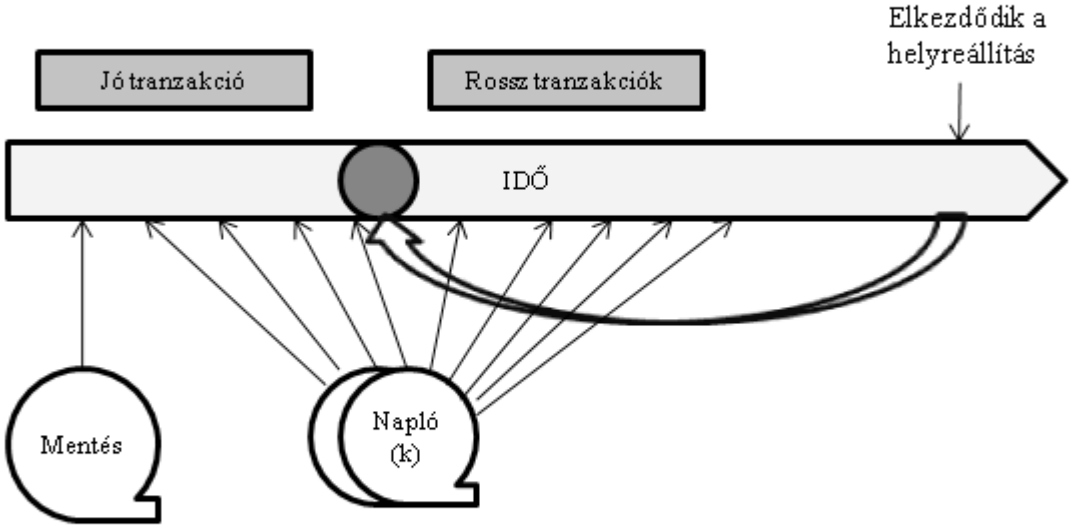
1. Helyreállítás az utolsó lehetséges időpontra: olyan katasztrófák után szokták használni, amely elpusztítja az adatközpontot. Ezt okozhatja média hiba, természeti katasztrófa, stb. Az alkalmazások teljesen elérhetetlenné válnak, amíg a helyreállítás be nem fejeződik.
   A sikeres helyreállításhoz a helyreállító folyamatnak képesnek kell lennie az adatbázis tartalmát a hiba előtti állapotra hozni. A helyreállító folyamatnak érvényes és teljes képmásolati mentést kell találnia és helyreállítania a képmásolatokat. A helyreállító folyamat az adatbázisnapló segítésével előregörget, azaz a mentés óta történt adatbázis-változtatásokat újraalkalmazza a képmásolatból helyreállított adatbázison.
   A 9-1-es ábrán ennek a helyreállításnak a lépéseit láthatjuk.



9-1. ábra Helyreállítás az utolsó lehetséges időpontra

Ha az utolsó teljes képmásolat elveszett vagy megrongálódott, még lehetséges a helyreállítás, ha az előző képmásolat létezik. A helyreállítási folyamat egy korábbi teljes mentésről indul, alkalmazza az inkrementális másolatokat, aztán előregörgeti az archív adatbázisnapló-állományokat majd az aktív naplóállományt. Minél több naplót kell előregörgetni, annál több időt vesz igénybe a helyreállítás.
Ha nem érhető el képmásolat, mint kezdési pont, meg lehet próbálni csak az adatbázisnapló alapján helyreállítani az adatbázist. Ha az adatokat betöltötték és a betöltés naplózva volt, a helyreállítás lehet egyszerűen a naplórekordok alkalmazása.
Ezzel a típusú helyreállítással egy-egy adatbázis-objektumot is helyreállíthatunk.

2. Point-in-time (PIT) helyreállítás (adott időpontra): Általában alkalmazás szintű problémák kezelésére szolgál. A point-in-time helyreállítás minden olyan tranzakciónak a hatását eltávolítja, amely az adott időpont óta történt. A helyreállítási pontot egy aktuális dátummal és idővel adhatják meg.
   A point-in-time helyreállítás véghezviteléhez egy képmásolati mentés alapján kell helyreállítani az adatbázist és az adatbázisnapló alapján előregörgetni azt, azaz az adatbázisnapló használatával a változásokat alkalmazni. A naplórekordokat csak az adott időpontig kell feldolgozni.
   A 9-2. ábra segítségével a point-in-time helyreállításnak a célját érthetjük meg.



9-2. ábra Point-in-time helyreállítás

A sikeres point-in-time helyreállításhoz a helyreállító folyamatnak az adatbázis-tartalmat egy előző konzisztens pontban lévő állapotra kell tudnia hozni. A point-in-time helyreállítás kétféleképp történhet:

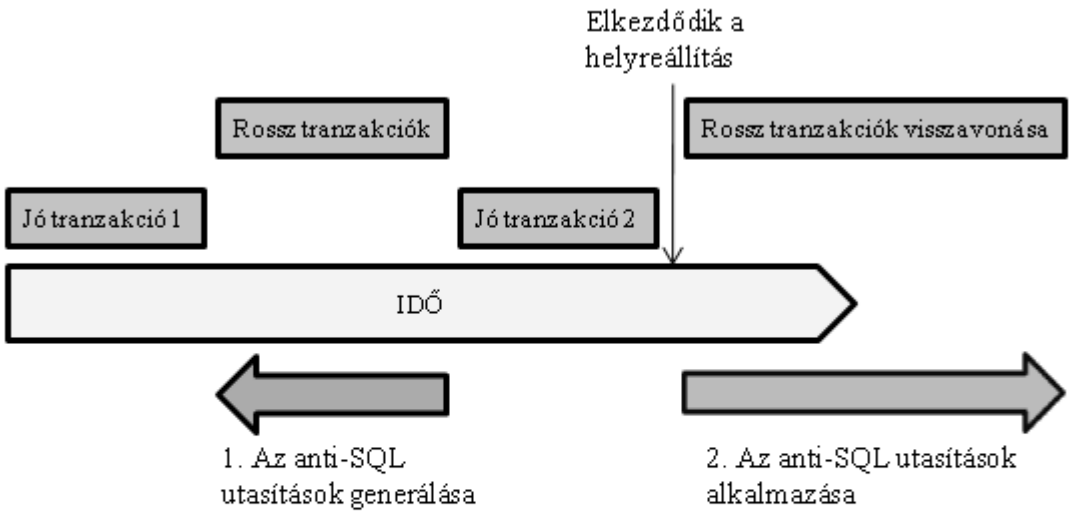
- Visszatöltjük az adatbázisba a képmásolati mentést, majd az adatbázisnapló alapján előregörgetünk, azaz az adatbázisnapló használatával újraalkalmazzuk a változásokat. A naplórekordokat azonban csak a megadott helyreállítási időpontig kell feldolgozni.
- Nem töltjük vissza a képmásolatot, helyette az adatbázisnapló alapján visszagörgetjük és eltávolítjuk azokat az adatbázis-változásokat, amelyek a helyreállítási pont után történtek.

Ha az adatbázis-kezelő rendszer támogatja a helyreállítás mindkét típusát, akkor az adatbázis-adminisztrátor választhat az megoldások közül. Érdemes arra alapoznia, hogy melyik megoldás megvalósítása esetén kevesebb a leállási idő. Ha sok változást kell eltávolítani, akkor a helyreállítás és előregörgetés kevesebb leállást vesz igénybe. Ha az eltávolítandó változások száma minimális, akkor az adatbázisnapló alapján történő visszagörgetés vesz kevesebb leállási időt igénybe.
A point-in-time helyreállítás típusától függetlenül az adatbázis-adminisztrátornak egy helyreállítási pontot kell választania, amely egy olyan pontot reprezentál, ahol az adatok konzisztensek voltak. Ez a pont a korábban létrehozott konzisztenciapontok egyike lehet, hiszen ebben a pontban biztosított az adatintegritás, a hivatkozási integritás, és a tranzakciók integritása is.
Meg kell határozni, hogy pontosan mi futott a hiba óta, amely miatt a helyreállítás történik. Az adatbázis-adminisztrátor megvizsgálja az ütemezett feladatokat, riportot készít az adatbázisnaplóból, és áttekinti a számítógép üzeneteit, hogy a hiba után futott folyamatokat

meghatározza.
A point-in-time helyreállítással nem csak a teljes adatbázis tartalmát lehet egy adott időpillanatra helyreállítani, hanem csak bizonyos objektumokét is. Ebben az esetben azonban még körültekintőbben oda kell figyelni az adatbázis-objektumok függőségére, hiszen ha egy adatbázis-objektumot helyreállítunk, akkor a tőle függőeket is helyre kell állítani.

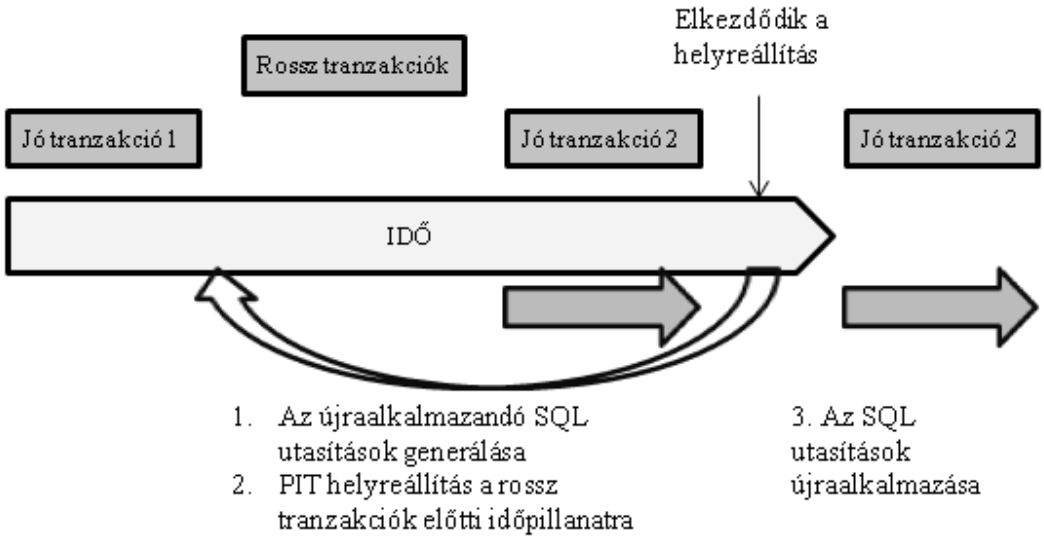
3. Tranzakció-helyreállítás: ide tartozik az a hagyományos helyreállítási típus is, amelyre akkor van szükség, amikor az adatbázis nem normál módon áll le. Ebben az esetben még lehetnek olyan tranzakciók, amelyeket már véglegesítettek, de a hozzájuk tartozó megváltozott adatok még nem kerültek az adatbázisba, azonban a tranzakció véglegesítésének a ténye már az adatbázisnaplóban van. Előfordulhat az is, hogy olyan adatok vannak az adatbázisban, amelyek olyan tranzakciók hatására módosultak, amelyeket még nem véglegesítettek. A tranzakció-helyreállításnak a hagyományos megközelítése az, hogy amikor az adatbázis újraindul, akkor egy helyreállító folyamat is újraindul, amely megvizsgálja az adatbázisnaplót és az adatbázis tartalmát, és a véglegesített tranzakcióhoz tartozó módosított adatokat az adatbázisba írja, a nem véglegesített tranzakcióhoz tartozó adatok módosítását pedig visszavonja.
   Ha nem hagyományos értelemben tekintünk a tranzakció-helyreállításra, akkor a tranzakció-helyreállítás alatt egy olyan alkalmazás-helyreállítást értünk, ahol egy bizonyos időtartamban történt tranzakciók hatásait eltávolítjuk az adatbázisból. Ez a helyreállítási típus nem adatbázis-objektumokra vonatkozik, hanem tranzakciókra.
   A tranzakció helyreállítás segítségével lehetővé válik a felhasználó számára az adatbázis bizonyos részeinek helyreállítása a felhasználó által megadott feltételekre alapozva. Ebben a környezetben a tranzakció a felhasználó által definiált nézete a folyamatnak. A legfontosabb feltétel, hogy a tranzakciók között, melyeket vissza próbálunk állítani és az adatbázis-kezelő rendszer többi tranzakciója között nem lehet korreláció.
   Például felhasználói szintű tranzakciók lehetnek: minden adatbázis módosítás, amelyet VALAKI userid-jű felhasználó hajtott végre szerda 11.50 óta; vagy minden adatbázis törlés, amelyet a FIZETES nevű alkalmazás tegnap 8.00 óta hajtott végre.
   Számos oka lehet, ami miatt a tranzakció-helyreállításra szükség van: a rendszerszoftverben hibák vannak, nem megfelelően tesztelt kód futott, valaki a feladatütemező szoftvert megváltoztatta, stb., vagy nincs is probléma, csak egy programot ugyanazokon az adatokon még egyszer kellene futtatni.
   Ha azonosítottuk a tranzakciót, amelyet helyre kell állítani, akkor három opció áll rendelkezésünkre:
   - point-in-time helyreállítás: Azonosítsunk minden olyan adatbázis-objektumot, amelyre az alkalmazás hatással volt, és végezzünk hagyományos point-in-time helyreállítást, hogy eltávolítsuk a tranzakciók hatását. Aztán manuálisan futtassuk újra az érvényes munkákat. Azaz nem SQL szkript segítségével, hanem ugyanazon a módon, ahogy először végre lett hajtva, például ugyanazokkal az értékekkel újrafuttatjuk az alkalmazásokat.
   - UNDO helyreállítás: csak a rossz tranzakció hatásait távolítjuk el. Egyszerű SQL alapú tranzakció helyreállítás, csak SQL utasításokat foglal magában. Az UNDO helyreállítás végrehajtásához az adatbázisnaplót kell végigvizsgálni a tranzakció azonosításához és egy anti-SQL utasításokat kell generálni: az INSERT-ből DELETE legyen, a DELETE-ből INSERT, az UPDATE-t fordítsuk meg.
     Ha az anti-SQL utasítások generálva vannak, SQL szkriptként lehet futtatni. Ehhez online adatbázis szükséges. Az UNDO helyreállítás alatt is történhet rendszerleállás és az UNDO tranzakció is hibázhat, amelyet egyszerűen visszagörgethetünk, mert SQL utasításokból áll.

A 9-3-as ábrán az UNDO helyreállítás folyamatát figyelhetjük meg.



9-3. ábra UNDO helyreállítás

- REDO helyreállítás: minden tranzakciót eltávolítunk, amely egy adott időpont után történt és csak a jó tranzakciókat alkalmazzuk újra. A megmenteni kívánt tranzakcióhoz generálunk SQL-t. Egy szokásos point-in-time tranzakciót végzünk, hogy eltávolítsunk minden tranzakciót az adott időpontig, majd a jó tranzakciókat az SQL-szkript segítségével újraalkalmazzuk.
  A 9-4-es ábrán az REDO helyreállítás folyamatát figyelhetjük meg.



9-4. ábra REDO helyreállítás

4. Ha a mentéshez és helyreállításhoz tároláskezelő szoftvert használunk, az adatbázis-objektumokhoz nem létezik önálló képmásolati mentés. Ebben az esetben a tároláskezelő szoftvert kell alkalmazni a helyreállítás végrehajtására. Az aktuális helyreállítási eljárás a használt tároláskezelő szoftver típusától függ, és attól hogy az az adatbázis-kezelő rendszer helyreállítási mechanizmusával hogyan dolgozik együtt.
5. Az offsite katasztrófa-helyreállítás a legritkább, de a legátfogóbb típusa az adatbázis-helyreállításnak. Egy offsite katasztrófa-helyreállítás akkor szükséges, ha egy természeti katasztrófa vagy valamilyen más baleset lehetetlenné teszi az elsődleges adatfeldolgozó központ működését. Ebben az esetben a teljes adatbázis-környezetet kell újra felállítani, és helyre kell állítani az adatbázis-kezelő rendszert, az adatbázis-objektumokat és az adatokat.

Optimális helyreállítási stratégia választása

Eredetileg a helyreállítást katasztrófák és hardverhibák esetén kellett legtöbbször elvégezni, de ez ma már nem igaz. A helyreállításokat ma legtöbbször alkalmazásproblémák miatt kell végezni. Ma a rendszerleállásokat inkább a szoftver, mint a hardverhibák okozzák.
Valójában nagyon kevés adatbázis-adminisztrátornak kell a tesztek kivételével igazi katasztrófa-helyreállítást elvégeznie. Ma a médiahiba is ritka. A felhasználói hibák és az alkalmazáshibák a legáltalánosabb okai az adatbázis-helyreállításnak és így az elsődleges okai a rendszer elérhetetlenségének.
Szoftverproblémákat és szoftverhibákat olyan tranzakciók okoznak, melyek hibásak, vagy javításra szorulnak. Az adatbázis méretben és összetettségben általában nő, így a lehetőség is, hogy a rossz tranzakciók olyan adat hibát okoznak, amelytől a cég üzleti folyamatai függenek. A tranzakció-helyreállítás válasz lehet az elérhetőségi problémákra, de sok esetben a tranzakció-helyreállítást nem lehet elvégezni vagy nem tanácsolható a végrehajtása. A végrehajtandó helyreállítás típusát, az

adatbázis-adminisztrátor határozza meg. A döntést a következő kérdésekre adott válaszok segíthetik:

- Tranzakció azonosítása: Minden problémás tranzakciót lehet azonosítani? A helyesen elvégzett munkát újra végre lehet hajtani?
- Adatintegritás: Más módosította a sorokat, mióta a probléma történt? Ha igen, folyamatban van még a módosítás? Elérhető még minden szükséges adat? Azóta történt újraszervezések, betöltések, vagy sok adat törlése szükségessé teszik egy képmásolati mentés használatát? A helyreállítás okozza más adatok elvesztését? Ha igen, lehet azonosítani az elveszett adatokat valamilyen módon és újralétrehozni azokat?
- Sebesség: Ha több technika érhető el, melyik a leggyorsabb? Szükséges inkrementális mentéseket használni? Mennyi adatbázisnapló szükséges a helyreállításhoz? Lehet bármit tenni a naplórekordok számának csökkentése érdekében, mint például a teljes, az inkrementális mentéseket és az adatbázisnaplót egyesíteni?
- Elérhetőség: Milyen hamar érhető el ismét az alkalmazás? Elkerülhető az offline mód?
- Káros hatás: A hiba mennyire hat az adatbázisra? Születtek döntések a rossz adatokra alapozva? Lehet a helyettesítő munkában bízni?

Ezeknek a kérdéseknek a segítségével valójában azt határozhatjuk meg, hogy mennyi a munka ára, azaz hogy meg lehet-e határozni, hogy mit kell újra elvégezni. Az a cél, hogy a helyreállítandó adatokat minél gyorsabban megtaláljuk, és a lehető legrövidebb helyreállítási mód segítségével helyreállítsuk azokat.

Sok tényező befolyásolhatja a helyreállítási folyamat időtartamát. A következő tényezők rövidíthetik meg az időtartamot:

- Minél kisebb a komponensek mérete, amelyeket helyre kell állítani, annál rövidebb a helyreállítási folyamat. Általában minél kevesebbet kell elvégezni, annál gyorsabban megy.
- Vegyük figyelembe az adatbázis-objektumok partícióit, és a partíciós szintek mentését és helyreállítását. Előfordulhat, hogy egy egész objektumra hatással levő hiba valójában egy-egy partícióra korlátozható.
- Használjuk a lemezen lévő képmásolati mentéseket és archivált naplóállományokat. A lemezen lévő állományok elérése gyorsabb és a feldolgozáshoz nincs szükség a szalagok cseréjére várni. Gyorsabb, ha lemezt használunk szalag helyett a helyreállításnál. Ha szükséges, akkor a helyreállítás előtt inkább másoljuk fel a szalagról a lemezre a mentéseket és a naplóállományokat.
- Teszteljük a képmásolati mentéseket, hogy érvényesek-e. Egy érvénytelen képmásolat használata meghosszabbítja a helyreállítási folyamatot.
- A mentés és helyreállítás automatizálása eltávolítja a manuális hibákat, így csökkenti a leállás időtartamát.
- Ha lehet, az adatbázistervet minél kevesebb függőséggel tervezzük meg. Az önálló adatbázis-objektumok a helyreállítást minimalizálhatják, mert kevesebb kapcsolódó adatbázis-objektumot szükséges egyszerre helyreállítani.

A különböző hibákhoz megfelelő típusú helyreállítás használata

Médiahiba esetén általában az utolsó lehetséges időpontig történő helyreállítást használjuk. Médiahiba esetén a hibás médián található adatbázis-objektumokat nem lehet elérni vagy módosítani. Az adatbázis-adminisztrátor a médiahiba bekövetkezte előtti időpontra kell, hogy helyreállítson minden, az adott médián található adatbázis-objektumot.

Tranzakcióhiba esetén point-in-time helyreállítást vagy tranzakció-helyreállítást kell végezni. A tranzakció-helyreállítást leginkább hibás programfutás miatt alkalmazzuk. A nem megfelelő futásból származó adatbázis-változásokat el kell távolítani minden olyan adatbázis-objektumból, amelyre hatással volt.

Adatbázis-kezelő rendszerbeli hiba vagy alrendszer hiba után az utolsó lehetséges időpontra állítunk

helyre. Az ilyen helyreállítás célja, hogy minden adatbázis-objektum adatát a hibapontot megelőző konzisztens állapotra állítsuk helyre.

Indexek helyreállítása

Mint, ahogy már az Adatbázismentés című fejezetben is említettük, két lehetőség van az indexek helyreállítására: az indexeket újraépítjük az adatokból vagy a mentésből helyreállítjuk. Az utóbbi eset természetesen csak akkor működik, ha az indexeknek létezik mentése.
Általában minél több adatot kell indexelni, annál nagyobb lesz az index és annál tovább tart újra felépíteni az adatokból. Ezért érdemesebb ilyenkor mentést és a helyreállítást választani.
Ha az indexek mentését és helyreállítását választjuk, akkor a mentéskor az indexeknek az adatokkal szinkronban kell lenniük, egyébként helyreállítás után integritási problémáink lehetnek.

A helyreállítási terv

Az adatbázis-adminisztrátornak minden adatbázis-objektumra vonatkozóan kell elkészítenie a helyreállítási tervet. A helyreállítási terv leírja, hogy a különböző típusú hibák, mint például hardverhiba, médiahiba vagy helyi katasztrófa esetén milyen helyreállítási eljárásokat kell használni.
A helyreállítási terv kialakításához az adatbázis-adminisztrátornak a következő szempontokat kell figyelembe vennie:

- A helyreállítási terv minden részét le kell írni, minden lehetséges hibához dokumentálni kell az egyes lépéseket.
- Minden adatbázis-objektum minden mentési és helyreállítási szkriptjét tartalmaznia kell.
- A tervnek minden olyan személyt tartalmaznia kell, aki a helyreállítás megvalósításkor hívható, illetve akik a helyreállításban részt vehetnek. Ezeknek a személyeknek legyen ott a neve, telefonszáma, illetve egyéb elérhetősége.
- Tartsuk a helyreállítási tervet naprakészen, azaz minden adatbázisobjektum-módosítás, vagy új adatbázis-objektum létrehozása esetén a helyreállítási terv is módosuljon.

A helyreállítási terv tesztelése

A helyreállítási tervet gyakran kell tesztelni. Teszteljük a helyreállítási eljárásainkat egy-egy adatbázis-objektumon, az adatbázison és a teljes adatbázis-rendszeren is. A teszteléssel az adatbázis-adminisztrátor azt biztosítja, hogy a mentési és helyreállítási szkriptek megfelelően működnek és hogy minden adatbázis-objektum helyreállítható.
A tesztelés legyen egy tréning arra az esetre, amikor egy elkerülhetetlen termelési adatbázis helyreállításra lesz szükség. A teszteléssel az adatbázis-adminisztrátor tulajdonképp gyakorolja a helyreállítást, jobban megismeri a helyreállítás végrehajtásához szükséges eszközöket és szkripteket, így a stresszes valódi helyreállításban is jól megállja majd a helyét.

Eldobott adatbázis-objektum helyreállítása

Lehetséges, hogy valaki nem szándékosan eldob egy olyan adatbázis-objektumot, amely az üzlet szempontjából fontos. Amikor az adatbázis-adminisztrátor felfedezi az eldobott objektum tényét, akkor olyan gyorsan helyre kell állítania az adatbázis-objektumot, amilyen gyorsan csak lehet. Így elkerülheti az elérhetőségi és integritás problémákat.
Egy eldobott objektum helyreállítása extra kérés egy normál helyreállításhoz képest. Az adatbázis-kezelő rendszertől és az elérhető eszközöktől függően nagyon bonyolult feladat is lehet.
Ha logikai mentésünk van akkor adatbázis-adminisztrátor szkriptek alapján újralétrehozza az adatbázis-objektumot, majd a logikai mentésben lévő adatokkal újra feltölti azt. Itt olyan probléma

merülhet fel, hogy a logikai mentés óta az adatbázis-objektumban történhettek változások. Ha nem akarjuk, hogy ezek a változások elvesszenek, akkor az adatbázisnapló alapján újra végre kell őket hajtani. De újra ne töröljük le az objektumot.
Az adatbázis-adminisztrátornak arra is ügyelnie kell, hogy az adatbázis-objektum eldobásával más is eldobódik: jogosultságok, hivatkozási megszorítások, ellenőrző megszorítások. Ha helyreállítjuk az eldobott adatbázis-objektumot, akkor ezeket is helyre kell állítani.
Az újabb adatbázis-kezelő rendszerek már rendelkeznek olyan beépített eszközökkel, amelyek segítésével az eldobott adatbázis-objektumokat egyszerűen, az adatbázisnapló segítségével helyre lehet állítani.

Tesztadatbázis létrehozása

Egy tesztkörnyezet felállításának a legjobb módja, hogy a termelési adatbázisról készítünk mentést. Egy ilyen mentés tartalmazza a termelési rendszer adott időpillanatbeli adatainak a halmazát. A tesztkörnyezet létrehozásához használjuk a helyreállítási segédprogramot. Azonban ügyeljünk rá, hogy az üzletileg kényes adatok ne kerüljenek át a termelési környezetből a tesztkörnyezetbe.

Összefoglalás

A fejezetben azt részleteztük, hogy ha valamilyen hiba történik az adatbázisban, akkor hogyan lehet az adatbázisban egy előző pillanatban létező adatokat helyreállítani. Azonban a legtöbb helyreállítás esetén nem a teljes adatbázist szükséges helyreállítani, hanem annak csak egy jól körülhatárolható részét. A probléma felfedezése után rengeteg kérdés megválaszolásával juthatunk el oda, hogy meghatározzuk, hogy valójában mi a hiba oka, mit kell helyreállítani, és hogyan kell helyreállítani.
Felsoroltuk a helyreállítás általános lépéseit, majd részleteztük, hogy milyen helyreállítási típusok léteznek: helyreállítás az utolsó lehetséges időpontra, point-in-time helyreállítás, tranzakció-helyreállítás (hagyományos értelemben és alkalmazás-helyreállításként), tároláskezelő szoftver segítségével történő helyreállítás és offsite katasztrófa helyreállítás. Az optimális helyreállítási stratégia kiválasztásához adtunk útmutatót, illetve megnéztük, hogy a különböző típusú hibákhoz milyen helyreállítás javasolható.
Beszéltünk az index helyreállításáról, a helyreállítási tervről, a terv teszteléséről, az eldobott adatbázis-objektumok helyreállításáról, és a tesztadatbázis létrehozásáról.

Ellenőrző kérdések

1. Milyen kérdéseket tegyünk fel, amikor az adatbázisbeli hiba okát és terjedelmét keressük, amelyből megállapíthatjuk, hogy milyen lépéseket kell tennünk a helyreállításhoz?
2. Milyen általános lépései vannak a helyreállításnak?
3. Milyen helyreállítási típusokról beszéltünk?
4. Mit jelent a point-in-time helyreállítás?
5. Mit jelent a tranzakció-helyreállítás?
6. Mit jelent a hagyományos tranzakció-helyreállítás?
7. Mit jelent az alkalmazás-helyreállításhoz szükséges tranzakció-helyreállítás? Hogyan lehet megvalósítani?
8. Mit jelet az, hogy az utolsó lehetséges időpontra állítunk helyre?
9. Napjainkban milyen típusú hibák fordulnak elő a leggyakrabban?
10. Milyen kérdéseket tegyünk fel, hogy megtaláljuk az optimális helyreállítási módot?
11. Milyen tényezők rövidíthetik le a helyreállítás időtartamát?
12. Hogyan állítsuk helyre az indexeket?
13. A helyreállítási tervet milyen szempontok alapján hozzuk létre?

14. Hogyan lehet egy véletlenül eldobott adatbázis-objektumot helyreállítani?
15. Mi a legegyszerűbb módja egy tesztadatbázis felépítésének?

10. fejezet – Katasztrófa-helyreállítási terv

A cég életében katasztrófáról nem csak akkor beszélünk, ha földrengés, árvíz vagy más hasonló természeti katasztrófa következik be, hanem olyankor is, amikor olyan esemény történik, ami csak a cég életére van hatással, mint például tűz a szerverszobában.
Egy cég életét tekintve a katasztrófa fogalmát a következőképpen tudjuk jól megfogalmazni: A katasztrófa egy nem tervezett, kiterjedt veszteség az üzletkritikus alkalmazások területén, mely a számítógép feldolgozó kapacitásának hiánya miatt történik.
A katasztrófa esetére összeállított helyreállítási tervezés egy előkészületi folyamat, arra az esetre, ha a cégnél valamilyen katasztrófa történik, amely a cég tulajdonát vagy műveleteit károsítja. Minden cégnek egy átfogó és tesztelt tervvel kell rendelkeznie, hogy egy katasztrófahelyzettel megbirkózzon. Ha egy cégnek nincs terve a katasztrófa utáni helyreállításra, akkor esetleg egy gyenge katasztrófa esetén sem tud visszatérni az üzleti életbe.
A helyreállítás tervezésének az alapja az, hogy a katasztrófa milyen hatással lesz a cég üzleti életére. Fel kell ismerni a lehetséges katasztrófákat, és megérteni azoknak a lehetséges következményeit.
Az adatbázis katasztrófa-helyreállítása egy lényeges része a teljes üzleti helyreállítási tervnek. Egy katasztrófatervnek mindenre ki kell terjednie, ki kell jelölni egy új helyet a cég üzleti folyamatainak a folytatásához, amit meg kell mondani az alkalmazottaknak, az ügyfeleket értesíteni kell, hogy a katasztrófa után hogyan végezhetik a cégnél az üzleti tevékenységüket. Az adatbázis-adminisztrátornak azonban csak az adatbázis és az adatbázis-kezelő rendszer helyreállítására kell koncentrálnia. Arra kell felkészülnie, hogy az új helyen új hardvert használva újra fel kell építenie a teljes adatbázist.

Kockázat és helyreállítás

A katasztrófa-helyreállítási terv célja az, hogy a lehető legnagyobb mértékben minimalizálni lehessen azokat a költségeket, amelyek abból származnak, hogy a cég IT szolgáltatásainak a kapacitása és erőforrásai elvesztek vagy károsodtak. Egy jó adatbázis katasztrófa-helyreállítási tervben elég jól meg vannak határozva az adatveszteséggel járó kockázatok, és elég jó becslést adtak arra, hogy milyen hatása lesz a cég üzleti életére az, ha a cég adatai elvesznek.
Hasonlóan, mint a lokális adatbázismentésnél és helyreállításnál, a katasztrófa-helyreállítás esetén is az adatbázis-adminisztrátornak az egyes adatbázis-objektumokhoz egy becslést kell végeznie a kritikusság alapján és az alapján, hogy az adat mennyire gyakran változik.
Az adatveszteséggel kapcsolatban három kockázati kategóriát különböztethetünk meg: a pénzügyi veszteség, az üzleti szolgáltatás szünetelés, és a jogos kötelezettség elmulasztása (pl. amikor a cég egy szerződésben foglaltakat nem tud teljesíteni). A kategóriákon belül a kockázatoknak más-más foka lehet. Az egyes alkalmazások elérhetetlensége más hatással lehet a cég egyes részeire.
A katasztrófa-helyreállítási terv elkészítése alatt tartsuk szem előtt, hogy az üzlet a legfontosabb, és csak utána jöhetnek a technikai szükségletek. A rendszert csoportosítsuk kritikus és nem kritikus alkalmazásokra, a csoportokat az üzleti igények alapján határozzuk meg. A rendszerek kritikusságának besorolását nem az adatbázis-adminisztrátor végzi, hanem egy vagy több olyan magasabb beosztású dolgozó a cégnél, akik átlátják ezeket a rendszereket.
A következő csoportok jöhetnek létre:

- Nagyon kritikus alkalmazások: ezeket kell először helyreállítani. A katasztrófa bekövetkezte előtti lehető legkésőbbi adatok visszaállítását fogja követelni. Próbáljuk ezen alkalmazásoknak a számát minimálisra csökkenteni.
- Üzletkritikus alkalmazások: egy-két nap múlva is elég helyreállítani. Pl. telefonszolgáltatónál a telefonos hálózat működéséhez szükséges alkalmazások nagyon

kritikusak, de a számlázási rendszer üzletkritikus.

- Kritikus alkalmazások: egy-két hét múlva is elég helyreállítani. Pl. a cég dolgozóinak a nyilvántartása, amely adatbázis alapján a juttatásaikat számolják.
- Szükséges alkalmazások. Itt olyan technikai alkalmazásokra lehet gondolni, amelyek megkönnyítik a napi munkát.
- Nem szükséges alkalmazások: ez az a kategória, ahol feltehetjük azt a kérdést, hogy miért is léteznek egyáltalán ezek az alkalmazások.

Az adatbázis-adminisztrátornak kell a tervet elkészíteni. Mindig a legkritikusabb alkalmazás és annak az adatainak helyreállításával kell kezdenie.
Az alkalmazások kritikussági szintjének megállapítása nem az adatbázis-adminisztrátor, hanem az üzlet feladata.

Általános katasztrófa-helyreállítási irányelvek

A katasztrófa utáni helyreállításnál a cél a leállás és az adatveszteség minimalizálása. Ne feledjük, hogy az adatbázis-kezelő rendszer és adatbázis helyreállításának a terve csak egy része a teljes katasztrófa-helyreállítási terv elkészítésének. Az adatbázis-adminisztrátor által elkészített, az adatbázis helyreállítására vonatkozó tervet be kell tudni illeszteni a cég teljes katasztrófa-helyreállítási tervébe.

A távoli oldal

Szükség van egy olyan távoli helyre, ahol a cég adatainak másolatát, mentését tárolni lehet. Ennek a helynek olyan messze kell lennie az elsődleges helytől, hogy egy természeti katasztrófa esetén a két helyszín közül legalább az egyik ne sérüljön. Ha a cég elég nagy és két adatközpontja van, akkor az egyik központ mentéseinek tárolására a másik központ alkalmas lehet. Más cégek a távoli helyet csak a mentés tárolására használják, az adatokat rendszeresen odaküldik. Egy katasztrófa esetén az itt található mentéseket használják.
A mentés tárolásának a helyszíne és a helyreállítás helyszíne különbözhet egymástól. Ideálisan a mentést a helyreállítási oldalon kellene biztonságosan tárolni. Ha a két helyszín nem egyezik, és katasztrófa történik, akkor a helyreállításhoz szükséges adatokat a helyreállítás helyszínére kell mozgatni valamilyen módon, ami időbe telik.

Az írott terv

Az írott terv minden jó katasztrófa-helyreállítási tervnek az alapja. Minden, a helyreállításban részt vevő kulcsembernek meg kell kapnia. A legjobb, ha a kulcsemberek a cégnél is, és a cégen kívül (pl. otthon) is tartanak belőle egyet. A helyreállítási helyszínén azonban mindenképp tárolni kell belőle legalább egy példányt.
A sok példány miatt nagy kihívás a tervet naprakészen és relevánsan tartani. A mindennapi tevékenységnek része kell, hogy legyen a terv karbantartása. Új objektumok, alkalmazások esetén, vagy régi objektumok eldobása esetén a tervnek is változnia kell.
Ha a terv változik, a régit semmisítsük meg, és cseréljük le az újra.
A tervnek bizonyos feltételeknek meg kell felelnie:

- Explicit műveleteket tartalmazzon, hogy mit kell tenni, ha a katasztrófa bekövetkezik.
- A műveletek megfelelő sorrendben kövessék egymást.
- Határozzuk meg az eszközt, amivel dolgozni kell és a szükséges pontos mentési információt.
- Dokumentálja, hogy a szükséges információk hol vannak tárolva és hogy lehet őket elérni a helyreállítás helyéről.
- A tervet úgy kell elkészíteni, hogy más képzett szakember is eligazodjon a terven, ne csak az aki ismeri. A helyreállítás munkáját ne akadályozza az, hogy a tervet ismerő

adminisztrátorok nem érhetőek el.

A katasztrófa-helyreállítási tervben minden, a helyreállításban szereplő résztvevőnek szerepelnie kell, ami nem csak az adatbázis-adminisztrátorokat, rendszerprogramozókat jelenti, hanem a főnököket, a végfelhasználókat, de akár az ügyfeleket is.

A tervnek a következőket kell tartalmaznia:

- helyszínek: a távoli helyszínek címének listája, telefonszámmal, e-mail címmel, fax számmal. Esetleg közeli hotel címekkel, megközelítés lehetőségével, mindezek árával, stb.
- személyek: a helyreállítási csapat neveinek és elérhetőségeinek listájával.
- hitelesítés: a helyreállításhoz szükséges biztonsági hitelesítések listája, és a személyek, akikhez az rendelve van.
- helyreállítási eljárások és szkriptek minden rendszerszoftverhez, alkalmazáshoz és adathoz: teljes eljárások, amelyek a megfelelő sorrendben, lépésről lépésre részletezve vannak. A távoli helyen legyen ott minden szoftverhez, és a szoftvereknek minden karbantartásához a telepítőprogram. A dokumentációban írjuk le, hogy pontosan hol vannak, vagy ha nincsenek a távoli helyen, akkor hogyan lehet őket elérni.
- riportok: lista a mentési szalagokról, azoknak a tartalmáról, arról, hogy mikor küldték el az elsődleges helyszínről, illetve mikor érkezett meg. A névkonvenciók leírása is legyen benne.
- Írjuk bele a tervbe azt is, hogy a cég üzleti élete alapján mi a legfontosabb cél a helyreállítás során. Nem mindegy, hogy egy esetleg adatveszteséggel rendelkező adatbázist állítunk nagyon gyorsan helyre vagy lassabban egy olyan adatbázist, amelyben nincs adatveszteség.

A katasztrófa-helyreállítási terv tesztelése

Próbáljuk ki a helyreállítási oldalon, hogy a terv működik-e. A terv tesztelését legalább évente egyszer el kell végezni. Akkor is érdemes tesztelni a katasztrófa-helyreállítási tervet, ha a következők történnek:

- jelentős változások a napi műveletekben
- rendszer hardverkonfigurációjának a változása
- adatbázis-kezelő rendszer frissítése vagy rendszerszoftver frissítése
- a helyreállításért felelős személy megváltozása
- az elsődleges adatközpont költözése
- változás a napi mentési eljárásokban
- új alkalmazások létrehozása vagy meglévő üzletkritikus alkalmazások frissítése
- jelentős adatmennyiségbeli vagy a napi tranzakciómennyiségbeli növekedés

A tesztelést érdemes arra is használni, hogy megtaláljuk és kijavítsuk a hibákat és a gyengeségeket a tervben. A tesztelés legyen életszerű, kezdjük a főnöknél, hogy tudja, kit kell keresni, ha baj van.

Személyzet

A katasztrófa-helyreállítási terv megtervezéséhez és kidolgozásához a megfelelő csapatot kell összeválogatni. A csapattagoknak az egész rendszert ismerniük kell, illetve IT és üzleti ismeretekkel is rendelkezniük kell.

Az elkészült katasztrófa-helyreállítási tervbe aztán minden, a helyreállításban érintett személyt be kell avatni.

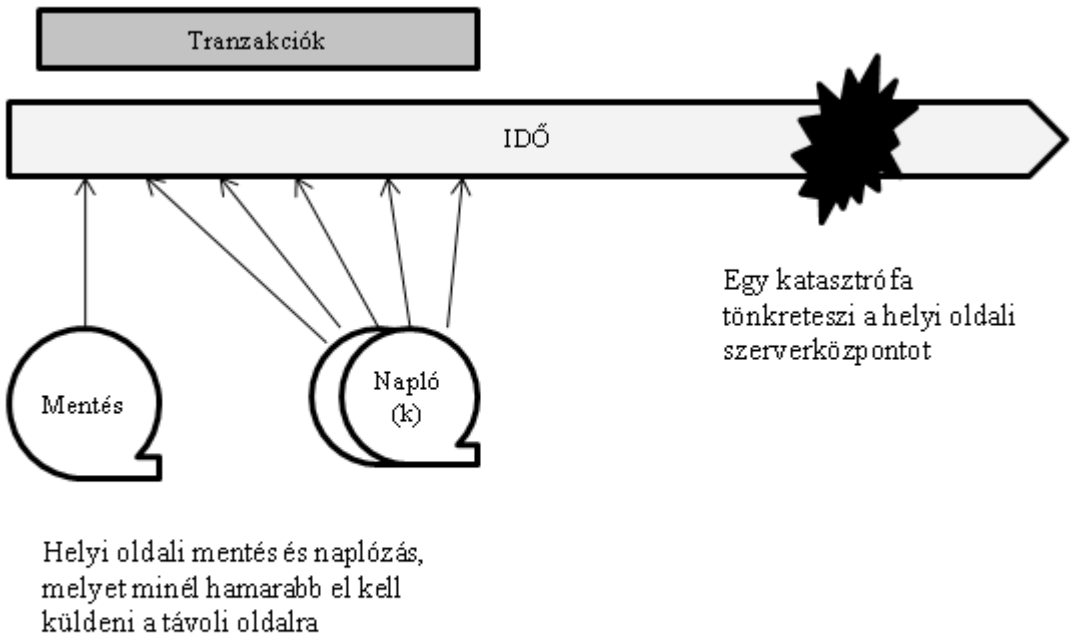
A terv elkészítésénél arra is oda kell figyelni, hogy a jogosultságok megfelelően legyenek kiosztva. Ha a helyreállító személyzetnek nincs megfelelő jogosultsága, akkor nem fog tudni dolgozni, nem tudja elvégezni a helyreállítást.

Az adatbázismentés a katasztrófa helyreállításhoz

A katasztrófa-helyreállítási eljárások nagyban függenek a mentési módszertől.

Képmásolati mentés

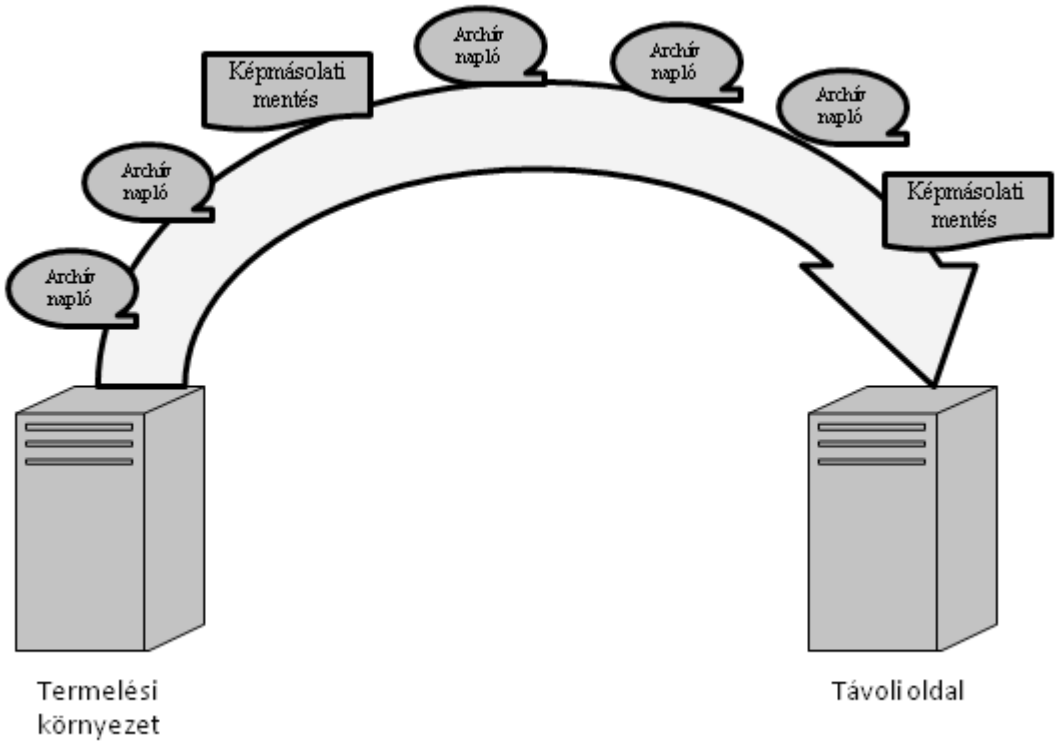
Hozzunk létre több output állományt a képmásolás mentési folyamat során és küldjünk legalább egy másolatot a katasztrófa-helyreállítási oldalra. Legyünk biztosak, hogy a létrehozás után olyan gyorsan odaér, amilyen gyorsan csak lehet. Minél hamarabb odaér, annál kisebb az esélye annak, hogy a helyi oldalon vagy az út során a helyi központot elpusztító katasztrófában tönkremegy.
Általában az indexeket nem kell menteni, azokat újra létre lehet hozni.
Legyünk biztosak, hogy a távoli oldalra elküldött minden mentési állományról készült riport. A riportról tartsunk egyet a helyi és a távoli oldalon is. A riportnak minden, a helyreállításhoz szükséges információt részleteznie kell, beleértve az állománynevet, a mentés típusát (teljes vagy inkrementális), hogyan készült a képmásolat (adatbázis-segédprogrammal vagy állomány-rendszerbeli programmal), a mentés napját és idejét, a távoli helyre történő átküldés napját és idejét, és az érkezés napját és idejét.
Az adatbázisnaplókat is menteni kell és át kell őket küldeni. Ha nem küldjük át az adatbázisnapló állományait a távoli oldalra, az azt jelenti, hogy az utolsó mentési időpontban lévő adatbázis-állapotot fogjuk tudni helyreállítani. Ha nincs mentve az adatbázisnapló, akkor az utolsó mentés óta történt változások mind elvesznek.
A 10-1-es ábrán a képmásolati mentés és a naplók archiválását láthatjuk, még a katasztrófa bekövetkezte előtt.



10-1. ábra Katasztrófa előtti mentés

A 10-2-es ábrán pedig azt, hogy ezeket a képmásolati mentéseket és az archivált naplóállományokat

folyamatosan küldeni kell a távoli oldalra.



10-2. ábra Távoli oldalra küldés

A helyreállítás után az adatbázis-adminisztrátornak az adatintegritást ellenőriznie kell.
A távoli oldalon az egyes adatbázis-objektumokhoz tartozó mentési generációkból legalább az utolsó hármat tartsuk meg. Ez biztosítja azt, hogy ha valami történik az egyik mentési generációval, akkor még van kettő, amikor az adatbázist helyre lehet állítani.

Mentés tároláskezelő szoftver segítségével

Ebben az esetben nem az adatbázis-kezelő rendszer felügyeli a mentést, hanem a tároláskezelő rendszer. A mentési konzisztencia érdekében a mentés elvégzéséhez az adatbázis-kezelő rendszert le kell állítani. A következő lépésekből áll:

1. Állítsuk le az adatbázis-kezelő rendszert, hogy a helyreállításhoz egy rendszerszintű szilárd pontot kapjunk.
2. Másoljunk le minden adatbázis-objektumot a tároláskezelő szoftver használatával azaz mentsük el a lemez teljes tartalmát egy másik tárolóeszközre, ami lehet pl. szalag.
3. Ha a másolás sikerült, indítsuk újra az adatbázis-kezelő rendszert.
4. A lementett adatokat küldjük el a távoli oldalra.

Ennek a megoldásnak a legnagyobb problémája az, hogy le kell állítani az adatbázist. Ez a mentés olyan cégeknél lehet hatékony, ahol nem okoz problémát, ha napi szinten leáll az adatbázis.

Más megközelítések a katasztrófa utáni helyreállításra

Napjainkban az interneten át nagyon könnyen meg lehet valósítani a mentés átküldését a távoli helyre. Ezért sok cég úgy készül fel a katasztrófára, hogy az adatbázisáról távoli tükrözést készít. Azaz az eredeti adatbázisban történt minden változás megjelenik a távoli adatbázisban is, esetleg

egy kis késéssel. Ha katasztrófa történik, a vezérlés egyszerűen átkerül a távoli oldali adatbázisba, amelyet tartalék adatbázisnak is hívhatunk.
Sok cég alkalmaz teljes hardver redundanciát. Ilyenkor két teljes tükörrendszer van, ami párhuzamosan fut. Ha az egyik rendszer nem alkalmazás hiba miatt áll le, akkor a másik rendszer észrevétlen módon veszi át a szerepét. A hiba elhárítása után szinkronizálni kell a rendszereket, és ismét futhatnak párhuzamosan.

Néhány irányelv a katasztrófa utáni helyreállításhoz

A helyreállítás sorrendje

Bizonyosodjunk meg róla, hogy az operációs rendszer és az adatbázis-kezelő rendszer a megfelelő verzióval és a megfelelő karbantartási szinten lettek telepítve mielőtt az adatbázis-objektumok helyreállítását megkezdenénk. Szigorúan kövessük az írott dokumentáció helyreállítási lépéseit.

Adatlappangás

Az a kérdés, hogy milyen régi az az adat, amelyet a távoli oldalra küldött mentés alapján helyre lehet állítani. Ha a mentések napi gyakorisággal, pl. éjszaka történnek, akkor előfordulhat, hogy a mentésből helyreállítható adatokon az utolsó változás 24 órával korábban történt. A legtöbb esetben elfogadhatatlan, hogy ilyen régi adatok alapján folytatódjon a munka. Azonban az adatmentés költsége lehet, hogy túl sok, ha 24 órán belül több mentés is van. Ennek a problémának az a megoldása, hogy az adatbázisnaplót is menteni kell, és át kell küldeni a távoli oldalra.
Az aktív adatbázisnapló-állomány illetve az utolsó archív adatbázisnapló-állományok közül néhány a katasztrófa időpontjában lehet, hogy nem érhető el, így a helyreállítási oldalon nem lehet alkalmazni. Emiatt néhány adat lehet, hogy nem teljesen állítható helyre. Minél hamarabb van a helyreállítási oldalon az adatbázismentés és az adatbázisnapló, az adatok pontossága annál jobb.

Alkalmazásokhoz kapcsolódó adatok archiválása

Ne feledkezzünk el a következő létfontosságú adatokról sem:

- DDL könyvtárak az adatbázis-objektumokhoz, a mentéshez és a tesztszkriptekhez
- alkalmazásprogramok forrásai és futtatható állományok
- tárolt eljárások forráskódjai
- felhasználó által definiált függvények forráskódjai
- futtatható állományok
- könyvtárak és jelszavak más szállítótól származó adatbázis-adminisztrátori eszközökhöz
- az alkalmazások által használt adatok

Tömörítés

Tömörítés esetén figyeljünk oda, hogy ugyanaz a tömörítő program legyen a betömörítéskor és a kitömörítéskor is. Ha nem ugyanaz, akkor a távoli oldalon a mentés olvashatatlanná válhat.

Helyreállítás utáni képmásolat készítése

Nagyon fontos része a katasztrófa-helyreállítási folyamatnak, hogy készítsünk egy képmásolati mentést az adatbázis-objektumokról miután helyre lett állítva a távoli oldalon. Ez egyszerűsíti az adatok helyreállítását, ha egy hiba következik be miután a feldolgozás elkezdődött a távoli oldalon. Az új képmásolati mentés nélkül a katasztrófa-helyreállítási eljárást kell újra végrehajtani, ha egy hiba történik, miután a távoli oldalon a feldolgozás elkezdődött.

Katasztrófák okozta károk megelőzése

A földrengés által okozott károkat nem lehet megelőzni, de áramszünet által okozott károkat igen: egy jó szünetmentes tápegységgel. Az ilyen jellegű katasztrófákat is tervezni kell, hiszen a szünetmentes tápegység kapacitása is véges. Elképzelhető, hogy fel kell készülni egy teljes, de szabályos leállásra.
Az emberi hibák megelőzéséért is tehetünk, mégpedig dokumentáció készítésével az adatbázishoz és az alkalmazásokhoz. Ezekkel a dokumentációkkal tanítani lehet az adatbázis felhasználóit: a programozókat, a végfelhasználókat, az új adatbázis-adminisztrátorokat arra, hogy mit csináljanak adott szituációkban, adott hibaüzenetek esetén.

Összefoglalás

A fejezetben arról van szó, hogy hogyan tud egy cég felkészülni arra az eseményre, ha valamilyen katasztrófa történik, amely az adatközpontot is megsemmisíti. Ehhez először definiáltuk a cég életére vonatkozó katasztrófa fogalmát. Tudnunk kell azt is, hogy egy ilyen katasztrófa nem csak az adatbázisra, hanem a teljes cég életére hatással van. Ezért a helyreállítási tervnek csak egy része az adatbázis-adminisztrációt érintő adatbázis-helyreállítás.
A magasabb pozícióban dolgozó alkalmazottak üzleti szempontok alapján kockázati kategóriákba sorolják be az alkalmazásokat. Az adatbázis-adminisztrátornak a helyreállítási terv elkészítésénél figyelembe kell vennie ezeket a kockázati kategóriákat, és a legkritikusabb alkalmazások helyreállításával kell kezdeniük.
A katasztrófa-helyreállítási irányelvek között beszéltünk a távoli oldal szerepéről, az írott tervről, arról, hogy az írott terv milyen feltételeknek feleljen meg, mit tartalmazzon az írott terv, beszéltünk a terv teszteléséről, és a tervhez kapcsolódó személyzeti kérdésekről.
A katasztrófa-helyreállítás mindig függ attól, hogy milyen mentés áll a rendelkezésünkre. Két fő mentési típust néztünk meg, a képmásolati mentést és a tároláskezelő szoftver segítségével történő mentést. Ezeknek kívül még egy alternatívát adtunk, a távoli tükrözést.
A fejezet végén pedig irányelveket soroltunk a katasztrófa utáni helyreállításhoz: a helyreállítás sorrendje, adatlappangás, alkalmazásokhoz kapcsolódó adatok mentése és helyreállítása, tömörítés, helyreállítás utáni képmásolat készítése.

Ellenőrző kérdések

1. Mit nevezünk egy cég életében katasztrófának?
2. Mi a legfontosabb célja a katasztrófa-helyreállítási tervnek?
3. Kockázat szempontjából milyen alkalmazáskategóriákat határoztunk meg?
4. Miért szükséges a távoli hely a cég életében? Egy cég mire használja a távoli helyet?
5. Milyen feltételeknek kell megfelelnie a katasztrófa-helyreállítási tervnek?
6. Mit kell tartalmaznia a katasztrófa-helyreállítási tervnek?
7. Az egyszer jól megírt katasztrófa-helyreállítási tervet kell változtatni? Miért?
8. Mikor kell tesztelni a katasztrófa-helyreállítási tervet?
9. Milyen mentési technikákból származó mentéseket lehet használni a katasztrófa-helyreállításhoz?
10. Hogyan kell a képmásolati mentéseket előkészíteni, hogy alkalmasak legyenek a katasztrófa-helyreállításra?
11. Mi a hátránya annak, ha a tároláskezelő szoftver segítségével történik az adatbázis mentése?
12. A képmásolati mentésen és a tároláskezelő szoftver segítségével történő mentésen kívül milyen más lehetősége van egy cégnek a katasztrófa utáni helyreállításra felkészülni?
13. Milyen irányelveket soroltunk a katasztrófa utáni helyreállításhoz?

14. Mit jelent az adatlappangás?
15. Meg lehet-e előzni a katasztrófák okozta károkat?

11. fejezet - Adatok elérhetősége

Ha az adatok nem érhetőek el, akkor a programok nem tudnak dolgozni. Ha a programok nem dolgoznak, akkor a vállalat üzleti veszteséget szenved. Az adatbázis-adminisztrátor felelős az adatbázis elérhetőségéért: vagyis azért, hogy az adatbázis online és működőképes legyen. Ha egy adatbázis webes alkalmazásokat szolgál ki, akkor a hét minden napján 24 órában elérhetőnek kell lennie.

Az elérhetőség meghatározása

Az *elérhetőség* azt jelenti, hogy az adott erőforrás rendelkezésre áll az ügyfelek számára. Ha egy adatbázis elérhető, az adatok felhasználói, vagyis az alkalmazások, az ügyfelek, és az üzleti felhasználók hozzáférnek az adatokhoz.
Az elérhetőséget egy százalékos értékkel jellemezhetjük, amely azt határozza meg, hogy egy adott időtartam hány százalékában használható a rendszer produktív munkára. Pl. a 99%-os elérhetőség azt jelenti, hogy 1%-ban elérhetetlen, azaz egy évben összesen kb. 87 óra (kb. 3 és fél nap) leállás lehetséges. Cégenként, cégen belül rendszerenként, de akár felhasználónként is változik, hogy az alkalmazásokhoz mennyi elérhetőség szükséges.
Az *adatbázis elérhetőség* és az *adatbázis-teljesítmény* két különböző dolog, amelyeket gyakran összekeverünk egymással, mert valóban vannak hasonlóságok a kettő között. A legnagyobb különbség abban rejlik, hogy a felhasználó hogyan tud az adatbázishoz hozzáférni. Gyenge teljesítményű adatbázishoz hozzá lehet férni, az elérhetetlen adatbázishoz azonban nem. Az alacsony teljesítményből akkor válik elérhetetlenség, ha a teljesítmény annyira lecsökkent, hogy az adatbázis felhasználói nem tudják a feladatukat ellátni. Az adatbázis-adminisztrátornak az elérhetőségi és a teljesítménybeli problémákat egymástól különállóként kell kezelnie, még akkor is ha egy súlyos teljesítménybeli probléma egy lehetséges elérhetőségi problémát is jelent.
Az elérhetőség négy különböző részből tevődik össze. A négy komponens együtt biztosítja, hogy a rendszer fut és a cégünk üzleti tevékenysége irányítható. Azt mondhatjuk, hogy a rendszer elérhető, ha kezelhető, visszaállítható, megbízható és szervizelhető. Nézzük, hogy mit jelentenek a komponensek külön-külön:
● *Kezelhető,* azaz olyan hatékony környezetet lehet létrehozni és fenntartani, amely szolgáltatásokat nyújt a felhasználók számára.
● *Visszaállítható,* azaz hiba esetén a szolgáltatást helyre lehet állítani.
● *Megbízható,* azaz egy adott időtartamra vonatkozóan a szolgáltatást meghatározott szinten lehet nyújtani.
● *Szervizelhető,* azaz a fennálló problémákat meg lehet határozni, meg lehet állapítani azok okait és ki lehet azokat küszöbölni.
Mind a négy komponensnek hatása van a rendszer, az adatbázis, és az alkalmazás általános elérhetőségére.

Megnövekedett elérhetőségi követelmények

Manapság egyre több cég követeli meg a teljes idejű rendszer-elérhetőséget. A leállás költsége egyre növekszik, így az üzleti szempontból kritikus rendszerek és szoftverek teljesítményének optimalizálására fordítható idő egyre csökken.
Ha a rendszeres karbantartási folyamatok nincsenek elvégezve, akkor a teljesítmény látja ennek kárát. Az adatbázis-adminisztrátornak egyensúlyt kell teremtenie a 24/7-es elérhetőség (azaz a hét minden napján 24 órában kell nyújtani az elérhetőséget) teljesítése és a rendszer karbantartásához szükséges leállás között.

Csökkenő karbantartási idő

A 24/7 órás rendszer-elérhetőség mellett az adatbázis-adminisztrátornak egyre nehezebb időt találni a rutin rendszerkarbantartások elvégzésére. Azoknak az adatbázisoknak, amelyek naponta nagyon sok tranzakciót hajtanak végre, időnkénti karbantartásra és újraszervezésre van szükségük, mert az állandó használattal az adatbázisok töredezetté válnak, az adatokhoz vezető útvonalak elvesztik hatékonyságukat, és csökken a teljesítmény. Az adatokat rendszerezett módon kell visszahelyezni, és a törlésekkel létrejött lyukakat meg kell szüntetni. A töredezettségmentesítéshez és az újraszervezéshez az adatbázist le kell állítani.

Döntéstámogatás, adattárházak

Sok cég a legfontosabb üzleti adatokat döntéstámogatásra is fel akarja használni. Ehhez az üzleti adatokat másolni kell különböző adatbázis-környezetek között, illetve elérhetővé kell tenni felhasználóbarát formában. Az operatív adatok elérhetőségét negatívan befolyásolják a döntéstámogatási kérések követelményei, mert az adatkinyerési folyamat alatt rengeteg operatív adat nem érhető el frissítésre. Az adattárházak növekedésével egyre több és több adatra van szükség az operatív adatbázisokból, amelyet egyre több folyamat fog kinyerni. Ezek a folyamatok lassítják az operatív adatbázist, és bonyolultabbá teszik az adminisztrálását is.

Állandó elérhetőség

A 24/7-es elérhetőség mellett használatos a 24/24 elérhetőség kifejezés is. Vannak olyan cégek, amelyek az összes időzónában bonyolítanak le műveleteket, és az adatoknak elérhetőnek kell lenniük azon felhasználók számára is, akik nem ugyanabban az időzónában dolgoznak, mint a működő adatbázis-kezelő rendszer.
Példák ilyen rendszerekre: Repülőgépjárat-foglalási rendszerek, hitelkártya-elfogadási funkciók, telefonos cégek alkalmazásai, tőzsde. Ezeknél a rendszereknél minden percben hatalmas mennyiségű pénz mozoghat, ezért a leállás egész egyszerűen nem engedhető meg.
Az adatbázis-adminisztrátornak és az IT szakembereknek olyan technikákra van szükségük, amelyek a karbantartást, a biztonsági mentést, és a helyreállítást az arra kijelölt idő alatt elvégzik.

Az állásidő költsége

Az állásidő költsége cégenként változik. Minden cégnek magának kell megbecsülnie az állásidő költségét, mely becslést az ügyfeleire, a rendszereire és az üzleti műveleteire alapozhatja.
A brókerek számára az állásidő katasztrófa. Más piaci területeken szolgáltató cégek számára, amelyek manuális rendszerek használatával vészelik át a leállást, az állásidő nem olyan nagy csapás. A leállás nyomot hagy minden cég üzleti tevékenységén és bármilyen méretű állásidő költséget jelent a cég számára. Mikor az állásidő költségét becsüljük, vegyük figyelembe a következőket is:
● a leállás során elvesztett üzletek
● bármilyen per jogi költségei
● a csökkenő részvényértékek
Az leállás negatívan befolyásolhatja a vállalat imázsát.
Néhány cég nem akar pénzt költeni a szoftvereire és a szolgáltatásaira, hogy fejlessze az elérhetőséget, mert nem ismerik az állásidő valódi költségét. Ez a gondolkodás akkor változhat meg, ha elkészül a becslés, amely az összes lehetséges kockázati tényezőhöz tartozó költséget tartalmazza.

Mennyi elérhetőség elegendő?

Az elérhetőséget alapvetően a teljes idő azon százalékában fejezik ki, mely alatt egy szolgáltatásnak

működnie kell. Például egy 99%-os elérhetőséggel rendelkező rendszer az idő 99%-ban elérhető lesz és működni fog míg 1%-ban lesz elérhetetlen. Az elérhetőség meghatározására szolgáló másik mérőszám az MTBF (mean time between failure), azaz a meghibásodások közötti átlagos időtartam. Pontosabban az MTBF a *megbízhatóság* mértéke, nem pedig az elérhetőségé. Azonban a megbízhatóságnak bizonyos hatása van az elérhetőségre, általában minél megbízhatóbb a rendszer, annál elérhetőbb is.
Az internet korában az adatbázisoktól azt várják el, hogy leállás nélkül működjenek, azaz az év 365 napján, a nap 24 órájában. Ez egy évben 525600 perces működési időt jelent. Azonban a tisztán 100%-os elérhetőséget elérni nem lehet. Az *öt kilences* kifejezést gyakran használják a magas elérhetőségű rendszerek leírására, ami 99,999%-os működést jelent. Az *öt kilences* olyan rendszereket ír le, amelyek alapvetően 100%-osan elérhetőek, de természetesen néhány leállás elkerülhetetlen. A 99,999%-os elérhetőség évi 5 perc leállást jelent, míg például a 99%-os évi 87,6 órát, azaz több mint 3 napot.
Néhány rendszer megvalósítja az *öt kilences* elérhetőséget. Annak ellenére, hogy egy cég nem követeli meg a magas elérhetőséget, az adatbázis-adminisztrátor még tehet lépéseket ebbe az irányba. De azért, mert a magas elérhetőséget egy rendszerrel meg tudjuk valósítani, még nem feltétlenül jelenti azt, hogy meg is kell valósítani. Egy magas elérhető rendszer jóval többe kerül, mint egy hagyományos rendszer, amelybe terveztek leállásokat is. Az adatbázis-adminisztrátor feladata felvilágosítani a végfelhasználókat, hogy egy magas elérhetőségű rendszer milyen költségekkel jár.
Ha egy új rendszerhez, adatbázishoz, vagy alkalmazáshoz cél a magas elérhetőség, akkor körültekintő elemzés szükséges annak meghatározására, hogy a felhasználók mennyi állásidőt viselnek el, és hogy a leállásnak milyen hatása lehet. A magas elérhetőséget a felhasználók nagyon szeretik, annyit szeretnének belőle, amennyit csak lehet. Az adatbázis-adminisztrátornak a feladata, hogy a követelmények realitását vizsgálja.
Az, hogy egy cég mennyi elérhetőséget valósít meg az adatbázisa esetében, a költségektől függ, azaz attól, hogy alkalmazás tulajdonosa mennyi elérhetőséget tud finanszírozni. Előfordulhat, hogy a magas elérhetőség valósítható, de nem költséghatékony, így a cég nem támogatja. Az adatbázis-adminisztrátornak proaktív módon kell az alkalmazás tulajdonosaival együtt dolgoznia, és teljes mértékben tisztázniuk kell az elérhetőséggel járó költségeket.

Elérhetőségi problémák

Alapvetően az adatbázis-adminisztrátor feladata az adatbázisok elérhetőségének biztosítása. Azonban előfordulhatnak olyan okok, amelyek túlmutatnak az adatbázis-adminisztrátor feladatain. Ekkor az adatbázis-adminisztrátornak meg kell találnia, hogy az ok melyik területen keresendő, és a megfelelő szakemberhez kell fordulni. Nézzük meg, hogy milyen okai lehetnek annak, ha az adatbázis nem érhető el:

- Az adatközpont elvesztése
- Hálózati problémák
- A szerver hardver (központi feldolgozó egység, memória) elvesztése
- Lemezzel összefüggő leállások
- Operációs rendszer meghibásodása
- Adatbázis-kezelő rendszer szoftverhibája
- Alkalmazási problémák
- Biztonsági és jogosultsági problémák
- Adatok sérülése
- Adatbázis-objektumok elvesztése, véletlen eldobása
- Adatvesztés

- Adatreplikálási és propagálási hibák
- Súlyos teljesítménybeli problémák
- Helyreállítási kérdések
- Adatbázis-adminisztrátor hibák
- Tervezett és nem tervezett leállások

Az elérhetőség biztosítása

Tekintsünk meg néhány olyan technikát, amelyek segítik az elérhetőség biztosítását.

Rutinkarbantartás működő rendszereken

Ha a rendszerünk teljesítményén szeretnénk javítani, miközben egyre kevesebb a cégnél az informatikai szakember, és egyre kisebb a költségvetés, akkor mindenképp be kell szereznünk néhány olyan terméket, amelyek leegyszerűsítik és automatizálják a karbantartási funkciókat. Az adatbázis-adminisztrátornak szüksége van olyan eszközökre, melyek órákról percekre vagy egy percen belülire csökkentik a karbantartási időt és a karbantartási feladatokat úgy tudják ellátni, hogy közben a felhasználók a munkájuk elvégzéséhez szükséges adatokhoz folyamatosan hozzáférnek. Néhány adatbázis-kezelő rendszer rendelkezik ilyen beépített eszközökkel. Az eszközök használatával az adatbázis elérhető marad, azaz a felhasználók olvasni és írni is tudják az adatokat, és az adatbázis-adminisztrátor is el tudja végezni a karbantartási feladatait. Az adatbázis integritása nem sérül a karbantartó programok használata során. Ha az adatbázis-kezelő rendszernek nincs ilyen eszköze, akkor vásárolhatunk más cégtől is ilyen terméket.
A következő feladatokhoz van a legnagyobb szükség olyan segédprogramokra, amelyek nem szakítják meg az adatbázis működését:
● Adatbázis újraszervezése a teljesítmény javításának céljából
● Adatbázismentés
● Adatbázis-helyreállítási megoldások, melyek a helyreállított adatokat úgy használják, hogy nincs szükség leállásra
● Adatkimentési és betöltési folyamatok a forrásadattár és az operatív adattár közötti adatmozgatásra döntéstámogatási rendszerek és adattárházak számára
● Statisztikagyűjtő segédprogramok, melyek elemzik az adatokat és statisztikát készítenek az adatbázis optimalizáló használatához
● Integritást ellenőrző segédprogramok a hivatkozási integritás és a szerkezeti adatintegritás ellenőrzésének a céljából
Vegyünk egy példát. Ha az adatbázis-adminisztrátor az adatbázist úgy akarja újraszervezni, hogy közben az adatbázis elérhető maradjon, illetve csak nagyon minimális mértékben ne legyen elérhető az adatbázis, akkor az nem megoldás, hogy leállítjuk az adatbázist, újraszervezzük, majd újraindítjuk. Helyette inkább hozzunk létre egy adatmásolatot, és szervezzük át a másolatot. Ekkor az olvasási és az írási hozzáférés az eredeti adatokon folytatódhat, majd amikor a másolat újraszervezése készen van, az újraszervező folyamat az adatbázisnaplókat használja, hogy az eredeti adatokon végzett módosításokat a másolaton alkalmazza. Amikor a másolat utolérte az eredetit, akkor a vezérlést áthelyezzük az új, másolati adatbázisba, azaz a másolatból lesz az eredeti, az eredetit pedig törölni lehet.
Ne felejtsük el, hogy a legtöbb adatbázis-karbantartó művelet hat az elérhetőségre. Az adatok biztonsági másolatai, az adatok helyreállítása, integritássértéseket ellenőrző adatellenőrzés, adatbázis-statisztika készítése, az új adatok betöltése az adatbázisba mind kedvezőtlenül hat az elérhetőségre. Amikor az adatbázist online állapotban, működés közben tartjuk karban, nagy hasznát vehetjük azoknak az eszközöknek, amelyek a legújabb tárolási eszközökkel együtt dolgozva a leállás minimalizálására és kiküszöbölésére szolgálnak. Néhány tárolási egység gyors pillanatfelvételt tud készíteni az állományokról. Ha az adatbázist karbantartó műveletek élnek e

technika adta lehetőségekkel, a leállást percekről vagy órákról másodpercekre le lehet csökkenteni.
A legtöbb adatbázis-karbantartás hatással van az elérhetőségre. A legnagyobb veszélyt az elérhetőségre nézve az adatbázis-változások jelentik. Az elérhetőségre való hatás függ a változás típusától és attól, hogy az adatbáziskezelő-rendszer hogyan valósítja meg a változást. Ha olyan egyszerű változásra van szükség, amelyhez az ALTER utasítást lehet használni, mint például bizonyos adatbázis konfigurációs paraméterek változtatása, akkor az, bár hatással van az elérhetőségre, mégis kisebb probléma, mert ezt a változtatást gyorsan végre lehet hajtani. Az adatbázis-objektumokon történő változásoknak, mint például egy táblabeli oszlop definíciójának a változtatása, nagyobb hatásuk van az elérhetőségre. Ha az adatbázis-objektum szerkezeti definícióját változtatjuk, akkor azt legtöbbször csak úgy lehet megtenni, hogy közben az adatbázis-objektumot a felhasználók nem érhetik el. Minél összetettebb a változás típusa, annál nagyobb a hatása az elérhetőségre. Bizonyos változások objektumok eldobását és adatok törlését igénylik. Nyilvánvaló, hogy egy ilyen változás leállást okoz. Minél hosszabb ideig tart a változás végrehajtása, annál nagyobb leállás lesz. A nagy sebességű LOAD (betöltő) és UNLOAD (kimentő) segédprogramok csökkenthetik a leállás időtartamát. A változás automatizálása tovább növelheti az elérhetőséget.

Adatbázis-adminisztrátori feladatok automatizálása

Ha az adatbázis-adminisztrátori eljárások egy részét automatizáljuk, az növelheti a teljes adatbázis elérhetőségét. Egy megfelelően létrehozott, automatizált folyamat ritkábban hibásodik meg, mint a manuálisan megvalósított utasítások halmaza. Az ember hibát követ el, így minél összetettebb a feladat, annál hasznosabb az automatizálás.
Az adatbázis-változások végrehajtása összetett feladat. Így jogosan feltételezhetjük, hogy a változások automatizálása javíthatja az elérhetőséget. Ha automatizált adatbázis-adminisztrátori eszközöket használunk, csökken az emberi hiba lehetősége. Sokkal több időbe telik az adatbázis-adminisztrátornak, hogy manuálisan írjon szkripteket a változásokhoz, mint egy változáskezelő eszköznek ahhoz, hogy generálja azokat. Ráadásul nem valószínű, hogy az eszköz hibát követ el. Tehát az adatbázis-változások automatizálásával kevesebb idő szükséges azok végrehajtásához, mint ahhoz, hogy az adatbázis-adminisztrátor a kívánt változásokat elemezze, hogy a változás végrehajtására a szkripteket fejlesszen, és, hogy az adatbázis-változásához a szkripteket futtassa. Természetesen a változások elemzése automatizált eszköz segítségével sem maradhat el, de az eszköz ebben a feladatrészben is nagy segítséget nyújthat.
Az automatizálásnak az adatbázis mentésénél és helyreállításánál vehetjük még nagy hasznát. Egy cégnél előre számítani kell arra, hogy a rendszer valamilyen hiba miatt leáll, vagy valamilyen katasztrófa történik. Ezekre az eseményekre fel kell készülni, vagyis az adatbázist megfelelő módon archiválni kell. A cél, hogy a hiba vagy katasztrófa bekövetkezte után a cég minél hamarabb helyreállítsa az adatait. A biztonsági mentéshez és helyreállításhoz való proaktív megközelítés segít abban, hogy a hiba vagy katasztrófa után a rendszer adatvesztés nélküli minimális leállással tudjon működni. Míg egy nem végiggondolt mentési-, helyreállítási stratégia esetén a cégünk esetleg soha nem tud talpra állni. A legtöbb adatbázisrendszer és alkalmazás minimálisan járul hozzá az automatizált biztonsági mentéshez és helyreállításhoz, illetve nem biztosítanak olyan funkciókat, melyek a proaktív helyreállítási tervezést támogatnák. Az adatbázis-adminisztrátornak olyan eszközökre van szüksége, melyek lehetővé teszik a gyakori mentéseket úgy, hogy azok közben az online rendszerre minimális hatást gyakoroljanak. A mentési és helyreállítási szoftver egy másik fontos követelménye, hogy nagyon gyorsan tudja helyreállítani az adatokat, mégpedig olyan sorrendben, ahogy azt az üzleti alkalmazások kritikussága meghatározza. Ha szükséges, az adatbázis-adminisztrátornak helyre kell tudnia állítani minden adatbázis-objektumot a rendelkezésére álló biztonsági mentések és adatbázisnaplók segítségével. Ez az adatbázis-kezelő rendszer, az alkalmazás, és a környezet ismeretét követeli meg, illetve megfelelően dokumentált mentési-, helyreállítási stratégiát igényel. Elérhetőek olyan eszközök, amelyek probléma esetén a

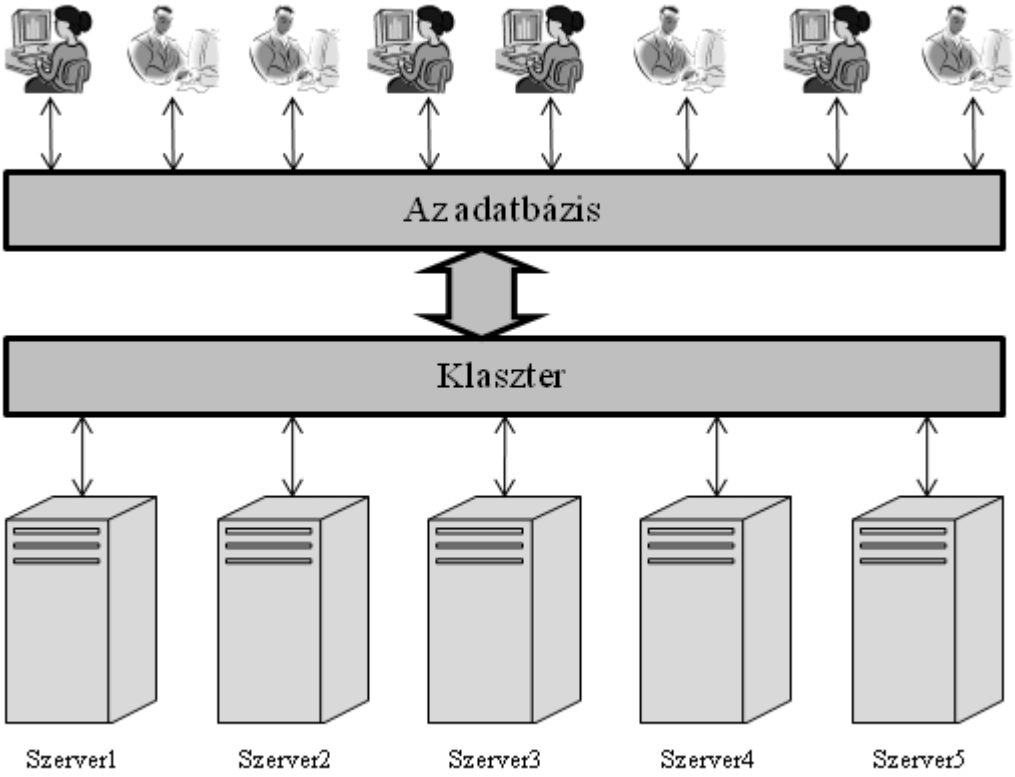
helyreállítás automatizálását segítik azáltal, hogy a helyzetet elemzik és olyan helyreállítási szkripteket készítenek, amelyekkel a rendszer rövid időn alatt helyreállítható.

A magas elérhetőségű eszközök kihasználása

Ha az adatbázis-kezelő rendszerünket úgy tervezték, hogy ki tudja használni a klaszteres megoldásokat vagy a párhuzamos adatfeldolgozást, akkor törekedjünk rá, hogy cégünk adatbázisainál is ki tudjuk használni ezeket a technológiákat. Már az adatbázis tervezésénél is gondoljunk arra, hogy ha nem is azonnal, de később be lehessen vezeti ezeket a megoldásokat. Az adatbázis-kezelő rendszerek általában képes a legújabb hardverbeli és operációs rendszerbeli képességeket is kihasználni, csak a cégünk pénztárcáján múlik, hogy valóban ki tudjuk-e őket használni.
A legtöbb adatbázis-kezelő rendszernek sok olyan eszköze van, amely az elérhetőséget támogatja. Két legalapvetőbb példa ezek közül: a segédprogramok és az adatbázis rendszerparaméterei. Segédprogramok futtatása, valamint az adatbázis rendszerparamétereinek a megváltoztatása régen olyan feladatok voltak, amelyek a legtöbb esetben leállást igényeltek. Általában ahhoz, hogy a segédprogramot futtatni lehessen, a karbantartott adatbázis-objektumokat az adatbázis-kezelő rendszerről le kellett kapcsolni. Ha rendszerparaméter változtatására volt szükség, akkor az egész adatbázis-kezelő rendszert le kellett állítani, és az újraindítással történt meg a változás. A legtöbb adatbázis-kezelő rendszer az újabb verziókban már olyan technikát biztosít, amellyel a rendszerparamétereket úgy lehet megváltoztatni, hogy közben az adatbázis-kezelő rendszert nem kell leállítani.
Ha az adatbázis-kezelő rendszernek új verziója kerül a piacra, akkor az adatbázis-adminisztrátornak érdemes felülvizsgálni, hogy a régi, leállást igényelő karbantartó műveletek helyett az új verzióba kerültek-e be olyan új megoldások, amelyek ugyanazt a feladatot már leállás nélkül is meg tudják oldani. Ha igen, akkor érdemes ezeket az új megoldásokat kihasználva újratervezni ezeket a karbantartási műveleteket.

Klasztertechnológia kihasználása

A klaszter összekapcsolt szerverek csoportja, amelynek a segítségével növelhetjük a szerverek megbízhatóságát. Emlékezzünk vissza a klaszterezés definíciójára, amely Az adatbázis-környezet létrehozása című fejezetben is szerepelt: A klaszterezés több független számítógép összekapcsolása, melyeknek az együtt végzett munkája a felhasználó számára úgy tűnik, mintha egy önálló, magas elérhetőségű rendszer használnánk. Tekintsük meg a 11-1-es ábrát.



11-1. ábra Klaszter

Többféleképp meg lehet valósítani a szerverek klaszterezését. Lehetőség lehet például, hogy a számítógépeknek csak a tároló eszközeiket osztjuk meg. Egy másik lehetőség, hogy egy szoftver segítségével a munkaterhelést szétosztjuk a számítógépek között. És természetesen még ezeken kívül is léteznek más megvalósítások a klaszterezésre.
A klaszterezés egyik nagy előnye, hogy ha a növekvő üzleti igények miatt az áteresztőképességet növelni szeretnénk, akkor új szervereket vagy csomópontokat kapcsolhatunk a klaszterhez mégpedig úgy, hogy közben az adatbázis-környezetünk elérhető marad.
A klaszterezésnek a másik nagy előnye a megbízhatóság. A klasztert meg lehet úgy valósítani, hogy ha az összekapcsolt szerverek közül az egyik meghibásodik, akkor annak a feladatát a többi szerver át tudja venni. Ezáltal csökken a leállás esélye, ami fokozza az elérhetőséget. A klaszterben minden szerver kapcsolatban van a többi szerverrel, így amikor a klaszternek egy csomóponttal megszűnik a kapcsolata, a klaszter felismeri ezt a hibát és elindít egy olyan folyamatot, ami a hiba kiküszöbölésére szolgál.
A szerverhiba kezelését tekintve a klasztereket többféleképp lehet megvalósítani. Például, az egyik megoldás, hogy ha egy szerver meghibásodik, akkor ennek a csomópontnak a feladataihoz kapcsolódó összes folyamatot egyetlen másik szerver veszi át. Egy másik lehetőség, hogy a klaszternek van egy olyan szervere, amelyik általában nem dolgozik. Amikor a hiba megtörténik, a tétlen csomópont veszi át a meghibásodott szerver feladatait. Harmadik megoldás lehet, hogy a meghibásodott szerver feladatát a klaszter a többi csomópont között szétosztja. Az, hogy melyik csomópont mennyi feladatot kap, azon múlik, hogy melyiknek mekkora az áteresztőképessége.
A klaszterezés növeli az elérhetőséget, mert a meghibásodott csomópontokat leállás nélkül el lehet távolítani a klaszterből és amikor a szerver újra működőképes lesz, újra csatlakoztathatjuk a klaszterhez.
A rengeteg előny ellenére sok IT cég nem használja ki a klaszter megoldást, aminek az oka

elsősorban a költségekben keresendő. A klaszterhez több gép szükséges, és alapvetően igaz, hogy egy gép még mindig olcsóbb, mint több gép, főleg kezdetben. Másik ok lehet, hogy az alkalmazásokat a klaszter megvalósításnak megfelelően valószínűleg módosítani kell, hogy a szerverhiba kezelése lehetővé váljon. Ez a változtatás is költségekkel jár.
A legtöbb operációs rendszer támogat bizonyos fokú klaszterezést. Az adatbázis szempontjából az adatbázis-kezelő rendszer szoftverét úgy kell beállítani, hogy ha tudja, akkor kihasználhassa az operációs rendszerbeli klaszter-támogatást.
A klaszterezés amiatt is hasznos, hogy a rutin karbantartások hatását elfedjük vele. Ha egy szervert karbantartás miatt a rendszerről le kell kapcsolni, az általa végzett munkát áttehetjük egy másik szerverre. Például, ha szükséges memóriát hozzáadni egy szerver alaplapjához, azt a szervert le kell állítani. Ha a leállított szerver egy klaszterben vesz részt, a munkaterhelését a klaszter egy másik csomópontjára lehet átirányítani. A memóriabővítés után pedig a szervert visszahelyezzük a klaszterbe, hogy újra ellássa ott a feladatát. Így a karbantartás nem igényel leállást.

Összefoglalás

A fejezetben meghatároztuk az elérhetőség fogalmát. Megnéztük, hogy milyen komponensekből tevődik össze: kezelhetőség, visszaállíthatóság, megbízhatóság, szervizelhetőség. Összehasonlítottuk az elérhetőséget és a teljesítményt. Megnéztük, hogy miért olyan fontos, hogy az adatbázisunk szinte folyamatosan elérhető legyen. Okokat kerestünk arra, hogy mi az ami hátráltatja az elérhetőséget: a karbantartás, és a döntéstámogatási rendszerek. Megnéztük, hogy milyen anyagi vonzata van annak, ha az adatbázis nem érhető el. Az elérhetőséget jellemezhetjük egy %-os értékkel, például 99,999%-os elérhetőség, megnéztük, hogy ez mit jelent. Felsoroltuk, hogy milyen problémák okozhatják, hogy az adatbázis nem érhető el. Illetve technikákat kerestünk arra, hogyan tudjuk a rendszerünknél a magasabb fokú elérhetőséget biztosítani: olyan segédprogramok használatával, amelyek működő rendszereken is el tudják látni a feladataikat; az adatbázis-adminisztrátori feladatok automatizálásával; azzal, hogy az adatbázis-kezelő rendszerünk által nyújtott elérhetőséget támogató eszközöket kihasználjuk, illetve a klaszter technológia kihasználásával.

Ellenőrző kérdések

1. Mit jelent az elérhetőség?
2. Milyen értékekkel jellemezhetjük az elérhetőséget?
3. Mi a különbség az elérhetőség és a teljesítmény között?
4. Az elérhetőségnek milyen tényezői vannak? Melyik mit jelent?
5. Milyen kapcsolata van egymással a karbantartásnak és az elérhetőségnek?
6. Milyen hatással vannak az adattárházak az operatív adatbázisok elérhetőségére?
7. Milyen költsége van a leállásnak?
8. Milyen problémák okozhatják azt, hogy az adatbázisunk nem érhető el?
9. Milyen lehetőségeink vannak az elérhetőség növelésére?
10. Milyen feladatokhoz van a legnagyobb szükség olyan segédprogramokra, amelyek nem szakítják meg az adatbázis működését?
11. A klaszterezést hogyan lehet megvalósítani a szerverhibák kezelésének szempontja alapján?

12. fejezet – Teljesítmény

A legtöbb cég monitorozza és hangolja az IT infrastruktúrájának a teljesítményét. Ez az infrastruktúra szervereket, hálózatokat, alkalmazásokat, asztali gépeket, és adatbázisokat tartalmaz. Az adatbázis-adminisztrátorok azonban gyakran túl elfoglaltak ahhoz, hogy odafigyeljenek a napi taktikai adatbázis-adminisztrációs feladatokra, így a rendszerüket nem proaktív módon figyelik és hangolják, annak ellenére, hogy valójában proaktív módon szeretnék. Így a teljesítmény javítására szolgáló lépéseket általában reaktív módon teszik meg, válaszul az olyan eseményekre, mint például egy felhasználónak válaszidőproblémája van, vagy egy táblatér kifut a lemezterületből stb.
A teljesítményproblémák kezelésének igazából egy cégszintű törekvésnek kellene lennie. A cég teljesítményproblémáinak a kezelése gyakran az adatbázis-adminisztrátori csoport feladata lesz, mert a legtöbb esetben az adatbázis-kezelő rendszert jelölik meg bűnösként mindaddig, amíg ártatlannak nem bizonyul. Minden teljesítményproblémáért az adatbázis-kezelő rendszert okolják, tekintet nélkül a valódi okra. Az adatbázis-adminisztrátornak ki kell tudnia deríteni, hogy mi a valódi oka a teljesítmény csökkenésének, hogy bebizonyítsa, hogy azt nem az adatbázis-kezelő rendszer okozza. Emiatt az adatbázis-adminisztrátornak ismernie kell a teljes IT infrastruktúrának legalább az alapjait, illetve szüksége van sok olyan segítőkész ismerősre, akik más területeken szakemberek (mint pl. operációs rendszer, hálózat, kommunikációs protokoll stb. szakértők). Ha az adatbázis-adminisztrátor ismeri az IT infrastruktúra nagy részét, akkor hatékonyan tud válaszolni a teljesítményproblémákra.
A teljesítményproblémák kezelését megkönnyíthetik a piacon fellelhető eseményvezérelt eszközök. Ezek az eszközök előre definiált műveleteket hívnak meg bizonyos jól definiálható események hatására. Például egy esemény az adatbázis proaktív újraszervezését indíthatja el, ha az eléri a tárolási kapacitását. Más eszközök pedig a teljesítményproblémák elemzését és kezelését könnyítik meg.

A teljesítmény definíciója

Az adatbázis-teljesítmény definícióját nagyon egyszerűen a következőképp adhatjuk meg: A felhasználó információt kér az adatbázistól. Az adatbázis-kezelő rendszer kielégíti a kérést. Azt a sebességet nevezhetjük adatbázis-teljesítménynek, amellyel az adatbázis-kezelő rendszer kielégíti az információ kérést.
Azonban ettől jobb, teljesebb definícióra van szükségünk az adatbázis-teljesítményre. Öt tényező befolyásolja az adatbázis-teljesítményt: *munkaterhelés, áteresztőképesség, erőforrás, optimalizáció, versengés*.
A *munkaterhelés* az online tranzakciók, a batch feladatok, az ad hoc lekérdezések, az adattárház elemzés és a rendszerparancsok kombinációja, amelyeket a rendszeren bármely időben végrehajtottak. A munkaterhelés drasztikusan ingadozhat napról napra, óráról órára, percről percre. Néha megjósolható, mint a hónap végi záráskor, más időben viszont megjósolhatatlan. A teljes munkaterhelésnek van az egyik legnagyobb hatása az adatbázis teljesítményére.
Az *áteresztőképesség* definiálja a számítógép teljes képességét az adatfeldolgozásra, vagyis az az adott mennyiségű munka, ami egy időegység alatt elvégezhető. Függ az input/output műveletek sebességétől, a processzor sebességétől, a számítógép párhuzamos képességeitől, illetve az operációs rendszer és a rendszerszoftver hatékonyságától.
A rendszer rendelkezésére álló hardver- és szoftvereszközöket a rendszer *erőforrás*aiként ismerjük. Erőforrásként tekintünk például a fizikai adatbázisra, a lemezes tárolási eszközökre, a véletlen elérésű memória chipekre, a gyorsító vezérlőkre, az alkalmazáskódokra, a processzorra, stb. Nagyon lényeges, hogy maga az adat is erőforrás, és igen magas üzleti értéke van.
Az adatbázis-teljesítmény negyedik eleme az *optimalizáció*. Az optimalizáció során az adatbázis-

adminisztrátor, a rendszer-adminisztrátor vagy az alkalmazásfejlesztő úgy módosítja az alkalmazást, az adatbázis-kezelő rendszert vagy valamely, az adatbázis-kezelő rendszerhez kapcsolódó szoftvert, hogy a rendszer ezáltal hatékonyabban működjön, vagy kevesebb erőforrást használjon. A relációs adatbázisokban az egyik legfontosabb cél az, hogy egy adatbáziskéréshez a leghatékonyabb elérési utat használjuk. Ehhez elsősorban lekérdezésoptimalizációra van szükség, amelyet az adatbázis-kezelő rendszer költségformulák alapján valósít meg. Azonban a leghatékonyabb elérési út megtalálásához sok más tényezőt is optimalizálni kell, mint az SQL utasításokat, az adatbázis-paramétereket, az alkalmazói programokat, vagy az adatbázis-kezelő rendszer konfigurációs paramétereit stb.
Ha egy erőforrásra nagy az igény, *versengés* léphet fel. A versengés olyan helyzet, amelyben a munkaterhelés két vagy több komponense egy erőforrást próbál használni konfliktusos módon, pl. két különböző tranzakció ugyanazt az adatot akarja frissíteni. Ahogy a versengés nő, az áteresztőképesség úgy csökken.
Az öt teljesítménybefolyásoló tényező alapján definiáljuk az adatbázis-teljesítményt. Az *adatbázis-teljesítmény* úgy definiálható, mint az erőforrások optimalizálása, hogy növeljük az áteresztőképességet és csökkentsük a versengést, lehetővé téve a lehető legnagyobb munkaterhelést.
Az adatbázis-teljesítményt nem lehet elkülönítve kezelni. Az alkalmazások általában az IT infrastruktúra más alrendszereivel és komponenseivel kommunikálnak. Ezeket is figyelembe kell venni a teljes teljesítmény tervezésénél. Nem biztos, hogy az adatbázis-adminisztrátorra kell hárítani a teljesítmény hangolásának a teljes felelősségét. Lehet külső szakértőket hívni, vagy a feladatot megosztani több adatbázis-adminisztrátor vagy más szakember között.

Az adatbázis-teljesítmény feltérképezése

Az adatbázisbeli teljesítményproblémák kezelésének tervezése minden alkalmazás megvalósításnál kritikus komponens. Az adatbázis-adminisztrátornak kell kitalálni egy tervet, amellyel megbizonyosodhat arról, hogy az adatbázisbeli teljesítményproblémák elemzését és kezelését végrehajtották a cég minden adatbázis-alkalmazásához. A teljes teljesítménykezelési terv tartalmazza az eszközöket is, amelyek segítik az alkalmazásteljesítmény monitorozását, illetve az adatbázis és az SQL utasítások hangolását.
Követve a 80/20 szabályt az első lépés a legproblémásabb területeket azonosítani. Ez nem olyan könnyű, mint amilyennek látszik.
A 80/20 szabály, vagy más néven Pareto-elv egy régi alapelv, amely azt állítja, hogy a 80%-a az eredménynek 20% erőfeszítésből származik. Ez a szabály általában a legtöbb erőfeszítés alkalmazásánál igaz. Egy kis erőfeszítés meghozza a legtöbb jutalmat. Az adatbázis-teljesítmény hangolásakor az adatbázis-adminisztrátornak először arra a teljesítményproblémára kell koncentrálnia, amely a legnagyobb valószínűséggel a problémát okozza, mert a hangolási erőfeszítése itt térül meg a legjobban.
Az a legvalószínűbb, hogy az adatbázis-alkalmazás teljesítményproblémájában a bűnös egy nem megfelelő SQL utasítás vagy alkalmazáskód. Tapasztalatok azt mutatják, hogy az adatbázis teljesítményproblémáinak a 75-80%-a visszavezethető a szegényesen kódolt SQL utasításokra vagy alkalmazás logikára. Ez nem jelenti azt, hogy az alkalmazásokban szükségszerűen rosszak az SQL utasítások. Lehet egy alkalmazás 100%-osan hangolva volt a gyors adatbázis-elérésre, amikor az először a termelési környezetbe került, de idővel a teljesítménye csökkenhetett. Az ilyen teljesítménycsökkenést sok dolog okozhatja, mint például az adatbázis növekedése, új felhasználók, az üzleti követelmények változása, stb.
A gyenge teljesítményű SQL utasításokat sok probléma okozhatja, például:

- teljes tábla átfuttatások, azaz a lekérdezésben szereplő táblát végig kell nézni
- megfelelő index hiánya
- az SQL utasítás nem használja a megfelelő indexet

- elavult adatbázis-statisztika
- a táblák összekapcsolása nem optimális sorrendben történik
- alkalmazások összekapcsolása a hatékonyabb SQL utasítások összekapcsolása helyett
- nem megfelelő összekapcsolási metódusok használata (nested loop, merge scan, stb.)
- hatékony SQL utasítás beágyazva egy nem hatékony alkalmazáskódba, például ciklusba
- nem hatékony allekérdezési formulák
- szükségtelen rendezés (GROUP BY, ORDER BY, UNION)

Nagyon nehéz feladat megtalálni a legköltségesebb SQL utasításokat. A másik nehézséget okozó feladat azoknak az SQL utasításoknak a felfedezése, amelyek több száz vagy több ezer programban vannak elrejtve. A teljesítményre ugyancsak nagy hatással lehetnek az interaktív felhasználók, akik ad hoc lekérdezéseket generálnak.

Érdemes SQL monitort használni, hogy azonosítani lehessen az SQL utasítások futását a rendszerben. Ez az eszköz SQL utasításokat használ, amely segítségével vissza tudja vezeti az utasítást arra, hogy ki és melyik programból adta azt ki és milyen erőforrásokat használt fel. Ha azonosítottuk a leginkább erőforrás-igényes utasításokat, akkor hangolhatjuk a legköltségesebb utasításokat.

Nem mindig nyilvánvaló, hogy a szegényesen kódolt SQL utasítást hogyan lehet hangolni. Az SQL utasítások megfelelő kódolása és hangolása egy bonyolult feladat. Nem feladatunk foglalkozni vele.

Más tényezők is befolyásolhatják negatívan az adatbázis-teljesítményt. Ellenőrizzük időnként az adatbázis-kezelő rendszernek és a szerver operációs rendszerének a teljes teljesítményét. Néhány részlet, amelyet az ellenőrizésnek tartalmaznia kell:

- memória allokáció (pufferek vagy gyorsítótárak az SQL utasításoknak, az adatoknak)
- naplózási opciók (napló gyorsítótár mérete, naplóállományok)
- input/output műveletek hatékonysága (táblák és indexek elkülönítése a lemezen, adatbázis-állományok mérete, töredezett és kiterjesztett állományok)
- a szerveren a teljes alkalmazás- és adatbázis-munkaterhelés
- adatbázisséma-definíciók

Monitorozás és kezelés

Sajnos az adatbázis-adminisztrátorok általában a teljesítményproblémákat reaktív módon kezelik. A probléma már fellépett, amikor a felhasználó válaszidőproblémával hívja az adatbázis-adminisztrátort vagy amikor egy adatbázis kifutott a helyből. A probléma bekövetkezte után kell orvosolni azt. Az ilyen tevékenység tisztán reaktív.

Sok proaktív lépést is reaktívként kell tekinteni. Egy kész alkalmazás kódjának az átírása nem tekinthető proaktívnak. Proaktív szemlélet az lenne, ha a problémát javították volna, mielőtt az alkalmazást befejezik, azaz a kódot hatékonyan írták volna meg.

Sok eseményvezérelt eszközt lehet használni a teljesítményhangolás egyszerűbbé tételére. Azaz amikor előredefiniált események történnek, akkor az eseményvezérelt eszköz automatikusan előredefiniált műveletek indít el. Ez az első lépés a teljesítménykezelés felé. A teljesítmény kezelése különbözik a teljesítmény monitorozásától, mert a kezelés kombinálja a monitorozást egy részletes tervvel, amely a fellépő problémát feloldja.

A teljesítménykezelés 3 különböző komponenst tartalmaz: *monitorozás*, *elemzés* és *javítás*.

A *monitorozás* a környezet vizsgálatát, az eszközök kimenetének áttekintését és a rendszer futásának általános figyelését tartalmazza. A monitorozás a problémák azonosításának folyamata.

Az *elemzés* üzenetek vagy riportok százait vagy ezreit generálhatja. Egy megfigyelő folyamat összegyűjti az odatartozó információkat a teljesítmény elemzéséhez és optimalizációs döntésekhez. Egy megfigyelő folyamat nem tud döntéseket hozni az összegyűjtött információ alapján. A döntéshozatalhoz elemzés szükséges, és azt általában egy tanult technikai szakember, az adatbázis-

adminisztrátor végzi.
Az *optimalizáció* javító művelet. Néhány teljesítménykezelő eszköz lehetővé teszi, hogy a szakember a teljesítménykezelés bizonyos részeit automatizálja. Azaz ezek az eszközök automatizálják a javító műveletek indulását, amikor az eseményfigyelő eszközök előre megadott feltételeket azonosítanak. Az adatbázis-adminisztrátornak úgy kell beállítani az automatikus indítást, hogy az biztosan jó időben induljon el. A teljesítménykezelő eszközök és megoldások egyre intelligensebbekké válnak. Olyan beépített tudással rendelkeznek, amelynek a segítségével az eszköz automatikusan optimalizálni tudja az adatbázis-kezelő rendszert. És olyan képességgel rendelkeznek, hogy az eszköz meg tudja tanulni, hogy mi működik legjobban a hangolási gyakorlatokból.
Teljesítménykezelést csak egy proaktív teljesítményterv használatával lehet végrehajtani. Még a probléma bekövetkezte előtt sok problémát lehet azonosítani és megoldást adni rájuk. Ha megfelelő tervvel rendelkezik az adatbázis-adminisztrátor, akkor a teljesítményproblémák javítása könnyebé válik, illetve néhány teljesítményprobléma elkerülhető.
A valódi teljesítménykezeléshez az adatbázis-adminisztrátornak egy alkalmazás teljesítményét már annak befejezése előtt terveznie kell. Ehhez az adatbázis-adminisztrátornak az alkalmazásfejlesztés teljes életciklusában részt kell vennie és biztosítania kell, hogy a teljesítmény az alkalmazásba bele legyen tervezve.

Reaktív és proaktív teljesítménykezelés

A teljesítményproblémák reaktív kezelésére mindig szükség van, mert előre nem látott teljesítményproblémák mindig jelentkeznek. Lehetetlen minden típusú teljesítményproblémát előrelátni; egy idő után minden rendszer és alkalmazás változik. A teljesítményproblémák reaktív kezelése nem rossz dolog, de kézi és időigényes folyamat. A proaktív teljesítménykezelés előregondolt, tervezett és automatizált, amely csökkenti a reaktív monitorozást és hangolást. Így a proaktív teljesítménykezelés csökkenti a ráfordított időt, erőfeszítést és az emberi hibát.

A teljesítmény becslése a termelési rendszerbe kerülés előtt

Az adatbázis-adminisztrátoroknak és a fejlesztőknek ideális esetben az alkalmazásokat a lehető legjobb teljesítményűre kell tervezniük. A teljesítményt az alkalmazásfejlesztés életciklusában korán, a tervezés és a létrehozás alatt figyelembe kell venniük. Ehhez a tervezéshez jó, ha valamilyen módszertant alkalmaznak. Az olyan szigorú folyamatok hatására, amelyek az értékelhető eredményre fókuszálnak, az alkalmazások és az adatbázisok jó teljesítménye egyszerűen megvalósulhat. Így elkerülhető a költséges újratervezés, legalábbis a teljesítmény problémák tekintetében.
Minél korábban történik a probléma azonosítása és javítása, annál kisebb a költsége. A proaktív teljesítménykezelés csökkentheti az alkalmazásfejlesztés költségét, mert a teljesítménykezelés azelőtt megtörténhet, mielőtt az alkalmazás a termelési környezetben működni kezd. Egy működő alkalmazásban a hibajavítás sokkal költségesebb. A termelési környezetben lévő teljesítményproblémák növelhetik azt az időt, amely az üzlet szempontjából kritikus munkák elvégzéséhez szükséges, mint például az ügyfelek kiszolgálása. Továbbá néhány teljesítményprobléma leállást okozhat. Itt jegyezzük meg, hogy a teljesítmény és a karbantarthatóság a rendszer két olyan tulajdonsága, amelyek egymás ellen hatnak.
Az alkalmazások teljesítményének becslése különbözik az egyedi adatbázis-lekérdezések elemzésétől és optimalizálásáról. A teljesítményt a teljes alkalmazásra kell értelmezni, mert az egyedi lekérdezések optimalizálása más lekérdezések kárára válhat. Ezért jó, ha az adatbázis-adminisztrátor egy modellt készít, amelynek a segítségével elemezni lehet, hogy az egyes lekérdezések optimalizálásának milyen hatása van az adott lekérdezés és más lekérdezések teljesítményére. Egy ilyen modell lehetővé teszi a teljes rendszer teljesítményének az

optimalizálását. A pontos teljesítménymodell létrehozása egy iteratív folyamat. Az egyes változásokat át kell vizsgálni és frissíteni kell, illetve meg kell mérni, hogyan hatnak a hatékonyságra.
Az adatbázis-adminisztrátoroknak, a rendszer-adminisztrátoroknak, az alkalmazásfejlesztőknek és a kapacitástervezőknek együtt kell dolgozniuk, meg kell osztaniuk minden olyan információt és üzleti rendszerkövetelményt, amely hatással lehet a teljesítménykritériumokra.

Történeti információk összegyűjtése

Az erőforrás használati információk és a teljesítménystatisztikák összegyűjtése és elemzése egy másik értékes teljesítményhez kapcsolódó feladat. A történeti teljesítmény és erőforrás használati információk lehetővé teszik az adatbázis-adminisztrátornak, hogy hetekkel, esetleg hónapokkal előre megjósolja a hardverfrissítések szükségességét. Az adminisztrátorok nyomon követhetik a kulcsfontosságú teljesítménystatisztikákat (input/output műveleteket, naplóváltásokat, pufferek használatát stb.) és a történeti információkat az adatbázisban nyomkövetési táblákban tárolhatják. Így értékes történeti információkhoz fér hozzá az adatbázis-adminisztrátor, amely információkról riportokat lehet készíteni, illetve elemezni lehet őket. Az adatbázis-adminisztrátorok nyomon követhetik a teljesítményt és az erőforrás-fogyasztást, és megjósolhatják, hogy mikor lesz a hardvererőforrás a megnövekedett adatbázis-használat miatt nagyobb mértékben kihasználva. A történeti információk megmutatják azokat a periódusokat, amikor a megnövekedett felhasználói aktivitás miatt az adatbázis-teljesítmény kisebb, mint az átlagos. A történeti teljesítményinformációk nagyon hasznosak az adatbázis-adminisztrátor számára, amikor megpróbálják megérteni az alkalmazás-, az adatbázis- és a rendszerteljesítmény jellemzőit.

Szolgáltatási szint kezelése

A szolgáltatási szint kezelése (service-level management, SLM) egy proaktív módszertan. Olyan eljárásokat tartalmaz, amelyek azt biztosítják, hogy az IT felhasználók a szolgáltatásokat olyan szinten kapják meg, amely megfelel az üzleti elvárásoknak és elfogadható az ára. A hatékony szolgáltatási szint kezelése érdekében egy cégnek az alkalmazásai között fontossági sorrendet kell felállítania és meg kell határoznia, hogy mennyi az az idő, erőfeszítés és tőke, amelyet arra az alkalmazásra lehet fordítani.
A szolgáltatási szint az alkalmazások működési viselkedésének egy mértéke. A szolgáltatási szint kezelésével biztosítható, hogy az alkalmazások a hozzájuk rendelt erőforrásokat olyan megadott mértékben használják ki, amelyet az alkalmazásnak a cégben betöltött szerepe alapján határoztak meg. A cég szükségleteit figyelembe véve a szolgáltatási szint kezelése az elérhetőségre, a teljesítményre vagy mindkettőre fókuszálhat. Az elérhetőség értelmében a szolgáltatási szint definiálható 99.95%-os működési időre hétköznapokon, reggel 9 és este 10 között. A szolgáltatási szint lehet pontosabb is, például egy tranzakciónak legyen az átlagos válaszideje 2 másodperc vagy kevesebb, 500 vagy kevesebb felhasználó munkaterhelése mellett.
Ahhoz, hogy egy szolgáltatási szintről szóló megállapodás (sevice-level agreement, SLA) valóban hasznos legyen, minden, a megállapodásban részt vevő félnek egyet kell értenie az elérhetőségre és a teljesítményre vonatkozó célokkal. A végfelhasználóknak meg kell elégedniük az alkalmazások teljesítményével, illetve az adatbázis-adminisztrátoroknak és a technikusoknak hajlandóknak kell lenniük a célnak megfelelően kezelni a rendszert. A kompromisszum létfontosságú ahhoz, hogy egy cégnél megvalósítható célok legyenek a szolgáltatási szintről szóló megállapodásba foglalva.
A gyakorlatban sok cég nem foglalkozik a szolgáltatási szint kezelésével. Ha új alkalmazást szállítanak sok határozatlan követelmény és ígéret lehet a másodperctöredék válaszidőre, de a prioritások és a költségvetés miatt az ilyen szolgáltatási szint ritkán valósul meg. Kivéve akkor, ha az IT funkciók ki vannak adva más cégnek (outsourcing). Gyakran a belső IT csoportok vonakodnak a szolgáltatási szintről szóló megállapodást aláírni, mert tudják, hogy bármely, a

szolgáltatási szintről szóló megállapodásban leírt cél elérését nehéz teljesíteni. Ha a cégek a szolgáltatási szintről szóló megállapodásban lévő célok megvalósításának a nehézségeit egyszer legyőzték, akkor a szolgáltatási szintről szóló megállapodásban szereplő célok teljesítését egy alacsonyabb költségű külső cégre bízhatja a belső IT csoport helyett.
Az, hogy a szolgáltatási szint kezelésének a feltételei nem teljesülnek, a legtöbb cégnél nem csak az IT csoporton, hanem az üzleti felhasználókon is múlik. Az üzleti felhasználók gyakran kérnek jobb szolgáltatást, de nem akarnak semmilyen erőfeszítést tenni érte, azaz nem tudják fontossági sorrendbe rendezni, vagy pontosan megfogalmazni a szükségleteiket, illetve nem akarnak többet fizetni az egyes szolgáltatásokért.
Másik lehetséges probléma a szolgáltatási szint kezelésével a szolgáltatás környezete. A legtöbb IT szakember a szolgáltatási szintet elemenként tekinti. Más szóval az adatbázis-adminisztrátor a teljesítményt az adatbázis-kezelő rendszer szintjén, a rendszer-adminisztrátor az operációs rendszer vagy a tranzakció feldolgozó rendszer szintjén, stb. tekintik. Az szolgáltatási szint kezelése az egész alkalmazás szolgáltatására vonatkozik. Nehéz a felelősséget egy tipikus IT infrastruktúrán belül megosztani. AZ IT csoport általában önálló emberek csoportjaként működik, akik együtt nem dolgoznak jól. Gyakran előfordul, hogy az alkalmazásfejlesztői csoport, amely a végfelhasználókkal kommunikálva igyekszik azok igényeit megfelelően kiszolgálni, függetlenül dolgozik az adatbázis-adminisztrátortól, aki függetlenül dolgozik a rendszer-adminisztrátortól. A szolgáltatási szint kezelésének megvalósításához az IT infrastruktúrán belül a különböző csoportoknak hatékonyan kell kommunikálniuk és kooperálniuk egymással. Ha ez nem megy, akkor a szolgáltatási szint kezelését nehéz vagy lehetetlen lesz megvalósítani.
A szolgáltatási szint kezelése nagyon hasznos egy cég életében, mert megjósolhatóvá teszi a teljesítmény kezelését. A szolgáltatási szintről szóló megállapodás nélkül az adatbázis-adminisztrátor és a végfelhasználók nem fogják tudni, hogy egy alkalmazás mikor működik megfelelően. Nem minden alkalmazásnak kell vagy lehet másodperc töredéke alatt válaszidőt szolgáltatnia. A szolgáltatási szintről szóló megállapodás nélkül az üzleti felhasználóknak és az adatbázis-adminisztrátoroknak különböző elvárásai lehetnek, amely elégedetlen üzleti vezetőket és frusztrált adatbázis-adminisztrátorokat eredményez.
A szolgáltatási szint kezelésével az adatbázis-adminisztrátorok úgy szabályozhatják az erőforrások használatát, hogy az megfeleljen a szolgáltatási szintről szóló megállapodásban definiált alkalmazások kritikusságának. Emellett a költségek kontrollálhatóak lesznek és a tőkét valóban arra az üzletrészre fordíthatják, amely a legfontosabb az cég számára.

A teljesítményhangolás típusai

Az adatbázis-alkalmazások a megfelelő működésükhöz állandóan kapcsolatban kell lenniük a különböző erőforrásokkal, és megkövetelik, hogy az erőforrások is állandó kapcsolatban legyenek egymással. Az alkalmazások hangolása emiatt összetett feladat, hiszen nem elég csak önmagában az alkalmazást hangolni, hanem hangolni kell az adatbázist és a rendszert is. Az adatbázis-alkalmazások hangolását három részre oszthatjuk: rendszerhangolás, adatbázis-hangolás és alkalmazáshangolás. Ezek a területek kapcsolatban vannak egymással és bizonyos közös hangolási elveket tekintve együtt is kezelendők.

Rendszerhangolás

A következők tartoznak a rendszerhangoláshoz: az adatbáziskezelőrendszer-szoftver telepítésének módja, a memória, a processzor, a lemez, más erőforrások és konfigurációs opciók hangolása; illetve az operációs rendszer, hálózati szoftverek, a tranzakció feldolgozók, köztes termékek hangolása. Telepítési, konfigurációs és integrációs témákat tartalmaz és a szoftverek kapcsolódását az adatbázis-kezelő rendszerhez és az adatbázis-alkalmazásokhoz. Bővebben a Rendszerteljesítmény című fejezetben olvashatunk róla.

Adatbázis-hangolás

Az adatbázis-hangolás a következőket foglalja magában: fizikai tervezés, normalizálás, táblák száma, indextervek, DDL és paraméterek használata, adattárolási kérdések, az adatbázis-állományok fizikai helye, új állományok, kiterjesztések, állománynövekedés. Bővebben az Adatbázis-teljesítmény című fejezetben olvashatunk róla.

Alkalmazáshangolás

Az alkalmazáshangolás tárgyalása nem fér bele ennek a jegyzetnek a kereteibe. Itt most csak felsoroljuk azokat a főbb témaköröket, amelyek ide tartoznak: adatbázis-statisztikák készítése, monitorozás, profilok kezelése, lekérdezések elemzése, elérési utak választása, költségek elemzése, optimalizálás, végrehajtási utak elemzése, SQL hangolása, hatékony kódok írása.

Teljesítményhangoló eszközök

Az adatbázis-teljesítmény hatékony kezelésében adatbázis-eszközök segítenek. Néhány adatbázis-kezelő rendszer beépített eszközöket biztosít az adatbázis-teljesítmény felügyeletére és hangolására. Ezeket az eszközöket kétféle kategóriába sorolhatjuk: kimondottan a teljesítmény felügyeletére és hangolására szolgáló eszközök, és azok az eszközök, amelyek az alapfunkciójuk mellett még a teljesítmény optimalizálását is szolgálják.
A teljesítmény felügyeletére és hangolására szolgáló eszközök:

- Teljesítménymonitorok: lehetővé teszik az adatbázis-adminisztrátornak és a teljesítményelemzőknek, hogy felmérjék az adatbázist elérő alkalmazások teljesítményét. Három mód van ezekre: valós idejű, közel valós idejű vagy történeti trendeken alapuló monitorozás.
- Hatékonyságbecslő eszközök: jósló teljesítménybecslést biztosítanak a programokhoz és az SQL utasításokhoz az elérési útra, a működési környezetre és egy szabály- vagy következtető motorra alapozva.
- Kapacitástervező eszköz: Segítségével az adatbázis-adminisztrátor elemezni tudja, hogy a jelenlegi környezetben a munkaterheléshez elegendő-e a jelenlegi áteresztőképesség az elérhető erőforrásokkal, illetve lehetővé teszi, hogy az adatbázis-adminisztrátor mi-lenne-ha típusú kérdésekre kapjon választ a kapacitással kapcsolatban.
- SQL elemző és hangoló eszköz: grafikus és/vagy karakteres leírása egy lekérdezés elérési útjának, amelyet a relációs optimalizáció határozott meg. Ez az eszköz SQL utasítások és programok esetén is használható.
- Advisory (tanácsadó) eszköz: bővíti az SQL elemző és hangoló eszközt tudásalap biztosításával, amely tippeket ad, hogyan lehet az SQL utasításokat optimális teljesítményűre újraírni.
- Rendszerelemző és rendszerhangoló eszközök: az adatbázis-adminisztrátor számára lehetővé teszi, hogy lássa és módosítsa az adatbázis- és rendszerparamétereket, mindezt grafikus felületen.

Ezeken kívül vannak olyan adatbáziseszközök, amelyek használatával az adatbázis-teljesítményét optimalizálni lehet:

- Újraszervező eszköz: az adatbázisok belső elrendezésük (töredezettség, sorok rendezettsége, tárolás) miatt teljesítményproblémával küzdhetnek. Az újraszervező eszköz automatikusan újraépíti az adatbázisokat úgy, hogy azok optimálisan legyenek szervezve.
- Tömörítő eszközök: csökkenthető a lemezen a tárolási igény, kevesebb input/output művelet szükséges. De növekedhet a processzor leterheltsége a betömörítés és kitömörítés eljárásai miatt.
- Rendező eszközök: mielőtt az adatokat betöltenénk az adatbázisba, ezzel az eszközzel

rendezhetjük őket. Így az adatbázisban a rendezés szerinti sorrendben lesznek tárolva. Ezáltal az adatok rendezett formában való lekérdezése sokkal gyorsabb lehet.

Általában létezik egy központi felület, amelyről ezek az eszközök elérhetőek.

Sok adatbázis-kezelő rendszer a teljesítményproblémák kezelésére különböző adatbáziseszközöket biztosít. Ha csak egy adatbázis-kezelő rendszert használunk, akkor érdemes az adatbázis-kezelő rendszer saját megoldásait használni. Heterogén környezetben több különböző adatbázis-kezelő rendszer és operációs rendszer esetén egy másik cég által biztosított eszköz jobb megoldás lehet.

Összefoglalás

A fejezetben áttekintettük a teljesítményhez kapcsolódó legalapvetőbb fogalmakat. Kétféleképp definiáltuk a teljesítményt. A bonyolultabb definícióban megadtuk a teljesítményt befolyásoló öt tényezőt és azok definícióit: munkaterhelés, áteresztőképesség, erőforrás, optimalizáció, versengés. Megállapítottuk, hogy teljesítményprobléma esetén mindig a legkritikusabb SQL kódokat érdemes megkeresni. Megvizsgáltuk, hogyan lehet ezeket a kritikus SQL utasításokat megtalálni, illetve azt, hogy mi lehet az oka annak, hogy ezeknek az SQL utasításoknak gyenge a teljesítménye. Megemlítettük azt is, hogy teljesítményproblémák esetén természetesen nem csak az SQL utasításokban kell keresni az okokat, hanem a problémát a rendszer más részei is okozhatják, mint az adatbázis-kezelő rendszer szoftvere vagy az operációs rendszer, illetve valamilyen más része az IT infrastruktúrának.

A monitorozás és kezelés alfejezetben különbséget tettünk a teljesítményproblémák reaktív és a proaktív kezelésében. Definiáltuk a teljesítménykezelést, és megnéztük a komponenseit. Az adatbázis-adminisztrátornak tudnia kell, hogy nem lehet minden teljesítményproblémát proaktív módon kezelni. A proaktív teljesítménykezeléssel kapcsolatban rávilágítottunk, hogy az alkalmazásoknak a teljesítményével már az alkalmazás tervezésekor foglalkozni kell, és végig kell kísérni a teljes életciklus alatt. Az alkalmazások teljesítményének a becsléséhez az adatbázis-adminisztrátor készíthet egy modellt, amellyel elemezni tudja, hogy az egyes lekérdezések optimalizálása milyen hatással van a többi lekérdezés teljesítményére. Az adatbázis-adminisztrátor a teljesítmény elemzésére használati információkat és teljesítménystatisztikákat gyűjthet össze a működő rendszerről. Ezeknek az információknak a segítségével előre meg lehet jósolni bizonyos események bekövetkeztét. Így teljesítmény vagy leállási problémákat lehet elkerülni.

A szolgáltatási szint kezelésében az üzleti felhasználóknak és az adminisztrátoroknak kompromisszumra kell jutniuk a rendszer teljesítményével kapcsolatban. Ezt a kompromisszumot jó, ha valamilyen mérhető értékekhez kötik, mint például a válaszidő hosszúsága vagy az elérhetőség százalékos meghatározása. Ezt a kompromisszumot a szolgáltatási szintről szóló megállapodásban rögzítik. Sok cégnél ez a megállapodás különböző okok miatt nem jön létre, melyeket a fejezetben részleteztünk.

Megnéztük, hogy a teljesítményhangolásnak milyen típusai vannak. Majd a fejezet végén áttekintettük a teljesítményhangoló eszközöket.

Ellenőrző kérdések

1. Mi a teljesítmény definíciója?
2. Mit jelentenek a következő fogalmak: munkaterhelés, áteresztőképesség, erőforrás, optimalizáció, versengés?
3. Kinek a feladata egy cégnél a teljesítményproblémák megoldása?
4. Mit jelent a Pareto-elv vagy 80/20-as szabály?
5. Milyen problémák okozhatnak gyenge teljesítményű SQL utasításokat?
6. Milyen eszköz segítségével lehet megtalálni a gyenge teljesítményű SQL utasításokat?
7. Miket érdemes ellenőrizni a teljesítmény javítása érdekében az adatbázis-kezelő rendszer és az operációs rendszerhez kapcsolódóan?

8. Mit jelent a teljesítménykezelés? Milyen komponensei vannak?
9. Mi a különbség a teljesítményproblémák proaktív és reaktív kezelésében? Lehet minden teljesítményproblémát proaktívan kezelni?
10. Mikor kell az alkalmazáskódok teljesítményével először foglalkozni? Miért?
11. Miért érdemes az adatbázis-adminisztrátornak a lekérdezések teljesítményéhez modellt készítenie?
12. Mire szolgálnak a történeti információk?
13. Mit jelent a szolgáltatási szint kezelése?
14. Mit jelent a szolgáltatási szintről szóló megállapodás?
15. Sok cégnél miért nem tudnak megállapodást kötni a szolgáltatási szintről?
16. Miért hasznos a szolgáltatási szintről szóló megállapodás?
17. Milyen teljesítményhangoló eszközöket ismerünk?

13. fejezet – Rendszerteljesítmény

A szegényes teljesítményű rendszeren futó adatbázisok és alkalmazások teljesítménye is gyenge lesz. Semmilyen adatbázis-, alkalmazás- vagy SQL hangolás nem javít a teljesítményen, ha a gyengén megvalósított rendszer okozza a problémát. A rendszer az adatbázisok és alkalmazások teljesítményét is érinti, úgy ahogy az adatbázis-teljesítmény az őt használó alkalmazások teljesítményét érinti.

A nagyobb környezet

Az adatbázis-kezelő rendszer egy nagyobb, hardver és szoftverelemekből álló környezetben működik. Ezeket az elemeket úgy kell telepíteni, konfigurálni és hatékonyan kezelni, hogy az adatbázis-kezelő rendszer számára megfelelő módon működjenek. Az adatbázis-adminisztrátornak tudnia kell, hogyan hat egymásra az adatbázis-kezelő rendszer, a szerver hardvere, az operációs rendszer, és más szükséges szoftverek. Ezeknek az elemeknek és kapcsolatoknak a hangolása és konfigurálása drámai hatással lehet a rendszer teljesítményére.

Hardverkonfiguráció

A következő kérdések megválaszolása segíti az adatbázis-adminisztrátort abban, hogy biztosítani tudja az adatbázis-alkalmazásokhoz szükséges optimális hardverkörnyezetet:

- Megfelelőek a hardver képességei az adatbázis-kezelő rendszer környezet számára? Más szóval az adatbázis-kezelő rendszer szállítója támogatja ezt a hardvert?
- Naprakész a számítógép firmware (ROM BIOS)?
- Van megfelelő mennyiségű memória a gépben, amely elég az összes telepített szoftverhez?
- Megfelelő mennyiségű lemezterület lett lefoglalva és konfigurálva az adatbázis-kezelő rendszerhez?
- Milyen típusú a tárolási eszköz? Megfelelő a nagyméretű adatokhoz és a gyors adatbázis-lekérdezésekhez?
- Be vannak kötve és megfelelően működnek a hálózati kábelek?
- Teljesen kapcsolódtak és működnek a fizikai kapcsolatok?
- A hardver kapcsolódik szünetmentes tápegységhez?

Lemezes háttértár és az input/output műveletek

Az adatbázis-teljesítmény egyik legszűkebb keresztmetszete az input/output műveletek végrehajtásának fizikai költsége. Az adat a lemezen van, a lemez egy mechanikai eszköz. A géprészeknek mozogniuk kell, hogy a kódolt adatot a forgó lemezről kiolvassuk. Ez a fizikai művelet időt vesz igénybe, ezért minden, ami az input/output művelet idejét csökkenti, növelheti a teljesítményt.
A lemez elérésének optimalizálására használjunk SSD-t (solid state device). Az SSD egy olyan adattároló eszköz, ami félvezetős memóriában őrzi a tárolt adatot, azt hosszú ideig megőrzi, azaz állandó tár. Az SSD-t úgy konfigurálják, mint egy lemezegységet, mint egy merevlemezt. Az adatok olvasása az SSD-ről sokkal gyorsabb, mint a hagyományos merevlemezekről, mert az SSD mozgó alkatrészeket nem tartalmaz. A mozgó alkatrészek hiánya miatt kevésbé sérülékeny, mint a hagyományos merevlemez, csendesebb, nincsenek a mechanikából adódó késleltetések, az adathozzáférés egyenletesen gyors. Nevezik flash memóriának is és ram-drive-nak is.
Ha magas teljesítményszintet követel meg a cégünk, érdemes megfontolni azt, hogy az adatbázis-objektumokat SSD-re helyezzük, fizikai lemezmeghajtó, RAID vagy SAN használata helyett.

Az SSD-nek azonban van egy nagy problémája: a tartóssága. Az SSD eszközök érzékenyek a hirtelen áramkimaradásra, illetve csak korlátozott számban lehet újraírni őket. Emiatt SSD használata esetén különösen fontos az, hogy a megfelelő mentési és helyreállítási tervről gondoskodjunk.

Kapcsolat az operációs rendszerrel

Ha az operációs rendszerben teljesítményproblémát tapasztalunk, minden szoftvernek, amely azon az operációs rendszeren fut, teljesítmény problémája lesz. Az adatbázis-adminisztrátort a következő kérdések megválaszolása segíti abban, hogy az adatbázis-alkalmazásokhoz szükséges optimális operációsrendszer-megvalósítást biztosítsa:

- Elegendő mennyiségű memória lett allokálva az operációs rendszer feladatokhoz?
- Van elegendő mennyiségű lemezterület allokálva a swap területnek? A legtöbb operációs rendszer képes bizonyos mennyiségű lemezterületet swap területként allokálni. A swap területet akkor használják, ha az operációs rendszer kifut a memóriából.
- Hogyan lettek az adatbázis-állományok allokálva az adatbázis megvalósításakor? Az állományrendszerrel való munka néhány operációs rendszer esetén többletmunkát jelent a rendszernek. Ha raw lemezt használunk, az operációs rendszer vagy állományrendszer miatt adódó többletterhelést elkerülhetjük.
- Néhány operációs rendszer lehetővé teszi, hogy az operációs rendszer alatt futó feladatok között prioritást határozzon meg. Az adatbázishoz kapcsolódó feladatokhoz van rendelve prioritás? Megfelel a prioritás a különböző feladatokhoz?
- Az adatbázis-kezelő rendszer a szolgáltató által kért operációs rendszer verzión van-e telepítve? Van-e olyan hibajavítás az operációs rendszerhez, amely alkalmazható?
- Az operációs rendszer konfigurációs paraméterek az adatbázis-kezelő rendszer telepítésekor módosultak-e? Ha igen, történt-e megfelelő tesztelés, amely biztosítja, hogy a paraméterek megfelelően lettek módosítva és nincsenek hatással más folyamatokra, amelyek az adatbázis-szerveren futnak?

Más szoftvererek

Az adatbázis-kezelő rendszer kapcsolatban áll más szoftverkomponensekkel is. Így lehet megvalósítani a szolgáltatás szállítását a végfelhasználóhoz. Ilyen szoftverek például:

- tranzakció feldolgozók
- hálózati szoftverek
- üzenetsorba-állító szoftverek
- web kapcsolódási és -fejlesztő szoftverek
- programozási nyelvek

Ahhoz, hogy ezek a szoftverek megfelelően kapcsolódjanak az adatbázis-kezelő rendszerhez, konfigurálni kell őket. Általában az adatbázis-adminisztrátor feladata, hogy a konfigurációt elvégezze. Nagyobb cégnél nem biztos, hogy az adatbázis-adminisztrátor végzi a konfigurálást, hanem más képzett szakemberek, akik a szoftverek kezelésére és adminisztrálására specializálódtak.

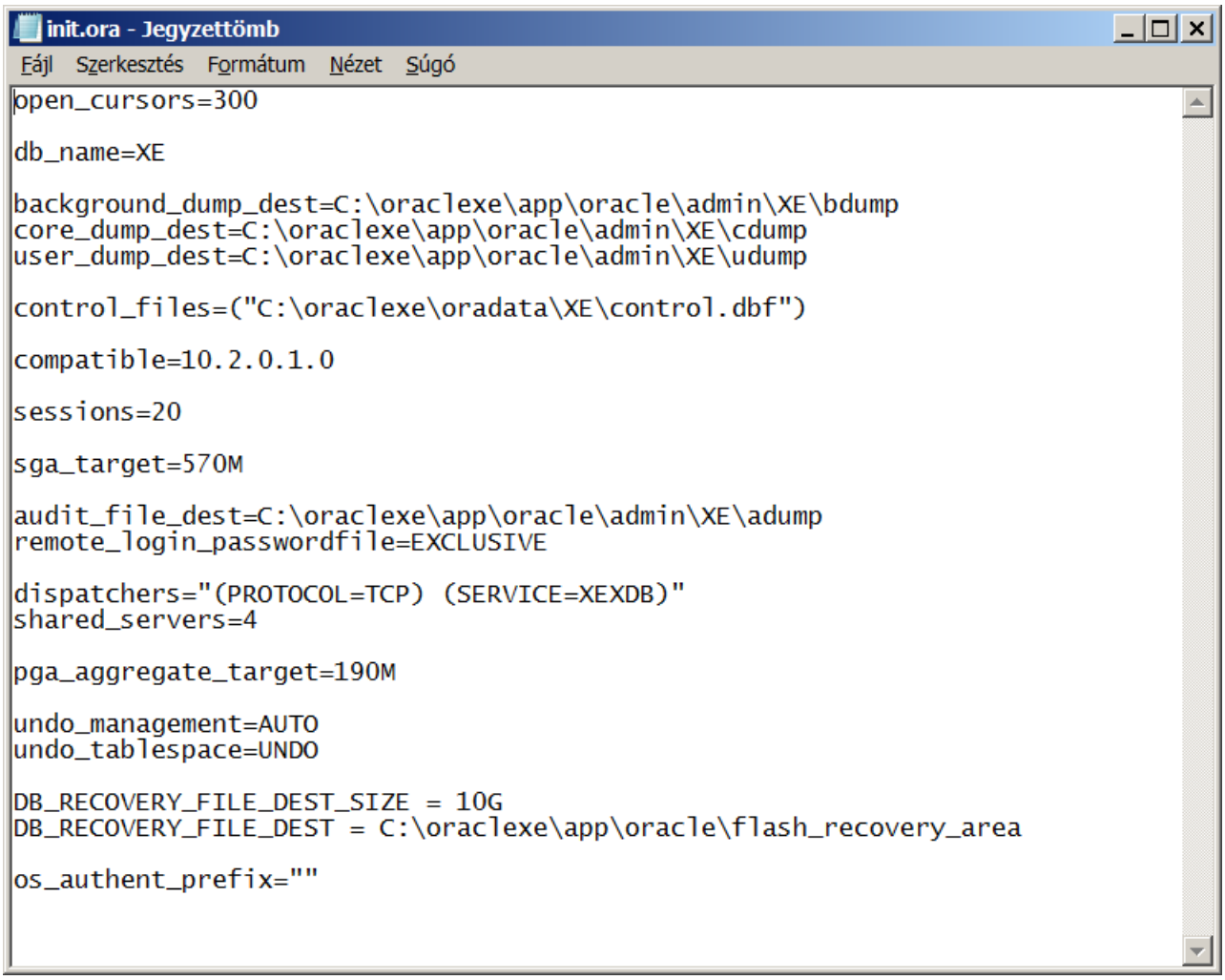
Az adatbázis-kezelő rendszer komponensei

Egy adatbázis-kezelő rendszer nagyon összetett rendszer, több százezer sor kód szükséges hozzá.
Az adatbázis-kezelő rendszerek szállítói az adatbázis-kezelő rendszerek funkcionalitását különböző részekre osztják. Az adatbázis-adminisztrátornak meg kell ismernie az adatbázis-kezelő rendszer alkotóelemeit és azt, hogy ezek a részek hogyan kapcsolódnak az adatbázis-kezelő rendszer más részeivel.
Az adatbázis-adminisztrátor csak akkor tudja az adatbázis-alkalmazásoknak a megfelelően

optimalizált környezetét biztosítani, ha ismeri az adatbázis-kezelő rendszer belső működését. Bármilyen teljesítményprobléma esetén, az adatbázis-adminisztrátornak tudnia kell, hogy az adatbázis-kezelő rendszer mely része okozhatta a teljesítménycsökkenést.

Az adatbázis-kezelő rendszer telepítése és konfigurálása

Minden adatbázis-kezelő rendszer rendelkezik olyan paraméterekkel, amelyeknek a módosításával az adatbázis-adminisztrátor az adatbázis-környezetet konfigurálni tudja. Adatbázis-kezelő rendszertől függően különböző módon lehet a paramétereket változtatni. Lehet, hogy egy olyan állományt kell módosítani, amelyben a paraméterek nevéhez értékek vannak rendelve, vagy lehet, hogy adatbázis-kezelő rendszer utasításokat kell kiadni, de az is lehet, hogy rendszereljárást kell futtatni a paraméterek módosításához.
Az adatbáziskezelőrendszer-szoftvereket a szállító általában alapértelmezett értékekkel adja, de az alapértelmezett értékek legtöbbször nem elegendőek egy robusztus termelési alkalmazás fejlesztésének támogatásához.
A 13-1-es ábrán az Oracle adatbázis-kezelő rendszernek az inicializációs paraméterállományát láthatjuk. Az Oracle-nek ez az egyik eszköze, hogy a paramétereket módosítsa.



13-1. ábra Paraméterállomány

A konfigurációk típusai

Az adatbázis-kezelő rendszert már a telepítéskor is lehet konfigurálni, de később is, amikor már

átadták működtetésre. Az adatbázis-kezelő rendszertől függően vannak olyan paraméterek, amelyeket dinamikusan lehet változtatni, azaz az adatbázis-kezelő rendszer működtetése közben változnak, és hatásuk azonnal érvénybe lép, vannak olyan paraméterek, amelyeknek a módosítása az adatbázis-kezelő rendszer újraindítását igénylik, és vannak olyan paraméterek, amelyeket a telepítés után nem lehet változtatni.
Ha az adatbázis-környezetünket konfiguráljuk nem érdemes az alapértelmezett értékekkel dolgozni. A legtöbb konfigurációs opció egy előredefiniált értéket kap, ha nem adunk meg értéket. A legnagyobb probléma az alapértelmezett értékekkel, hogy majdnem soha nem lesz a legjobb választás egy adott környezethez. Azonban kellő tapasztalat híjján érdemes alkalmazni őket, mert biztos, hogy nem a lehető legrosszabb értékeket állítjuk be. Mindenképpen jobbak, mintha az adatbázis-adminisztrátor próbálkozásszerűen állítaná be az értékeket.
Legyünk óvatosak azokkal a konfigurációs opciókkal, amelyek megváltoztatják az adatbázis-kezelő rendszer viselkedését. Ha rossz értékre állítjuk a konfigurációs paramétert, az sok bajt okozhat.

Memóriahasználat

A relációs adatbázisok imádják a memóriát. Minél több memóriát biztosítunk az adatbázis-kezelő rendszernek, annál jobb lesz a teljesítménye.
Az adatbázis-kezelő rendszer gyorsítótárakat használ arra, hogy az erőforrásokat minél tovább a memóriában tartsa. Minél tovább marad a memóriában az erőforrás, annál nagyobb az esély arra, hogy a következő erőforráskéréseknek nem lesz szükségük az input/output műveletekre.
Az adatbázis-kezelő rendszer többféle gyorsítót és puffert használ. Különböző adatbázis-kezelő rendszerek puffereinek a célja általában ugyanaz, csak más terminológiával.
Az *adatpuffer* vagy *adatgyorsító* az input/output műveletek minimalizálását szolgálják. Ha az aktuális adat az adatbázisból felolvasásra került, akkor újabb kérésnél nem kell még egyszer a lemezről felolvasni. A memóriában lévő adat elérése lényegesen gyorsabb, mint a lemezen lévő adat elérése. Ezért egy relációs adatbázis minden adatának elérése a gyorsítóterületen keresztül megy. Ha egy programnak egy táblából egy sorra van szüksége, az adatbázis-kezelő rendszer a lemezről az adatlapot kinyeri és az adatgyorsítóban tárolja. Alapvetően az adatbázis-kezelő rendszer a gyorsítót átmeneti területként használja. Ha a sor változik, akkor a változás az adatgyorsítóbeli adatlapra kerül. Az adatbázis-kezelő rendszer fogja az adatgyorsítóban található adatlapot visszaírni a lemezre. Ha egy alkalmazásnak szüksége van az adatra, és az olyan adatlapon van, amely már az adatgyorsítóban van, akkor az alkalmazásnak nem kell arra várnia, hogy az adatlapot az adatbázis-kezelő rendszer a lemezről kinyerje, hanem egyszerűen az adatpufferben található adatlapon található adatot használja fel.
Az *eljárásgyorsító* tárolja az SQL és a programhoz kapcsolódó struktúrákat. Az adatmódosító és lekérdező SQL utasítások előtt az adatbázis-kezelő rendszer az utasítást először optimalizálja. Az optimalizáló folyamat egy elérési utat reprezentáló belső struktúrát hoz létre, amelyet az adatbázis-kezelő rendszer az adat olvasásához használ. Az adatbázis-kezelő rendszer ezeket az elérési utakat az eljárásgyorsítóban tárolja, és újrahasználja őket, amikor ugyanaz a program vagy SQL utasítás újra lefut. Ez az alkalmazásteljesítményt optimalizálja, mert az optimalizáló folyamatnak nem kell mindig lefutnia, amikor ugyanaz az SQL utasítás fut. Az optimalizáció az SQL utasítás első futásánál végrehajtódik és a következő futtatások az elérési utat az eljárásgyorsítóból nyerik ki.
A *rendezési gyorsítót* az ideiglenes lemezterületek helyett használja az adatbázis-kezelő rendszer. Ebben a memóriabeli gyorsítóban tárolja az adatbázis-kezelő rendszer a közbenső rendezési eredményeket, melyekre olyan relációs adatbázis-műveletnek van szüksége, amelyek rendezést igényelnek, mint GROUP BY, ORDER BY, UNION, és bizonyos típusú összekapcsolások.
Az adatbázis-kezelő rendszer *belső struktúragyorsítókat* is használhat. A relációs műveletek megvalósításához az adatbázis-kezelő rendszer belső struktúrákat hozhat létre, amelyek a végfelhasználó számára nem feltétlenül láthatók. Az adatbázis-adminisztrátoroknak és néha a programozóknak tudniuk kell ezekről a belső struktúrákról.

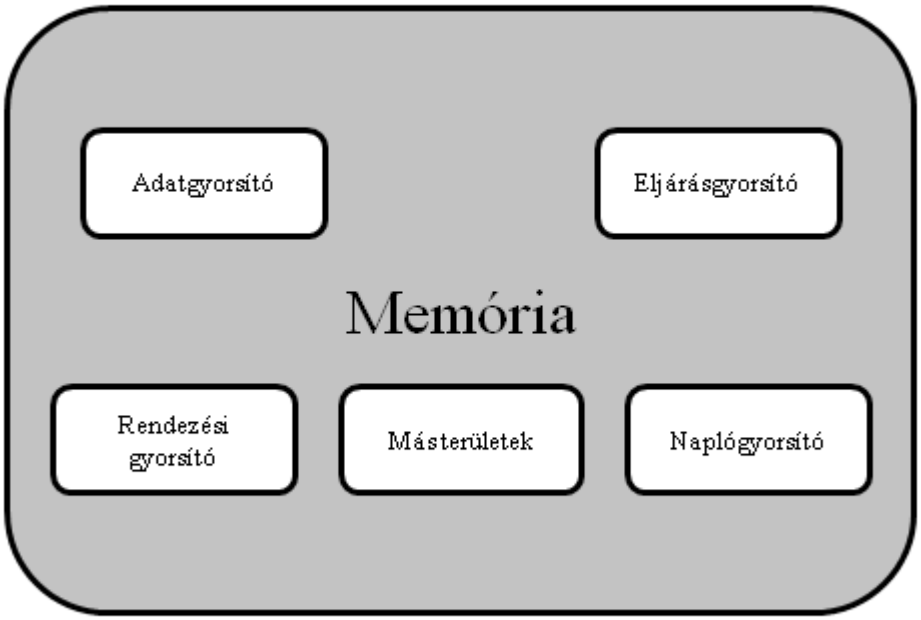
Az adatbázis-kezelő rendszer az adatbázisnapló-rekordok memóriában való tárolására a *naplógyorsítót* különíti el. Az adatbázis-kezelő rendszer kétféle naplógyorsítót valósíthat meg, egyet az írási és egyet az olvasási naplózáshoz. Az adatbázisnaplóban szerepel az adatbázison végrehajtott összes változás rekordja. Az írási naplógyorsítóba az adatbázis-kezelő rendszer az adatbázis módosításaihoz készült naplórekordokat írja. Az adatbázisnapló rekordjai a naplógyorsítóból idővel aszinkron módon a lemezen található aktív adatbázisnapló-állományba kerülnek. A naplóírás ezen módjával az adatbázisnapló nem lesz a rendszer- és az alkalmazásteljesítmény szűk keresztmetszete. Az olvasási naplógyorsítót a ROLLBACK és RECOVER műveletek használják. Egy visszagörgetésnek vagy egy helyreállításnak az adatbázisnapló elérésére van szüksége, hogy az adatbázis-változásokat visszagörgesse vagy újra végrehajtsa. Ha az adatbázisnapló-rekordokra szükség van, akkor azok a memória olvasási naplógyorsítójába kerülnek.

A memória további területei

A következők miatt szükség lehet arra, hogy az adatbázis-kezelő rendszer különböző területeket foglaljon le:

- Felhasználói kapcsolatok: A konkurens felhasználók adatbázis-kezelő rendszerhez való kapcsolódásának az adatbázis-kezelő rendszer memóriájára van szüksége a kapcsolat fenntartásához és kezeléséhez. Mindez független a kapcsolódás típusától.
- Eszközök: Az adatbázis által használt eszközöknek az adatbázis-kezelő rendszer által lefoglalt memóriaterület fenntartására és használatára lehet szükségük.
- Megnyitott adatbázisok: a legtöbb adatbázis-kezelő rendszer egy paramétert biztosít arra, hogy meghatározza azoknak az adatbázisoknak a maximális számát, amelyek egy időben lehetnek nyitva. Minden megnyitott adatbázisnak adatbázis-kezelő rendszer memóriára van szüksége.
- Megnyitott objektumok: az adatbázis-kezelő rendszer egy paramétert biztosíthat azoknak az objektumoknak a maximális számára, amelyek egy időben lehetnek megnyitva, beleértve a táblákat, indexeket, és más adatbázis-objektumokat. Minden megnyitott adatbázis-objektumnak memóriára van szüksége.
- Zárak: minden konkurensen fenntartott zárnak memóriára van szüksége. Az adatbázis-kezelő rendszernek kell biztosítania egy olyan paramétert, amely megadja az egy időben fenntartható a konkurens zárak számát.

A 13-2-es ábrán az adatbázis-kezelő rendszerek memóriaterületeit figyelhetjük meg.



13-2. ábra Memóriaterületek

Mennyi memória elég?

Ez egy nehéz kérdés. Az általános válasz: annyi, hogy a munka el legyen végezve, de ez nem segít az adatbázis-adminisztrátornak.
Minden adatbázis-kezelő rendszer használ memóriát, de különböző dolgokra különböző mennyiségűt. Az adatbázis-kezelő rendszer dokumentációjában keressük meg, hogy az egyes erőforrásoknak mennyi memóriára van szükségük.
A szükségestől egy kicsivel több helyet hagyjunk a nem várt alkalmazói programok, az adatbázis-kezelő rendszer és operációs rendszer frissítések miatt.

Az adatgyorsító részei

Egy adott időpontban az adatgyorsító háromféle adatlapot fog tartalmazni:

- Tiszta adatlapok: azok az adatlapok, amelyeket az adatbázis-kezelő rendszer korábban olvasásra használt és olvasáskonzisztens maradt. Ezeknek az adatlapoknak a tartalmát más adatbázis-folyamatok újra felhasználhatják.
- Piszkos adatlapok: azok az adatlapok, amelyek valamilyen módon módosítva lettek, de még nem kerültek a lemezre kiírásra. Ezek az adatlapok más adatbázis-folyamatok számára nem érhetőek el.
- Nem használt adatlapok: Ezeket az adatlapokat az adatbázis-kezelő rendszer jelenleg nem használja. Elérhetőek adatbázis-folyamatok számára. Az adatgyorsító nem használt adatlapjaira új adatok kerülhetnek.

Az adatlapok kezelésére az adatbázis-kezelő rendszerek saját belső algoritmust alkalmaznak.

Az adatgyorsító teljesítménye függ attól, hogy a memória mennyire hatékonyan lett allokálva. Ha nem az elérhető adatlapok dominálnak az adatgyorsítóban, az adatbázis-kezelő rendszer szinkron írásokat végezhet, hogy növelje a helyet az adatgyorsítóban. A szinkron írás lelassítja az adatbázis-feldolgozást, mert az írási kérésnek várnia kell, hogy az adat fizikailag a lemezre legyen írva. A használatban lévő adatbázis-alkalmazás feldolgozási típusától függően az adatbázis-adminisztrátor hangolhatja az adatgyorsító paramétereit és méretét, hogy hatékonyabb adatpufferezést érjen el.

Az adatgyorsító monitorozása és hangolása

A hatékony adatgyorsító biztosításához a legkritikusabb kérdés az, hogy mekkora a megfelelő méretű adatgyorsító. Ha túl nagy, akkor pocsékolja a memóriát, és az adatlapok a háttértárolón lévő segédtárba, a swap területre futhatnak. Ha túl kicsi, akkor gyakran kell a lemezre írni és az adatgyorsítóbeli adatlapokat a lemezről és a lemezre gyakran kell mozgatni.
Az adatgyorsító hangolásának összetettsége az adatbázis-kezelő rendszertől függ. Minden adatbázis-kezelő rendszer esetén igaz, hogy az adatbázis-adminisztrátornak figyelnie kell az adatgyorsító olvasási hatékonyságát.
Az adatgyorsító olvasási hatékonysága egy olyan százalékos érték, amely azt mutatja meg, hogy milyen jól teljesíti a gyorsító az elsődleges feladatát, azaz hogy elkerülje a fizikai input/output műveleteket. Az olvasási hatékonyságot a következőképp lehet kiszámolni:
Olvasási hatékonyság=((adatbázis I/O kérések)-(fizikai I/O-k)) / (adatbázis I/O kérések)
Más szóval az olvasási hatékonyság megmutatja, hogy mekkora arányban találhatóak meg az adatlapok az adatgyorsítóban. Minél nagyobb az értéke, annál hatékonyabb a gyorsító. Ha az adatlap megtalálható a pufferben fizikai input/output kérés nélkül, akkor a teljesítmény nőni fog.
Az aktuális input/output kérések és a fizikai input/output műveletek az adatbázis-kezelő rendszer nyomkövető állományaiban megtalálhatóak, vagy az adatbázisteljesítmény-monitor segítségével megtalálhatóak. Az adatbázis-kezelő rendszertől függően az adatbázis-adminisztrátornak be kell kapcsolnia a nyomkövetést ahhoz, hogy lekérdezhesse az adatbázis-kezelő rendszer ezen információit.
Egy jó adatgyorsító olvasási hatékonysága legalább 80%-os. Természetesen az olvasási hatékonyság értéke a feldolgozás típusától is függ. Sok szekvenciális feldolgozás esetén előfordulhat, hogy az adat túlcsordul az adatgyorsítón, akkor a hatékonyság csökken. Az olyan adatbázis-kezelő rendszerek esetén, ahol az egyes folyamatok az adatokat csak hetente vagy havonta használják, elegendő kevesebb olvasási hatékonyság is, mert kevesebb olyan adat lesz, amelyet valamely folyamat újra fel tud használni. Az adatbázis-adminisztrátornak tudnia kell azt, hogy az adott adatbázis-kezelő rendszerben milyen típusú adatbázis-feldolgozások futnak, tudnia kell, hogy az ilyen típusú adatbázis-feldolgozásokhoz mekkora olvasási hatékonyság elegendő, és ennek megfelelően kell a gyorsító méretét beállítania.
Ha az olvasási hatékonyság lényegesen 80% alatt van, tekintetbe vehetjük az adatgyorsító méretének növelését vagy az adatgyorsítóhoz rendelt táblák és indexek számának csökkentését. Az ilyen változás hatással van az elérhetőségre, mert az adatbázis-kezelő rendszernek le kell állnia és újra kell indulnia, hogy a változást regisztrálja.

Az eljárásgyorsító monitorozása és hangolása

Az adatbázis-adminisztrátornak figyelnie kell az eljárásgyorsító hatékonyságát, hogy segítse az adatbázis-alkalmazások és lekérdezések hatékonyságának növelését. Az eljárásgyorsító adatbázis-kezelő rendszerről adatbázis-kezelő rendszerre változik, de az általános ötlet azonos: az optimalizált SQL utasításszerkezetet a memóriában tartani, hogy újra fel lehessen használni a következő feladatnál, ahelyett hogy újrafordítani és újraolvasni kellene.
Az optimális teljesítmény biztosításához az eljárásgyorsítót megfelelően kell méretezni, hogy minden olyan SQL utasításhoz alkalmazkodjon, amely párhuzamosan futhat. Az eljárásgyorsítónak

is van olvasási hatékonysága. Azt mutatja meg, hogy az adatbázis-kezelő rendszernek milyen gyakran kell az újrafelhasznált SQL utasításokat újraoptimalizálni. Az eljárásgyorsító olvasási hatékonysága 60 és 80 % közé esik. Ez az érték függ az adatbázis-kezelő rendszertől, az alkalmazások típusától, és attól hogy ugyanaz a program vagy SQL hányszor fut.

Adatbázisnapló

Az adatbázisnapló vagy tranzakciónapló konfigurációja is hatással van a teljesítményre. Az adatbázisnapló az adatbázis-kezelő rendszerek alapvető komponense. Az adatbázisban az alkalmazásadatoknak minden módosítása az adatbázisnaplóban mentve van. Ezt az információt használva az adatbázis-kezelő rendszer nyomon követheti, hogy melyik tranzakció melyik módosítást végezte az adatbázison. A ROLLBACK és a RECOVER műveletek az adatbázisnaplót használják, hogy az adatbázist egy bizonyos pontig helyreállítsák.
A normál adatbázisalkalmazás-feldolgozásnál az SQL INSERT-ek, UPDATE-k, DELETE-k módosítják az adatot az adatbázisban. Ahogy ezek a módosítások megtörténnek, a tranzakciónapló nő. Mivel az egyes adatbázis-változások naplózásra kerülnek, az adatbázis-adminisztrátornak a tranzakciónapló méretét monitorozni kell. Az adatok állandóan változnak, ezért az adatbázisnapló állandóan nőni fog.
A tranzakciónapló egy írás előtti napló, ami azt jelenti, hogy a változásokat először naplózni kell, majd az adatbázistáblában csak ezután történhet meg az adat változása. Ha az adatbázis-módosítás sikeresen végbement a naplóban, akkor a tranzakció helyreállítása és így a konzisztencia is garantált.
Az adatbázis-kezelő rendszer ellenőrzési pontot hoz létre, hogy garantálja, hogy minden módosított adatbázislap biztonságban a lemezre íródott. Az adatbázisrendszer ellenőrzési pontjainak a gyakoriságát az adatbázis-adminisztrátor állíthatja be egy konfigurációs paraméter segítségével. Az ellenőrzési pont gyakoriságát általában egy előredefiniált intervallummal vagy a kiírt naplórekordok egy előredefiniált számával lehet megadni.
Az adatbázis-kezelő rendszer a naplóbejegyzések segítségével biztosítja az adatok konzisztenciáját. A tranzakciónaplót akkor használja az adatbázis-kezelő rendszer, ha az adatbázis-kezelő rendszer újraindul, ha a tranzakciót vissza kell görgetni, vagy ha az adatbázist egy előző állapotra vissza kell állítani. Vizsgáljuk meg az egyes eseteket.
Ha az adatbázis-kezelő rendszer újraindul, akkor egy tranzakció-helyreállítási folyamat indul el. Az tranzakció-helyreállítási folyamat alatt az adatbázis-kezelő rendszer ellenőrzi, hogy mely tranzakciókat kell visszagörgetni. Ez akkor történhet meg egy tranzakcióval, ha nem lehet tudni, hogy a módosítások a lemezre kerültek-e. Egy ellenőrzési pont kikényszeríti, hogy minden módosított adatlap a lemezre kerüljön. Ezért az ellenőrzési pont jelenti azt a pontot, ahonnan a helyreállításnak kezdődnie kell. Mivel minden ellenőrzési pont előtti adatlap-módosítás a lemezre került, nincs szükség az ellenőrzési pont „elé” visszagörgetni.
Ha a tranzakciót vissza kell görgetni, akkor az adatbázis-kezelő rendszer minden olyan módosításhoz az adatbázisba másolja a módosítás előtti képet, amely a tranzakció kezdete óta történt. Bővebbet az Adatbázis-helyreállítás című fejezetben olvashatunk.
A helyreállítás alatt az adatbázis-adminisztrátor a tranzakciónaplót használhatja, hogy az adatbázist előregörgesse. Először az adatbázis mentési másolatát állítja helyre, majd a mentés óta történt változások újra végrehajtásához az archivált naplóállományokat görgeti előre. Az előregörgetés alatt az adatbázis-kezelő rendszer az adatbázisba másolja az egyes módosítások hatása „utáni” képeket. A naplózott adat használatával az adatbázis-kezelő rendszer biztosítja, hogy az egyes módosítások ugyanabban a sorrendben legyenek alkalmazva, mint ahogy eredetileg bekövetkeztek.

Az adatbázisnapló konfigurációja

Az adatbázisnapló konfigurációja összetett feladat lehet. Az adatbázis-kezelő rendszertől függően

az adatbázis-adminisztrátornak lehet, hogy több konfigurációs döntést kell hoznia az adatbázisnaplóval kapcsolatban. Ezek a döntések lehetnek például: az olvasási és az írási naplópufferek definiálása, az naplóállományok definiálása vagy a naplóarchiválás beállítása.
Az adatbázisnaplóhoz az írási naplópuffer definiálása optimalizálhatja a naplóíró műveleteket. Az adatbázis-feldolgozás hatékonyabb, ha az adatbázisnapló-rekordok a memóriába kerülnek ahelyett, hogy közvetlenül a lemezre kerülnének. Az adatbázis-kezelő rendszer aszinkron módon tudja a naplórekordokat az írási naplópufferből a fizikai naplóállományba írni. Ha az adatbázis-naplózás ilyen módon van megvalósítva, akkor az adatbázis-feldolgozásnak nem kell várnia a szinkron lemezre történő naplóírásra.
Az adatbázisnaplóhoz olvasási naplópuffer definiálásával optimalizálhatjuk azokat a műveleteket, amelyek az adatbázisnaplót olvassák, mint a ROLLBACK, RECOVER műveletek.
Amikor az adatbázisnaplót konfiguráljuk, jó ötlet duál naplózást beállítani. A duál naplózással az adatbázis-kezelő rendszer a változásokat két szeparált és független naplóállományba fogja naplózni. A duál naplózás megvalósítása egy redundáns napló használatot eredményez, amely naplóhiba esetén lesz hasznos. Egy napló sok okból hibázhat, beleértve az eszközhibát vagy az egyszerű gondatlanságot. Duál naplózás beállításakor legyünk biztosak abban, hogy az egyes naplókat külön eszközökön helyeztük el és külön kezeljük, így minimalizáljuk azon lehetséges naplóhibákat, amelyek mindkét naplóban előfordulnak ugyanabban az időben.
Egy másik konfigurációs kérdés arról dönteni, hogy az adatbázisnapló-állományokat hogyan kezeljük, ha betelnek. A megvalósítás az adatbázis-kezelő rendszertől függ. Néhány adatbázis-kezelő rendszer automatikus naplókimentést (offloading) használ. A naplókimentés archiválási folyamat, ekkor a naplózás egy új aktív naplóba történik és a régi aktív napló tartalma egy archív naplóba mentődik.
Ha az adatbázis-kezelő rendszer automatikus archiválást hajt végre, akkor a teljesítményt azzal növelhetjük, hogy szalag helyett lemezre mentjük az archív naplót. A lemezre való archiválással a napló archiválási folyamat gyorsabban fut és a mentési és helyreállítási folyamatok is gyorsabbak lesznek, mert a naplórekordok lemezen vannak, ami nem csak gyorsabb input/outputot jelent, hanem a szalag csatolására sem kell várni. Az adatbázis-adminisztrátor egy tárolás-kezelő rendszert használhat, hogy automatikusan migrálja az archív naplót a szalagra egy előre meghatározott idő után.
A mentési-helyreállítási stratégia megvalósításához sok esetben az adatbázisnapló mentésére is szükség van. Ekkor a napló archiválását általában periodikusan el kell végezni. Az adatbázis-kezelő rendszer általában egy mentési parancsot biztosít a tranzakciónapló mentésére. Ha az adatbázis-kezelő rendszer befejezi a tranzakciónapló mentését, törli az adott tranzakciónapló-állomány tartalmát. Az adatbázis-kezelő rendszer később ezt az állományt aktív naplóállományként újrahasználja.

Minden adatbázis-művelet naplózva van?

Adatbázis-kezelő rendszertől függően bizonyos szituációkat és parancsokat lehet, hogy nem kell naplózni. Az adatbázis-kezelő rendszer kikapcsolhatja a naplózást, hogy elkerülje azt, hogy a gyorsan növekvő tranzakciónapló kifusson a helyből. Például a DDL műveletek és az adatbázis-segédprogramok futását nem szükséges naplózni. Óvakodjunk ezektől a szituációktól és tervezzünk a helyreállítási szükségleteinknek megfelelően.
Néhány nagy művelet alatt, mint a CREATE INDEX, az adatbázis-kezelő rendszer valószínűleg nem fog minden új adatlapot naplózni. Helyette az adatbázis-kezelő rendszer annyi információt fog az adatbázisnaplóba menteni, amennyi meghatározza, hogy egy CREATE INDEX történt, így egy előregörgetés esetén újra létre lehet hozni, egy visszagörgetés esetén pedig lehet törölni.
Továbbá néhány adatbázis-kezelő rendszer adatbázis szinten konfigurációs beállítást biztosít, hogy a naplózást bizonyos típusú feldolgozásoknál ki lehessen kapcsolni. Általában nagy mennyiségű adatváltozás esetén lehet ezt megtenni. A naplózás lelassíthatja ezeket a folyamatokat, ezért

ésszerűbb azt kikapcsolni. Ha ezek a műveletek nem lesznek naplózva a tranzakciónaplóba, akkor a műveletek után mentést kell végrehajtani, vagy egy esetleges helyreállítás esetén újra be kell őket tölteni.

Megnyitott adatbázis-objektumok

Az adatbázis-kezelő rendszernek az adatbázis-objektumokat meg kell nyitni, hogy az objektumokat kezelje és használja. Az egyes adatbázis-objektumok megnyitásához memóriaterületre van szükség. Az adatbázis-kezelő rendszereknél általában megadható egy olyan konfigurációs paraméter, amellyel a megnyitott adatbázis-objektumok számát lehet korlátozni. Az alapértelmezett értéket az idő múlásával a megfigyelések alapján lehet csökkenteni vagy növelni attól függően, hogy mennyi megnyitott adatbázis-objektumra van szükség, és mennyi memóriát tud az adatbázis-adminisztrátor biztosítani a számukra. Nem megoldás az, hogy a megnyitott adatbázis-objektumok számát úgy határozzuk meg, hogy minden adatbázis-objektum nyitva lehessen, mert ha egy adatbázis-objektum nyitva van, akkor memóriát fogyaszt. Memóriaterületből pedig mindig kevés van. A megnyitott adatbázis-objektumok számát ezért mérsékelni kell, amelyhez az adatbázis-adminisztrátor figyelembe veszi az adatbázis-megvalósítás méretét, az alkalmazásfeldolgozás típusát, és az elérhető memória méretét.

Zárolás és versengés

A párhuzamos feldolgozás támogatását segítő műveletek, mint a holtpontelemzés vagy a zárolás, beállításai nagy hatással lehetnek az adatbázis-teljesítményre. Az adatbázis-feldolgozás záráson alapszik. A zárolás biztosítja az adatok konzisztenciáját. Egyensúlyt kell teremteni a konkurencia és a teljesítmény között. A következő szituációk negatív hatással vannak a rendszer teljesítményére:

- Zárolás felfüggesztés: akkor fordul elő, ha egy alkalmazásfolyamat olyan zárat kér, amelyet már egy másik alkalmazásfolyamat fenntart és nem oszthat meg. A felfüggesztett folyamat ideiglenesen megáll, amíg a kért zárolás elérhetővé nem válik.
- Timeout: akkor történik, ha egy alkalmazásfolyamat megáll, mert tovább volt felfüggesztve, mint egy előre megadott időintervallum. Ezt az intervallumot általában egy konfigurációs paraméter adja meg.
- Holtpont: akkor alakul ki, ha két vagy több alkalmazási folyamat zárat tart fenn egy erőforráson, amelyre a másiknak szüksége van és amely nélkül nem haladhatnak tovább. A holtpont keresési ciklus, azaz két holtpont keresés közötti időintervallum is általában egy konfigurációs paraméter segítségével adható meg.

Egy relációs adatbázis esetén a zárolási folyamat nagyon összetett is lehet. Ez függ a feldolgozás folyamatától, az zár méretétől, melyet a tábla létrehozásakor határoztak meg (azaz, hogy milyen szemcsézettségű a zár: sorszintű, táblaszintű, stb.), a program vagy az SQL utasítás izolációs szintjétől, az adatelérés módjától, és az adatbázis-kezelő rendszer konfigurációs paramétereitől. Az adatbázis-zárolás hangolásához a rendszer, az adatbázis és az alkalmazás hangolása szükséges.

Rendszerkatalógus

A rendszerkatalógus fizikai helye és beállítása a rendszer teljesítményére hatással lesz. Az adatbázis-adminisztrátornak kell eldöntenie, hogy hová telepítse, milyen típusú lemezre, és mennyi helyet foglaljon le neki. Ezt a döntést általában telepítési időben kell meghozni.
A rendszerkatalógust egy szeparált lemezterületen helyezzük el, így más alkalmazás adatoktól függetlenül lehet kezelni és hangolni. Ha lehet egy vagy két lemezkötetet dedikáljunk a rendszerkatalógusnak. Az indexeket és a táblákat külön lemezköteten helyezzük el. Ha az adatbázis-kezelő rendszer még nem rendelkezik adatgyorsítóval a rendszerkatalógushoz, akkor különítsünk el neki egyet. Ez megkönnyíti a rendszer input/output műveletei hatékonyságának a nyomkövetését.

Ha a rendszerkatalógusban változtatást kell végezni, akkor segédprogramokat, mint REORG, COPY vagy RECOVER vagy állományrendszer parancsokat kell használni. A változások lehet, hogy növelik a rendszerkatalógust: pl. új index hozzáadásával vagy az adatbázis-kezelő rendszer új verzióra migrálásával.

Más konfigurációs opciók

Minden adatbázis-kezelő rendszernek sok konfigurációs és hangolási opciója van. Ezek az adott adatbázis-kezelő rendszernek a sajátosságai. Az adatbázis-adminisztrátornak ismernie és értenie kell ezeket az opciókat és azok hatásait.
Néhány példa:

- Beágyazott trigger hívások: Néhány adatbázis-kezelő rendszerben lehetőség van a beágyazott trigger hívások engedélyezésére és letiltására. Egy beágyazott trigger hívás akkor történik, ha egy trigger egy másik trigger indulását okozza. Néhány adatbázis-kezelő rendszer lehetővé teszi az egymásba ágyazott trigger hívások korlátozását egy maximális érték megadásával. Ha korlátozzuk a beágyazott trigger hívásokat az hatással lehet a teljesítményre is. Ha egy alkalmazás eléri a maximálisan beágyazottan meghívható triggerek számát, a triggereket vissza kell görgetni, ami teljesítménycsökkentést okozhat.
- Biztonsági opciók: adatbázis-kezelő rendszer konfigurációs opció vezérelheti a jogosultságok és a biztonság funkcionalitását. Néhány adatbázis-kezelő rendszer lehetővé teszi, hogy az adatbázis-biztonságot egy külső szoftver vezérelje.
- Elosztott adatbázisok: egy elosztott adatbázis implementációjának konfigurálásához az adatbázis-kezelő rendszer nagyon valószínű, hogy egy opciót biztosít a különböző helyeken lévő adatbázis-kapcsolódásokhoz.

Rendszer-monitorozás

Az adatbázis-kezelő rendszer környezetét állandóan figyelni kell a teljesítményproblémák és teljesítménycsökkenés miatt. Néhány adatbázis-kezelő rendszernek van beépített monitorozó eszköze.
A teljesítménymonitorozás a rendszerkörnyezetre is vonatkozik, nem csak az adatbázis-kezelő rendszerre. Monitorozni kell az operációs rendszert, a hálózatot, a tranzakciós szervert stb. Az adatbázis-adminisztrátornak tudnia kell kezelni és meg kell tudnia érteni a monitorozás outputját.

Összefoglalás

A fejezetben megvizsgáltuk, hogy az adatbázis-kezelő rendszer teljesítményére a környezetben lévő hardver és szoftverkomponensek milyen hatással vannak. Megvizsgáltuk a hardver, azon belül kiemelten a lemezes háttértároló, az operációs rendszer, és más komponensek hatását az adatbázis-kezelő rendszerre.
A következőkben az adatbázis-kezelő rendszer konfigurációs paramétereit vizsgáltuk. Az adatbázis-kezelő rendszereknél nagyon fontos kérdés, hogy a memóriát hogyan használják. Megvizsgáltuk, hogy az adatbázis-kezelő rendszerek általában milyen puffereket, gyorsítókat használnak: adatgyorsító, eljárásgyorsító, rendezési gyorsító, naplógyorsító. Megvizsgáltuk ezeknek a gyorsítóknak a funkcióit is. Részleteztük, hogy ezen kívül még milyen memóriaterületekre van szüksége az adatbázis-kezelő rendszernek.
Mélyebben részleteztük az adatgyorsítót, megvizsgáltuk, hogyan lehet hangolni és monitorozni. Az eljárásgyorsító esetén is megvizsgáltuk azt, hogy hogyan lehet monitorozni és hangolni.
A következőkben az adatbázisnapló hatását vizsgáltuk a teljesítményre, illetve megvizsgáltuk, hogy az adatbázis-adminisztráció milyen módon tudja a naplózáshoz kapcsolódóan a teljesítményt befolyásolni.

Az adatbázis-teljesítményére ugyancsak nagy hatással van az, hogy mennyi adatbázis-objektum van megnyitva, az, hogy a párhuzamos feldolgozáshoz kapcsolódó műveletek, mint zárolás, holtpont, versengés, hogyan jelennek meg az adatbázisban, az, hogy az rendszerkatalógus hogyan van elhelyezve és konfigurálva, illetve egyéb konfigurációs opciók.
Az adatbázis-kezelő rendszer teljesítmény témaköréhez mindenkor kapcsolódik a teljes rendszer monitorozása, ami nem csak az adatbázis-kezelő rendszert jelenti, hanem a hardver, az operációs rendszer és az más egyéb szoftverek monitorozását is.

Ellenőrző kérdések

1. Az adatbázis-adminisztrátor milyen kérdések megválaszolásával biztosíthatja az optimális hardverkörnyezetet az adatbázis-kezelő rendszer számára?
2. Mi az az SSD? Miért hasznos az adatbázis-kezelő rendszerek számára?
3. Az adatbázis-adminisztrátor milyen kérdések megválaszolásával biztosíthatja az optimális operációsrendszer-környezetet az adatbázis-kezelő rendszer számára?
4. Hogyan lehet az adatbázis-kezelő rendszer konfigurációs paramétereit módosítani?
5. Az adatbázis-kezelő rendszerek általában milyen gyorsítókkal rendelkeznek? Melyiknek mi a funkciója?
6. Az adatbázis-kezelő rendszernek a gyorsítókon kívül milyen memóriaterületekre lehet szüksége?
7. Az adatgyorsító milyen típusú adatlapokat tartalmaz? Melyik mit jelent?
8. Mekkora méretűre válasszuk az adatgyorsítót?
9. Mit jelent az olvasási hatékonyság? Mitől függ ez az érték? Az olvasási hatékonyságot milyen érték mellett tekintjük megfelelőnek?
10. Mi az az adatbázis-napló? Mire használja az adatbázis-kezelő rendszer?
11. Mit jelent a duál naplózás?
12. Mi az az ellenőrzési pont?
13. A naplózás támogatására milyen memóriaterületeket használ az adatbázis-kezelő rendszer?
14. Miért szükséges archiválni a naplót?
15. Milyen hatással vannak a megnyitott adatbázis-objektumok a teljesítményre?
16. Az adatbázis-objektumok zárolása és az objektumokért a versengés milyen hatással van a teljesítményre?
17. Hogyan kell elhelyezni a rendszerkatalógust a teljesítmény szempontjából?

14. fejezet – Adatbázis-teljesítmény

Az adatbázis-teljesítmény a tervezés, a paraméterek, az adatbázis-objektumok fizikai konstrukciója, a táblák, az indexek, és az állományok hangolásával és optimalizálásával foglalkozik. Az adatbázis-objektumok aktuális struktúráját folyamatosan monitorozni és változtatni kell, ha az adatbázis hatékonysága ezt megköveteli. Egy gyengén tervezett és szervezett adatbázisban a lekérdezések teljesítményének optimalizálására nem elegendő az SQL utasításokat és a rendszert hangolni.

Az adatbázisok optimalizálásának technikái

Az adatbázis-adminisztrátornak az adatbázis-struktúrák teljesítményének optimalizálásához az adatbázis-kezelő rendszer eszközei közül a megfelelő technikát kell alkalmaznia. A legtöbb adatbázis-kezelő rendszer támogatja a következő technikákat, legfeljebb más-más néven:

- particionálás: egy adatbázistáblát több részre osztunk, amelyek külön állományban tárolódnak
- raw partíció vagy állományrendszer
- indexelés
- denormalizálás
- szabad hely biztosítása az adatnövekedéshez
- tömörítés
- állományok megfelelő elhelyezése
- adatlap méretezése: megfelelő adatlapméret a hatékony adattároláshoz és az input/output műveletekhez
- újraszervezés

Particionálás

Egy adatbázistábla az adatok halmazának logikai megjelenése, amely fizikailag a tárolón helyezkedik el. Az egyik döntés, amit az adatbázis-adminisztrátornak minden tábla esetén meg kell hoznia, hogy hogyan tárolja az adatokat. A különböző adatbázis-kezelő rendszerek különböző mechanizmusokat biztosítanak arra, hogy a fizikai állományokat az adattáblákhoz rendeljük. Az adatbázis-adminisztrátornak választania kell a következő lehetőségek közül, hogy az egyes táblákat állomány(ok)hoz rendelje:

- Egy táblát egy állományhoz: Ez a legáltalánosabb választás. Minden, a táblába beszúrt sor ugyanabban az állományban van. Nem biztos, hogy ez a leghatékonyabb megoldás.
- Egy táblát több állományba: Ezt az opciót használják a leggyakrabban a nagy táblákhoz vagy az olyan táblákhoz, amelyekhez szükség van tárolási szinten fizikailag elkülöníteni az adatokat. Adatbázis-kezelő rendszertől függően egy táblát úgy lehet több állományhoz rendelni, hogy az adatbázis-adminisztrátor a táblát partíciókra bontja és különböző táblaterekhez rendeli, vagy a táblához rendelt táblateret particionálja.
- Több tábla egy állományba: a kis táblákhoz (mint pl. a kódtáblák) használják ezt az opciót.

A particionálás segíti a párhuzamos adatfeldolgozást. A párhuzamos adatfeldolgozás esetén az adatbázis-kezelő rendszer az adatbázis adatait több folyamat segítségével éri el. Lehet, hogy az adatbázis-kezelő rendszer egy SQL utasításhoz több párhuzamos kérést hív meg, amely több szimultán olvasó motort használ. A párhuzamosság lényegesen csökkentheti az adatbázis-lekérdezésekhez szükséges időt.

Raw partíció vagy állományrendszer

Az adatbázis-adminisztrátornak az adatok tárolásához választania kell az állományrendszer és a raw partíció között. A témáról még az Adat- és tároláskezelés című fejezetben is olvashatunk.
Állományrendszer esetén az írásokat az operációs rendszer puffereli. Amikor az operációs rendszer az írásokat puffereli, az adatbázis-kezelő rendszer nem tudja, hogy az adatok fizikailag a lemezen vannak-e vagy sem. Ha az adatbázis-kezelő rendszer gyorsítótár-kezelője (cache manager) próbálja az adatokat a lemezre írni, az operációs rendszer késleltetheti az írást, mert az adat még az állományrendszer gyorsítótárában van. Ha hiba történik az adatbázis adatai nem lesznek 100%-ig helyreállíthatóak.
Ha raw partíciót használunk, az adat az adatbázis-gyorsítótárból közvetlenül a lemezre kerül, nincs köztes állomány-rendszerbeli vagy operációs rendszerbeli gyorsítótár. Ha az adatbázis-kezelő rendszer gyorsítótár-kezelője írja az adatot a lemezre, akkor az beavatkozás nélkül íródik fizikailag a lemezre. A raw partíció esetén az adatbázis-kezelő rendszer biztosítja, hogy elég hely legyen elérhető. Állományrendszer esetén az operációs rendszer kezeli az adatbázis használatához szükséges tárhelyet.
Teljesítmény tekintetből nézve nem előnyös a kétszeres gyorsítótárazás, azaz az állomány-rendszerbeli vagy operációs rendszerbeli gyorsítótárazás és az adatbázis-kezelő rendszerbeli gyorsítótár. Az adatbázis-kezelő rendszer gyorsítótára elegendő. Sőt a plusz munka, hogy másodjára is gyorsítótárazzuk az adatot, erőforrás-igényes, ezért negatívan befolyásolja az adatbázis-műveletek teljesítményét.

Indexelés

Talán az egyik legnagyobb teljesítményhangolási technika az, ha az adatbázis tábláira az adatbázis-adminisztrátor indexeket hoz létre. Az indexeket alapvetően a teljesítmény növelésére használják. Az indexek különösen hasznosak a:

- a táblák összekapcsolásában, ha az adatbázis-adminisztrátor indexet definiál a kapcsolóoszlopon, akkor az összekapcsolás megvalósítása sokkal hatékonyabb lesz
- táblák közötti adatok összehasonlításánál
- az adatok csoportosításánál
- az adatok rendezésénél

Indexek nélkül az adatbázis adatainak elérése úgy menne végbe, hogy az adatbázis-kezelő rendszer egy tábla minden sorát végignézné. Ez nagyon nagy tábláknál nem hatékony.
Az indexek tervezése és létrehozása két teljesítmény-témakörhöz is odasorolható, hiszen az adatbázis-teljesítmény és az alkalmazásteljesítmény hangolásához egyaránt hozzátartozik. Az indexeket az adatbázis-adminisztrátor hozza létre CREATE utasítással.
Mielőtt az adatbázis-adminisztrátor az adatbázist indexek létrehozásával hangolná, fel kell mérnie, hogy mi lesz az index hozzáadásának a hatása. Ehhez az adatbázis-adminisztrátornak ismernie kell, hogyan éri el az adatbázis-kezelő rendszer annak a táblának az adatait, amelyre az indexet teszi. Hasznos információkat tartalmaz azon utasítások százaléka, amelyek inkább lekérdezik, mint módosítják a táblát; illetve az a teljesítményküszöb, amelyet a szolgáltatási szintről szóló megállapodás határoz meg a táblán való lekérdezésekhez. Az adatbázis-adminisztrátornak azt is vizsgálnia kell, hogy az új index hozzáadása milyen hatással lesz a későbbiekben az olyan adatbázis-segédprogramokra, mint a betöltés, az újraszervezés, és a helyreállítás.
Az adatbázis tervezésének az egyik nagy kérdése, hogy mennyi indexet kell egy táblához létrehozni. Az adatbázis-adminisztrátor csak a saját tapasztalatára támaszkodhat, amikor meghatározza az indexek megfelelő számát az egyes táblákhoz. Ezt a döntését arra alapozhatja, hogy az adatbázis-lekérdezések optimalizáltak legyenek és az adatbázis-beszúrások, módosítások, törlések teljesítménye ne sérüljön. Annak a meghatározása, hogy az egyes táblákra mennyi a

megfelelő számú index, az adatbázis és az adatbázist elérő alkalmazások elemzése szükséges.
Az indexelemzés általános célja, hogy az adatbázis-kezelő rendszer kevesebb input/output műveletet használjon a táblára vonatkozó lekérdezések kiszolgálásához. Egy index néhány lekérdezésen segíthet, azonban másokat akadályozhat. Az adatbázis-adminisztrátornak minden alkalmazás esetén fel kell becsülnie egy index hozzáadásának hatását, nem csak egy lekérdezésre vonatkozóan.
Egy index pozitívan hat a teljesítményre, ha kevesebb input/output művelet használatával kapjuk meg egy lekérdezés eredményét. Egy index negatívan hat a teljesítményre a módosításnál, ha az adat módosításánál az indexet is módosítani kell. A hatékony indexstratégia megvalósítása esetén az adatbázis-adminisztrátor a lehető legnagyobb mértékben igyekszik csökkenteni az input/output műveletek számát, de mindezt úgy, hogy közben az adatbázis-kezelő rendszer az indexek naprakészen tartására még elfogadható mértékű erőforrás-ráfordítást használjon fel.
Ha új indexet hozunk létre, akkor teszteljük le teljesen az index által támogatott lekérdezések teljesítményét. Teszteljük az adatbázis adatait módosító utasításokat, hogy megmérjük az új indexek frissítésével járó terhelést, nézzük meg a processzor időt, az eltelt időt, és az input/output műveletek követelményeit, hogy biztosak legyünk, hogy az index valóban a teljesítmény javulását segíti elő. Ne feledjük el, hogy a hangolás egy iteratív folyamat és időt vesz igénybe a változás hatásának meghatározása. Az index létrehozására nincsenek szabályok, próbáljunk ki különböző indexvariációkat és mérjük le az eredményt.

Mikor kerüljük az indexelést?

Ha a tábla túl kicsi, kevesebb mint 10 adatlap, akkor kerüljük az indexet. Az indexelt elérés egy kis tábla esetén kevésbé hatékony. Mivel az index olvasáshoz is input/output műveleti költség járul, nem kerül többe a tábla minden sorát végignézni.
Az index input/output műveletek ellenére egy kis tábla esetén is lehet előnye az indexnek, pl. ha az egyediséget ki akarjuk kényszeríteni, vagy ha a legtöbb adatkinyerés csupán egy sor elérése az elsődleges kulcs segítségével, illetve bonyolult funkcionális indexek esetében.
Ha az adatbázis-kezelő rendszer az indexben a változó hosszúságot kiterjeszti a lehető legnagyobb hosszúságra, akkor esetleg érdemes a változó hosszúságú oszlopok indexelését is kerülni. Ez a kiterjesztés azt eredményezheti, hogy az indexek mértéktelen mennyiségű lemezterületet használnak és nem hatékonyak. Ha a változó hosszúságú oszlopokat a WHERE utasításrészben használjuk, az indexhez használt lemezterület költségét össze kell hasonlítani a teljes keresés költségével.
Kerüljük az indexelést azon tábláknál, ahol az elérésnél mindig a teljes táblát keressük, és az olyanoknál, ahol soha nem használunk WHERE utasításrészt.

Index túlterhelés

Az indexek általában a SELECT utasításnak a WHERE részén alapulnak. Tekintsük a következő lekérdezést:

```
SELECT DOLGOZO_AZON, KERESZTNEV, FIZETES
FROM DOLGOZO
WHERE FIZETES >150000;
```

Ha olyan indexet hozunk létre, amelyben a FIZETES oszlop szerepel, az ennek a lekérdezésnek a teljesítményét növelheti. De az adatbázis-adminisztrátor a teljesítményt tovább növelheti, ha az indexet túlterheli a DOLGOZO_AZON, és a KERESZTNEV oszlopokkal. A túlterhelt index használatával az adatbázis-kezelő rendszer a lekérdezés eredményét csak az indexben tárolt adatok alapján visszaadhatja. Az adatbázis-kezelő rendszernek nincs szüksége további input/output műveletekre, hogy elérje a tábla adatait, mivel a lekérdezés által kért minden adat benne van a túlterhelt indexben.

Az adatbázis-adminisztrátornak érdemes megfontolnia a túlterhelt indexek használatát, hiszen így támogathatja a lekérdezések olyan módú megvalósítását, amelyben az adatbázis-kezelő rendszer csak az indexet használja, és a táblát nem. A túlterhelt indexeket akkor is érdemes használni, ha több lekérdezés számára lehet előnyös az index vagy az egyes lekérdezések nagyon fontosak.

Denormalizálás

A relációs adatbázis-kezelő rendszerek általában támogatják a denormalizált táblák kezelését.
A denormalizálás a normalizálás ellentéte. Denormalizálás esetén a táblákban redundáns módon jelennek meg az adatok. Ez gyorsítja az adatok kinyerését, viszont az adatokat több helyen kell karbantartani, így az adatok módosítása lassabb lesz. A denormalizált tábla jó döntés lehet akkor, ha a teljesen normalizált séma nem működik optimálisan.
Egy relációs adatbázis denormalizálásának egyetlen oka a hatékonyság növelése. A következő opciókat kell figyelembe venni:

- előre összekapcsolt táblák: amikor a táblákat általában összekapcsolva használjuk, és a kapcsolat létrehozása sok időt és erőforrást emészt fel
- riport tábla: amikor bizonyos kritikus riportokat túl költséges generálni
- tükör tábla: amikor a táblára két különböző környezetnek párhuzamosan van szüksége
- osztott táblák: amikor két különböző csoport egy tábla különböző részeit használja
- kombinált táblák: az egy-egy vagy egy-sok kapcsolatot egy táblába egyesíti
- fizikai denormalizáció: bizonyos adatbázis-kezelő rendszer jellemzők kihasználása

Tekintetbe kell még venni:

- A redundáns adatok olyan táblákban tárolását, amelyek segítségével csökken a táblák összekapcsolásának száma.
- Az ismétlődő csoportok (pl. egy személy telefonszámai) egy sorban tárolását, így csökken az input/output műveletek száma és a szükséges lemez terület.
- A származtatott adatok tárolását, hogy elkerüljük a számításokat és a költséges algoritmusokat.

Ki kell hangsúlyozni, hogy csak az üzletileg megtérülő mértékben szabad bevállalni anomáliákat vagy nagyobb karbantartási költségeket.

Szabad hely

A szabad helyre azért van szükség, hogy a táblatér adatlapjainak egy része üres és elérhető legyen, amikor új adatot akarunk benne tárolni. Az adatlapokon szükséges szabad hely nagyságát táblaterenként vagy esetleg adatbázis-objektumonként határozhatja meg az adatbázis-adminisztrátor. Ha egy táblatér adatlapjaihoz kellő mennyiségű szabad helyet határozunk meg, akkor az csökkentheti az újraszervezések gyakoriságát, csökkentheti a versengést és növelheti a beszúrások hatékonyságát. Az adatbázis-kezelő rendszerek valamilyen módon biztosítják, hogy egy adatbázis-objektumhoz a CREATE és az ALTER utasításban az adatlaponkénti szabad helyek mennyiségét meghatározzuk. Ez lehet a PCTFREE paraméter, amelyben az adatbázis-adminisztrátor egy százalékos értéket ad meg, amely azt jelöli, hogy az adatlapnak hány százaléka legyen szabad, azaz mennyi helynek kell elérhetőnek lennie a jövőbeli beszúrásokhoz. Azonban adatbázis-kezelő rendszerenként más-más eszközök és módszerek lehetnek a beszúrásokhoz szükséges szabad hely biztosításához.
Az egyes adatbázis-objektumokhoz a megfelelő mennyiségű szabad hely biztosítása a következő előnyökkel jár:

- a beszúrás gyorsabb, ha szabad hely érhető el
- változó hosszúságú soroknak és módosított soroknak van helyük nőni
- ha egy adatlapon kevesebb sor van, az jobb konkurenciát eredményez, mert ha az adatlap

zárolva van, más felhasználóknak kevesebb adat elérhetetlen

A szabad helynek hátrányai is vannak:

- a tárolási követelmények nagyobbak
- a keresés tovább tart
- ha egy adatlapon kevesebb sor van, akkor több input/output művelet szükséges ahhoz, hogy megkapjuk a kért információt
- mivel az adatlaponkénti sorok száma csökken, az adatgyorsítás hatékonysága csökkenhet, mert kevesebb sor nyerhető vissza egy input/output művelettel

Az adatbázis-adminisztrátornak adatbázis-objektumonként biztosítania kell az elérhető szabad helyek megfelelő mennyiségét, amely azzal jár együtt, hogy időről időre meg kell azt vizsgálnia. A szabad helyek megfelelő összegét a következőkre kell alapozni:

- a beszúrások és módosítások gyakorisága
- a sorláncolás, sormigráció, és az adatlaptörések valószínűsége

Statikus táblához ne definiáljunk szabad helyet, mert nem lesz szüksége több helyre.

Tömörítés

A tömörítés használható az adatbázis méretének csökkentésére. Az adatok tömörítésével az adatbázisnak kevesebb lemezterületre van szüksége. Néhány adatbázis-kezelő rendszer az adatbázis-állományok tömörítéséhez belső DDL opciókat biztosítanak. Ha ilyen nem érhető el, akkor a tömörítéshez más cégek szoftvereit is megvásárolhatjuk.

Ha a tömörítés adott, akkor az adat az adatbázisba szúráskor betömörítésre, olvasáskor kitömörítésre kerül. A tömörített adat írása és olvasása több processzorbeli erőforrást igényel, mint a nem tömörített adat írása és olvasása: a adatbázis-kezelő rendszernek a be- és kitömörítő kódot is futtatnia kell, ha a felhasználó beszúrja, módosítja vagy olvassa az adatot.

Az input/output adatlapszinten történik. Mivel egy adatlapon több sor fér el, egy input/output több adatot nyer ki, amely optimalizálja az adatkeresés teljesítményét és növeli annak a valószínűségét, hogy az adat a gyorsítótárban van.

A tömörítést az adatbázis-adminisztrátornak mindig mérlegelnie kell. Az egyik oldalról lemezterületet mentünk meg és csökkenhet az input/output költség. Másrészt nagyobb lesz a processzor terhelése az adatok be- és kitömörítése miatt.

Nem minden adatbázis-index vagy tábla esetén érdemes az adatbázis-adminisztrátornak elgondolkodnia a tömörítésen. Bizonyos esetekben lehetséges, hogy a tömörített állomány nagyobb lesz, mint a tömörítés nélküli.

Állomány elhelyezés

Az adatbázis adatait tartalmazó állományok helye meghatározó hatással lehet a teljesítményre. Egy adatbázis esetén nagyon sok input/output művelet van. Az adatbázis-adminisztrátornak mindent meg kell tennie, hogy a fizikai lemez írásának és olvasásának a költségét minimalizálja. Ehhez ismernie kell, hogy az adatbázis-kezelő rendszer az egyes adatelemeket hogyan éri el, és az adatokat a fizikai lemezeszközön úgy kell elhelyeznie, hogy a teljesítmény optimális legyen.

Először is az indexeket és az adatokat tartalmazó állományokat különítsük el, ha lehet. Az adatbázis-lekérdezésekhez gyakran a tábla és a táblához tartozó index adatainak elérésére is szükség lehet. Ha mindkettő ugyanazon a lemezen van, akkor az valószínűleg negatív hatással van a teljesítményre. A lemezről történő adatkinyeréshez az író-olvasó fej mozog a lemezfelületen, hogy az adatot tároló fizikai blokkokat a lemezről kiolvassa. Ha egy művelet ugyanarról a lemezeszközön levő állományokból kéri az adatot, lappangás történik: annak a folyamatnak, amely az egyik állományból olvasna, várnia kell, amíg a másik állományból az olvasás történik. Ha az adatbázis-kezelő rendszer az indexeket és az adatokat ugyanarra a lemezre teszi, az indexeket és az adatokat

nem tudja az adatbázis-kezelő rendszer párhuzamosan olvasni.
Másik szabály az állományelhelyezéshez, hogy az alkalmazások adatkéréseit elemezzük, és válasszuk külön azon táblák állományait, amelyeket gyakran együtt kérnek le. Az ok ugyanaz, mint az indexek és adatok esetén.
Az utolsó állományelhelyezési elvet akkor használhatjuk, ha egy tábla több állományba van tárolva, azaz particionált táblák esetén. Ekkor az egyes állományokat helyezzük különböző lemezeszközre, hogy támogassuk és optimalizáljuk a párhuzamos adatbázis-műveleteket. Ha az adatbázis-kezelő rendszer egy lekérdezést részekre törhet, hogy párhuzamosan futtassa, akkor a particionált táblák több állományának a különböző lemezeszközökön való elhelyezése minimalizálja a lemez-lappangást.

Adatbázisnapló elhelyezése

Ha a tranzakciónaplót az aktuális adatoktól elkülönítve helyezzük el, akkor az adatbázisnaplót az adatbázis-kezelő rendszer úgy tudja írni, hogy közben az adatbázis adatainak a lemezre írására és a lemezről olvasására nem kell várnia. Ha ugyanazon a lemezmeghajtón lévő két állományba ugyanabban az időpontban ír az adatbázis-kezelő rendszer, az a teljesítményre negatív hatással van. Ez a teljesítménycsökkenés írás esetén sokkal nagyobb, mint ha az adatokat olvasnánk. Ha az adatbázisnaplót elkülönítjük, akkor minimalizáljuk a párhuzamos írást ugyanarra a lemezre. Hiszen tudjuk, hogy minden adatbázis-módosítás a tranzakciónaplóba is bekerül.
Az adatbázisnapló elkülönítése a mentésnél is hasznos. Az adatbázisnaplót minden naplóváltáskor menteni kell, amely sokkal többször fordul elő, mint a teljes adatbázis mentése. Ha a tranzakciónaplót elkülönítjük az adatoktól, akkor a napló mentése kevésbé fogja a rendszer teljesítményét hátrányosan befolyásolni.

Elosztott adatok elhelyezés

Az adat elhelyezés célja az elérés optimalizálása. Ezt általában az adatbázis-adminisztrátor azzal teszi meg, hogy a fizikai eszközön csökkenti a versengést. Egy kliens/szerver környezetben ez a cél kibővül az alkalmazás-teljesítmény optimalizálásával, mégpedig úgy, hogy az adatbázis-adminisztrátor igyekszik a hálózati átviteli költségeket csökkenteni. Ehhez az adatnak azon az adatbázisszerveren kell lennie, ahol a legvalószínűbb vagy a leggyakoribb, hogy használják. Azaz az adatot igyekszik az adat használójához a legközelebb elhelyezni.
Az elosztott adatokról az Elosztott adatbázisok című fejezetben olvashatunk még.

Adatlapméret

A legtöbb adatbázis-kezelő rendszer esetén meg lehet határozni az adatlap méretét. Az adatlapról bővebben az Adat- és tároláskezelés című fejezetben olvashattunk.
Van olyan adatbázis-kezelő rendszer, amely teljesen korlátozza az adatlap méretét, azonban a legtöbb adatbázis-kezelő rendszer esetén az adatlap méretét az adatbázis-adminisztrátor választhatja meg. Az adatlap méretére azonban a lemezen használt állományrendszer, az operációs rendszer, és az adatbázis-kezelő rendszer is korlátokat határoz meg. Az adatbázis-adminisztrátornak a legjobb adatlapméret kiszámítását a sorok méretére, az adatlaponkénti sorok számára, és az adatlapon megkövetelt szabad hely mennyiségére alapozhatja.
A megfelelő adatlapméret kiválasztása fontos feladat, mert az adatbázis-adminisztrátornak ez az egyik eszköze arra, hogy az adatbázis input/output műveleteinek a teljesítményét optimalizálja.
Nagyméretű adatrekord akkor fordulhat elő, ha egy táblában olyan adattípusokat tárolunk, mint szövegek, képek, vagy más bináris nagyméretű objektumok. Ha más típusú rekordoknál is előfordul az, hogy az adatrekord nagyobb, mint az adatlap, akkor az adatbázis-adminisztrátor a telepítésnél nem jól választotta meg az adatlap méretét. Ez pedig teljesítményproblémákhoz vezethet.

Újraszervezés

A relációs technológiával és az SQL utasításokkal könnyű az adatmódosítás. Egy INSERT, UPDATE vagy DELETE utasítás a megfelelő WHERE utasításrésszel, és az adatbázis-kezelő rendszer elvégzi az aktuális navigációt és módosítást. Ahhoz, hogy ez az absztrakciós szint biztosíva legyen, az adatbázis-kezelő rendszer a háttérben nagyon sokat dolgozik. Többek között meg kell valósítania a fizikai adatbázisban az adatok módosítását, azaz adatot kell elhelyezni, módosítani vagy törölni a lemezen, amely esetleg adatmozgatással is jár. Elméletileg ez mindenkinek jó. A programozói interfész egyszerű, a relációs adatbázis-kezelő rendszer elvégzi a munka nagy részét: az adatok aktuális helyének manipulálását. Azonban a dolgok sajnos nem ennyire egyszerűek. Az adatbázis-kezelő rendszer fizikai adatkezelésének a módja teljesítményproblémákat okozhat.

Minden adatbázis-adminisztrátor találkozott már olyan szituációval, ahol egy olyan lekérdezés vagy alkalmazás, amelynek eddig a teljesítményével nem volt gond, a termelési rendszerben lelassul egy idő után. Ennek a lelassulásnak sok lehetséges oka lehet, például megnőtt a tranzakciók száma, vagy kiterjedt az adatmennyiség. A teljesítményproblémát az adatbázis szétszórtsága is okozhatja. Az adatbázis szétszórtsága akkor áll elő, amikor egy adatbázis logikai vagy fizikai tárolási allokációi sok különálló tárterületet tartalmaznak, amelyek túl kicsik, fizikailag nem folytonosak vagy nagyon távol vannak egymástól. Tekintsük át a leggyakoribb okokat:

- Töredezettség: sok a szétszórt, kis méretű tárolási terület. Ez elpazarolt helyet eredményez, amely a teljesítményt csökkenti. Mégpedig azért, mert ugyanazt az adatmennyiséget egy töredezett tárolási területről sokkal több input/output művelet segítségével lehet kinyerni, mint egy nem töredezett területről.
- Sorláncolás és sormigráció: akkor történik, ha a módosított adat nem fér el arra a helyre, amelyen eredetileg volt, és az adatbázis-kezelő rendszernek új helyet kell találnia a frissített sornak. A sorláncolással az adatbázis-kezelő rendszer a módosított adatsor egy részét a táblatéren belül mozgatja egy olyan helyre, ahol elég szabad hely létezik. A sormigráció esetén az adatbázis-kezelő rendszer a teljes sort máshova mozgatja a táblatéren belül. Az adatbázis-kezelő rendszer mindkét esetben mutatókat használ, amely a sor maradék részére vagy a teljes sorra mutat. A sorláncolás és a sormigráció esetén is egy sor olvasásához több input/output művelet szükséges. A teljesítmény sérül a többszörös input/output művelet miatt.
- Állománykiterjesztések: a kiterjesztés egy újabb állomány, amely az eredeti állományhoz van kötve és csak az eredeti állománnyal együtt lehet használni. Ha egy táblatér által használt állomány kifut a helyből, egy állománykiterjesztést kap. Az állománykiterjesztések nem folyamatosan tárolódnak az eredeti állománnyal. Ha vannak állománykiterjesztések, az adatkéréseknek az adatot állománykiterjesztésről állománykiterjesztésre nyomon kell követniük és ez szükségtelen túlterhelés jelent. Az újraszervezés megszüntetheti a problémát.

Az adatbázis-kezelő rendszertől függően még sok oka lehet a szétszórtságnak. Például ha több tábla van egy táblatérben és az egyik táblát eldobjuk, akkor a tárterület visszaszerzéséhez lehetséges, hogy a táblateret újra kell szervezni.

A szervezetlen adatstruktúrák javítására az adatbázis-adminisztrátor adatbázis vagy táblatér újraszervező segédprogramot (REORG) futtathat, hogy kikényszerítse az adatbázis-kezelő rendszertől az adatbázis-objektum újra strukturálását. Így az adatbázis-adminisztrátor megszüntethet pár teljesítményproblémákat okozó okot, mint a töredezettség, vagy a sorláncolás. Az újraszervezés elsődleges előnye, hogy az újraszervezés után az adatbázis-funkciók gyorsabbak és hatékonyabbak lesznek, mivel az adatok a lemezen optimálisabb módon lesznek elhelyezve. Az újraszervezés maximalizálja az adatbázisok elérhetőségét és megbízhatóságát.

A táblatereket és indexeket is lehet újraszervezni. Az adatbázis-kezelő rendszertől függ, hogy az

adatbázis-adminisztrátor hogyan futtatja a REORG-ot. Néhány adatbázis-kezelő rendszernek van beépített újraszervező segédprogramja. Más adatbázis-kezelő rendszerek nem rendelkeznek ilyennel, ekkor másik szállítótól lehet venni egyet.
Az adatbázis-adminisztrátor manuálisan is újraszervezheti az adatbázist, mégpedig úgy, hogy a teljes adatbázist újraépíti. Ez egy nagyon bonyolult feladat, sok, bonyolultan összekapcsolt lépést kell hozzá végrehajtani. Ha az újraszervezéshez érhető el segédprogram, a folyamat leegyszerűsödik. Néha csak egyetlen egyszerű parancs szükséges hozzá.
A hagyományos újraszervezéshez az adatbázist le kell állítani. Néhány REORG segédprogram online adatbázis mellett is végrehajtja az újraszervezést. Az ilyen újraszervezés adatmásolat készítéssel megy végbe. Az online REORG segédprogram újraszervezi a másolatot, miközben az eredeti adatok elérhetőek maradnak. Ha a másolt adatok újraszerveződtek, az online REORG az adatbázisnaplót használja, hogy a másolaton is végbemenjenek az adatváltozások, amelyek a folyamat alatt történtek. Ha a másolat felfrissült, a REORG áthelyezi a vezérlést az eredeti táblatérről a másoltra. Egy online újraszervezéshez tárterület szükséges. Az online újraszervezést érdemes olyan időszakban elvégezni, amikor az eredeti adatbázisban kevés tranzakció történik. Ellenkező esetben az újraszervezett adatbázisban a változások átvezetése nagyon sok időt emészt fel.

Mikor kell újraszervezni?

A rendszerkatalógus segíthet annak a meghatározásában, hogy mikor kell egy adatbázis-objektumot újraszervezni. Az adatbázis-kezelő rendszerek általában rendelkeznek egy olyan eszközzel, amely végigolvassa az adatbázis-tartalmat és az egyes adatbázis-objektumokról statisztikai információkat ment el. Adatbázis-kezelő rendszertől függően ez a statisztikai információ vagy a rendszerkatalógusban vagy egy speciális adatlapon az adatbázis-objektumon belül tárolódik.
Néha nehéz nyomon követni a szétszórtság más okait. Néhány adatbázis-kezelő rendszer statisztikát gyűjt a töredezettségről, a sorláncolásról, a sormigrációról, az eldobott objektumokhoz dedikált helyekről, amelyek segíthetnek felfedezni a rendezetlenség okait.
A táblatér nem az egyetlen adatbázis-objektum, amelyet újra lehet szervezni. Az indexeknél is előnyös (sőt szükséges – ha az indexelt tábla változása ezt igényli) lehet az újraszervezés. Ahogy a táblába új adat kerül, vagy a tábla módosul, az indexnek is módosulnia kell. Az ilyen változások okozhatják azt, hogy az index szétszórttá válik.

Automatizálás

Ha lehetséges, az adatbázis-adminisztrátornak automatizálni kell az újraszervezést. Az automatizálási eszköz lekérdezheti az adatbázis-statisztikát és elindíthatja az újraszervezést azoknál az objektumoknál, amelyek a statisztika szerint elérnek egy küszöbértéket.
Az újraszervezés költséges lehet, mert leállással és az erőforrások terhelésével jár. Nehéz meghatározni, hogy az újraszervezés mikor javít a teljesítményen. A bölcs adatbázis-adminisztrátor megtervezi és ütemezi az újraszervezéseket, hogy feloldja a szétszórtsági problémákat az adatbázisban.

Összefoglalás

A fejezetben áttekintettük, hogy milyen lehetőségeink vannak a relációs adatbázis-kezelő rendszerben az adatbázis-teljesítmény javítására. Különböző technikákat néztünk meg, mint: particionálás, raw partíció vagy állományrendszer, indexelés, denormalizálás, klaszterezés, illesztett adatok, szabad hely biztosítása, tömörítés, állomány megfelelő elhelyezése, adatlap megfelelő méretezése és az újraszervezés. Ezekről a technikákról részletesen olvashatunk a fejezetben.

Ellenőrző kérdések

1. Milyen technikákat soroltunk fel az adatbázis optimalizálására?
2. Mit jelent a particionálás? Hogyan támogatja a teljesítmény javulását?
3. Mi az a raw partíció?
4. Az indexelés hogyan támogatja a teljesítményt? Milyen esetekben hasznos az index?
5. Mennyi indexet kell egy táblára létrehozni?
6. Csak pozitívan hat az index az adatbázis-teljesítményre?
7. Új index létrehozásakor miket kell tesztelni?
8. Mikor kerüljük az indexelést?
9. Mit jelent az index túlterhelése?
10. Mit jelent a denormalizálás? Milyen lehetőségei vannak?
11. Miért szükséges szabad helyet biztosítani az adatbázisbeli táblatereken?
12. Hogyan lehet meghatározni, hogy egy táblatérben mennyi hely maradjon szabadon?
13. Milyen előnyei és hátrányai vannak annak, hogy a táblatérben szabad helyek maradnak?
14. Mire alapozhatja az adatbázis-adminisztrátor azt, hogy mennyi szabad helyet hagyjon az egyes táblaterekhez vagy adatbázis-objektumokhoz?
15. A tömörítés milyen hatással van a teljesítményre? Miért?
16. Milyen szempontok szerint kell az adatbázis-adminisztrátornak az adatbázis-állományait a lemezen elhelyezni?
17. Hogyan befolyásolja az adatlap mérete a teljesítményt?
18. Miért szükséges újraszervezni az adatbázist?
19. Hogyan lehet újraszervezni az adatbázist?
20. Mikor kell újraszervezni az adatbázist?
21. Lehet-e automatizálni az újraszervezést?

15. fejezet – Adatbázis-változáskezelés

A mai üzleti környezetben az egyetlen állandó dolog: a VÁLTOZÁS. Az állandóan változó piaci igényeknek meg kell felelni. Ezért az üzletnek, így a saját cégünknek is a piaccal együtt kell változni. Az üzlet mindig növekedésre törekszik, illetve próbálja fenntartani a profitképességet. Ahhoz, hogy a cégünk lépést tudjon tartani, a cégnek állandóan változnia kell, növelnie kell a termékek és szolgáltatások számát, illetve javítania kell a minőségét.
Az üzlet változása a cégen belül az emberekre külön-külön is hatással van, az embereknek, az ott dolgozóknak is változniuk, változtatniuk kell. Általában a változással való foglalkozást a dolgozók nehéznek találják. A változás legtöbbször új feladatokat és ezzel új felelősségeket jelent az alkalmazottak számára, amelyek megnehezítik a munkát. A változáskezelést sok szempontból elemezhetjük. Ezek közül az egyik szempont az IT oldali közelítés. Tekintsünk néhány egyszerű példát a változásra:

- Fizikai környezet vagy munkahely változása: több vagy kevesebb alkalmazott foglalkoztatása vagy egyszerűen csak egy alkalmazott cseréje, és az új alkalmazott új vagy különböző ismeretekkel rendelkezik, mint a régi.
- Szervezet változása: új folyamatok vagy módszerek adaptálása, hogy a termék és a szolgáltatás előállítása gyorsabb legyen.
- Hálózati infrastruktúra változása: növekvő és esetleg földrajzilag szétszórtabban elhelyezkedő munkaerő támogatásának a biztosítása.
- Alkalmazás és rendszer változás: új folyamatok bevezetése vagy meglévő folyamatok módosítása. Ez a változás új adatok gyűjtését, illetve az adatok átstrukturálását is igényelheti.
- Adatok típusának és struktúrájának változása: az adatbázis sémájának módosítása, hogy az új adatok helyet kapjanak.

A változás elkerülhetetlen, szükséges az üzlet túléléséhez és sikerességéhez. De minket az adatbázissal kapcsolatos változások érdekelnek. Természetesen a változással együtt az adatbázis struktúrájának a változása is együtt jár. Sok tényező járulhat hozzá ahhoz, hogy az adatbázis-struktúránkat változtatni kényszerüljünk:

- Alkalmazás programok változása. Ehhez a változáshoz új vagy módosított adatstruktúrák lehetnek szükségesek.
- Teljesítményváltozások, amely eredményeképp azt várjuk, hogy az adatbázis-alkalmazás gyorsabban fusson.
- Szabályozó változások, amelyek hatására új adatstruktúrákra lesz szükség, vagy ugyanazokat az adatokat hosszabb ideig kell tárolni.
- Az üzleti gyakorlat változása, amelyhez új típusú adatokra lesz szükség.
- Technológiai változások, amelyek lehetővé teszik, hogy a korábbi lehetőségekhez képest az adatbázis új típusú adatokat vagy nagyobb mennyiségű adatot tudjon tárolni.

A változás egész életünket, illetve a cégünk egész életét végigkíséri, nem tűnik el soha. Ezért fontos, hogy legyen olyan megoldásunk, amelynek a segítségével ezeknek az elkerülhetetlen változásoknak a kezelésével könnyedén elboldogulunk.

Szükséges követelmények a változás sikeres megvalósításához

Ahhoz, hogy a változást hatékonyan tudjuk kezelni és sikeresen meg tudjuk valósítani, ismernünk kell a következő tényezőket: proaktivitás, intelligencia, analízis (tervezés és kihatás), automatizálás, szabványosítás, megbízhatóság, megjósolhatóság, gyors és hatékony szállítás. Természetesen a változás hatékony kezeléséhez nem elegendő ismerni ezeket a tényezőket, alkalmazni is kell őket tudni. Tekintsük át őket:

Proaktivitás: a proaktív változás, a legértékesebb változástípus, mert a jövőbeli probléma megelőzését szolgálja. A fejlesztési ciklus minél korábbi fázisában azonosítjuk és valósítjuk meg a szükséges változásokat, annál kevesebb lesz a változás teljes költsége.
Intelligencia: Ha egy változás megvalósítására készülünk, akkor a változás minden lehetséges hatását fel kell deríteni, különben a változás a cégnek előre nem várt költségeket eredményezhet. Egy adott terület egyszerűnek vélt változása egy másik területen bonyolult változást okozhat. Vizsgálni kell az egyes változások hatását és ezeket a hatásokat egy változási folyamatban összesíteni kell. A változás kezelésének a folyamatában az intelligencia miatt alapos elemzést kell végeznünk. Az elemzés eredményeként egy megvalósítási terv áll elő, amely alapos munka esetén hatékony, és végrehajtása alacsony kockázattal jár. A valódi intelligencia megköveteli egy kontingencia terv létrehozását is, amelyet abban az esetben használunk, ha egy-egy változás vagy változások halmaza nem a tervezettnek megfelelően megy végbe.
Tervezési elemzés: a tervezés maximalizálja a változások hatékonyságát. Egy jól megtervezett változtatás időt takarít meg. Könnyebb egy jó terv alapján elsőre jól csinálni, mint tenni egy elhibázott próbát, amit ki kell javítani, majd csak utána jól csinálni. Egy hatékony cég a változás minden lehetséges hatását megérti, mielőtt a változás megvalósításához az erőforrásokat lefoglalja.
Kihatás elemzése: az átfogó kihatás- és kockázatelemzés lehetővé teszi a cégnek, hogy a teljes problémát és a benne rejlő kockázatot vizsgálja, így meg tudja határozni a változás megvalósításának a legjobb menetét. Egy változást általában többféleképpen is meg lehet valósítani. Az egyes megvalósítások hatása meglehetősen különböző lehet. Néhány megvalósítás több kockázatot foglal magában: hibák, indokolatlan nehézségek, további változások szükségessége, leállás, stb. Mindezeket a tényezőket számba kell venni, amikor a változás megvalósításához a legjobb közelítést keressük.
Automatizálás: a változási folyamat automatizálása csökkenti az emberi hibát, illetve az monoton feladatok terhét leveszi a túlterhelt személyzet válláról.
Az eljárás szabványosítása: szabványos eljárások használatával akkor is fennmarad a folyamatos termelékenységi szint, ha folyamatosan cserélődnek az alkalmazottak. Ha a cégünk jól szervezett és alaposan dokumentált feladatai vannak, akkor csökken az az idő, amelyet az alkalmazottak az új probléma, és annak megoldásának megértésére fordítanak.
Megbízható és megjósolható folyamat: ha egy terméket hozunk létre, akkor a cégünknek tudnia kell, hogy semmilyen ráfordított erőfeszítés sem veszett el, mivel az idő értékes. A megbízható folyamat teljesen dokumentált, a dokumentáció alapján bármelyik képzett szakember végre tudja hajtani a folyamatot. A folyamat egyes lépései megismételhetőek, azaz az ismétléssel is ugyanazt az eredményt kapjuk. A megjósolható folyamat esetén meg tudjuk mondani, hogy a jövőben végrehajtott folyamat egyes lépéseinek mi lesz az eredménye. A magas szintű megjósolhatóság segíteni fog biztosítani a folyamatos sikert és profitképességet. Megbízhatóság és megjósolhatóság a kulcstényezők az állandóan magas minőségű termék előállításában.
Elérhetőség: a legtöbb változás megvalósításához leállás szükséges. Az alkalmazásoknak, de gyakran az adatbázisoknak is, le kell állniuk. Manapság a magas elérhetőség a legtöbb alkalmazás esetén követelmény. A változás végrehajtása miatt történő leállások számának csökkenésével az alkalmazás elérhetősége nő. Ez nem jelenti azt, hogy a változás nem mehet végbe, mert nem lehet leállítani az adatbázist vagy az alkalmazást. Lehetőségünk van a változás megvalósítására olyan megoldás keresni, amely nem jár együtt leállással, vagy a leállást igénylő változásokat összegyűjteni és egy leállással több változást megvalósítani.
Gyors és hatékony szállítás: az ügyfelek gyors válaszreakciót követelnek a legtöbb terméknél és szolgáltatásnál. A profitképesség akkor a legjobb, ha a termék az első a piacon. Egy termék lassú vagy nem hatékony szállításának költsége nagyon nagy lehet. Ha változást valósítunk meg, a gyorsabb a jobb. Minél rövidebb ideig tart egy változás végrehajtása miatti kiesés, annál gyorsabban lehet a rendszert a piacra dobni.

A változáskezelés az adatbázis-adminisztrátor szemszögéből

Az adatbázis-adminisztrátor felügyeli az adatbázis változásait. Általában nem az adatbázis-adminisztrátor kéri a változást, hanem a programozó, az alkalmazástulajdonos vagy az üzleti felhasználó. Néha az adatbázis-adminisztrátor is kér változást, például teljesítmény okokból vagy egy új technológia vagy új eszköz bevezetésénél. Függetlenül attól, hogy ki kérte a változást, a adatbázis-adminisztrátor feladata az adatbázisban a változtatást elvégezni és biztosítani, hogy a változások sikeresen végbemenjenek, illetve, hogy az adatbázis más részeire ne legyenek hatással.
Az adatbázis-változások hatékony végrehajtásához az adatbázis-adminisztrátornak is tekintetbe kell vennie és alkalmaznia kell az előzőekben felsorolt tényezők: proaktivitás, intelligencia, analízis (tervezés és kihatás), automatizálás, szabványosítás, megbízhatóság, megjósolhatóság, gyors és hatékony szállítás.

A változások típusai

Az adatbázis-adminisztrátor feladatainak nagy része a változások kezeléséből áll. Az üzleti változások általában szükségessé teszik az alkalmazáskód vagy az adatbázis-struktúra változását. A legegyszerűbb üzleti változás is hatással van az adatbázisra, pl. az üzlet nő és ezért több felhasználó fogja használni az adatbázist, vagy újabb típusú adatok tárolására van igény vagy egyszerűen csak a tranzakciók mennyiség nő. Az üzleti változások mellett a technológiai változások is hatással vannak a rendszerünkre, pontosabban az adatbázis-kezelő szoftverre. Ilyen technológiai változások lehetnek az adatbázis-kezelő rendszer frissítései vagy a hardver elemek cseréje. Mind az üzleti, mind a technológiai változások az adatbázis-adminisztrátor feladatait szaporítják.

Adatbázis-kezelő rendszer szoftvere

Ha az adatbázis-kezelő rendszernek új verziója vagy új kiadása kerül a piacra, akkor az adatbázis-adminisztrátornak elő kell készülnie az új verzióra vagy új kiadásra történő migrálásra. A feladat összetettsége nagyban függ az új verzió által támogatott új tulajdonságoktól és funkcióktól, illetve attól is, hogy az új verzióban mi hiányzik, ami még a régiben megvolt. Ha az adatbázis vagy a programok a régi verzióban olyan funkciót használnak, amelyet az új verzió nem tartalmaz, akkor az adatbázist is és a programokat is változtatni kell. Előfordulhat az is, hogy megvan az új verzióban ugyanaz a funkció, de átnevezésre került. Az adatbázis-adminisztrátornak az új adatbázis-kezelő rendszer tulajdonságok megfelelő használatához megfelelő vezérelveket és módszereket kell létrehozni.

Hardver konfiguráció

Az adatbázis-kezelő rendszer igénye miatt szükség lehet hardver frissítésre vagy konfiguráció változásra. Az adatbázis-adminisztrátortól elvárják, hogy a rendszerprogramozóval vagy rendszer-adminisztrátorral együtt konfigurálja és karbantartsa a hardvert. Az adatbázis-kezelő rendszer igényelhet speciális hardverelemeket, hogy a különböző tulajdonságait ki tudja használni. Az adatbázis-adminisztrátor a rendszerprogramozóval együtt találja ki, hogy milyen hardverkonfiguráció szükséges.
A hardver változhat valamilyen adatbázistól független okból is, ami lehet például lemez- vagy memóriaváltozás. Ebben az esetben is előfordulhat, hogy az adatbázis-kezelő rendszernek is változnia kell, ami jelentheti az adatbázis-kezelő rendszer konfigurációjának a változását vagy adatbázis-struktúra változást.

Logikai és fizikai terv

Ha az adatbázis változik fontos, hogy az adatbázisterv is változzon. Ez azt jelenti, hogy a

koncepcionális és a logikai adatmodellt a fizikai adatbázissal szinkronban kell tartanunk. Ezt különbözőképp lehet végrehajtani.
Az adatadminisztrációban jártas cégek azt választanák, hogy először a koncepcionális és logikai szinten végezzük el a változásokat és aztán migráljuk azt a fizikai adatbázisba. Általában egy ilyen megközelítéshez szükség van egy adatmodellező eszközre, amely elkülönítetten kezeli a logikai és fizikai modellt. Ez az adatmodellező eszköz a legtöbb esetben megkönnyíti a különböző szintek változásainak megadását, illetve képes szinkronizálni a különböző modellek között mindkét irányban: a logikaitól a fizikaiig és vissza.
Robusztus adatmodellező eszköz hiányában a modellek szinkronizációja általában kézzel történik. Ha a fizikai adatbázis módosul, az adatbázis-adminisztrátornak kézzel kell a logikai adatmodellt módosítani (és esetleg a koncepcionális adatmodellt is). Az ilyen munka hosszadalmas, nagy pontosságot igényel, ezért unalmas lehet. Minden kézi változtatás hibalehetőséget tartalmaz. A logikai és a fizikai modellnek szinkronizáltnak kell lennie. Ha mégsem az, az adatmodell nem lesz használható az adatbázis-fejlesztéshez.

Alkalmazások

Az alkalmazásváltozásoknak és az adatbázis-változásoknak egymással szinkronban kell lenniük. Ezt általában könnyebb mondani, mint megtenni. Ha a fizikai adatstruktúrában történik a változás, akkor azt az alkalmazásváltozások a legtöbb esetben követniük kell. Például ha egy adatbázistáblához egy oszlopot adunk hozzá, akkor ez megköveteli az alkalmazásszoftver változását úgy, hogy az módosítani, illetve lekérdezni tudja az adatot az új oszlopból.
Ha az adatbázis-változás a termelési környezetbe kerül, az alkalmazásváltozásnak is át kell kerülnie. Ha nem kerül át, az adatbázis-változás hatástalan lesz, rosszabb esetben az alkalmazás működésképtelenné válik. Az adatbázis-adminisztrátor megteheti, hogy először az adatbázis-változást az alkalmazásváltozás nélkül vezeti át a termelési rendszerbe, hogy biztosítsa, hogy az adatbázis-struktúra megfelelően legyen megadva. Miután az adatbázis a termelési környezetben módosult, az adatbázis-adminisztrátor megvizsgálja az adatbázis pontosságát, majd csak ezután kezdheti el az alkalmazás-változás migrálását.
Az adatbázis-változás és az alkalmazásváltozás között szoros kapcsolat van. Ha az alkalmazásváltozást visszavonják, az adatbázis-változást is vissza kell vonni, és fordítva. Ha hibázunk a változások szinkronizálásában, az alkalmazáshibát vagy hatástalanságot eredményezhet. Fontos, hogy az adatbázis-adminisztrátor megértse a kapcsolatot és figyelje a változás folyamatát, így tudja biztosítani, hogy az adatbázis- és alkalmazásváltozások valóban bekövetkeznek egymás után.

Fizikai adatbázis-struktúra

A legtöbb adatbázis idővel változik. Nagyon ritka, hogy egy egyszer megvalósított adatbázis statikus marad. Néhány változás egyszerűen megvalósítható, mások nagyon összetettek, sok hibalehetőséget rejtenek és sok időre van szükség hozzájuk. A legbonyolultabb és a legtöbb időt felemésztő változástípus az adatbázis-adminisztrátor számára a fizikai adatstruktúra változása, amelyet ugyancsak tervezni, elemezni kell, majd csak ezután következhet a változás megvalósítása.

Az adatbázis-struktúra változásának hatása

Ha a cég adatkövetelményei változnak, akkor az adatokat tároló adatbázisnak is változnia kell. Ha az adatok nem megbízhatóak vagy nem érhetőek el, a rendszer nem szolgálja cégünk tevékenységét, sőt inkább fenyegeti a cég üzletének sikerességét.
Ezért az adatbázis változásainak a kezelésére olyan technikákra van szükségünk, amelyeknél a legfontosabb szempont, hogy segítségükkel elkerülhetjük a hibákat. Emellett még legyenek ezek a technikák automatikusak, hatékonyak, és könnyű legyen őket kezelni. Sajnos a mai adatbázis-

rendszerekkel nem könnyű az adatbázis-változásokat kezelni.
Relációs adatbázisokat DDL utasításokkal (CREATE, ALTER, DROP) kezelünk. Egy adatbázis-objektum nem minden részét lehet az ALTER utasítással változtatni. Bizonyos módosításokhoz az adatbázis-objektumot el kell dobni, és új paraméterekkel létrehozni. Az ALTER pontos specifikációja, hogy mit lehet és mit nem lehet megtenni vele, adatbázis-kezelő rendszerről adatbázis-kezelő rendszerre változik.
Például egy létező táblához ALTER utasítással lehet oszlopot adni, de csak a tábla végére. Vagyis az oszlop nem kerülhet két oszlop közé. Ha két oszlop közé szeretnénk tenni az új oszlopot, a táblát el kell dobni és újra létrehozni. Minden adatbázis-kezelő rendszernek megvan a saját korlátozása, hogy az ALTER utasítással mit lehet és mit nem lehet megtenni. Az ALTER utasítás nem csak táblákra, hanem más adatbázis-objektumokra is használható, ezekre ugyanúgy megvannak a korlátozások.
Ha egy adatbázisban egy objektumot el kell dobni és újra létrehozni, az adatbázis-adminisztrátornak számolnia kell a kaszkádolt eldobás hatással. A kaszkádolt eldobás azt jelenti, hogy ha egy magas szintű adatbázis-objektumot eldobunk, akkor vele együtt minden tőle függő objektum is eldobásra kerül. Ha eldobjuk az adatbázist, minden, az adatbázisban definiált objektum is eldobódik. Ha eldobunk egy táblát, az indexei eldobódnak. A kaszkádolt eldobás hatása bonyolítja az adatbázis-változáshoz tartozó feladatokat.
A rendszerkatalógus vagy adatszótár tárolja a metaadatokat az egyes adatbázis-objektumokról. A metaadatok többet tartalmaznak az adatbázis-objektumokról, mint a fizikai jellemzők: biztonsági információkat, adatbázis-statisztikákat is tartalmaznak. Ha az adatbázis-objektumot úgy változtatjuk, hogy eldobjuk majd újra létrehozzuk, akkor az eldobás előtt az adatbázis-objektumhoz minden információ elérhető az adatszótárban, ha tudjuk, hogy hol keressük. Az adatbázis-adminisztrátor feladatának része, hogy tudja, hogy hol keresse ezeket az információkat. Az eldobás és újralétrehozás után viszont mind a biztonsági, mind a statisztikai információ is elvesznek az adatszótárból. Mindkettőt az objektum létrehozása után újra létre kell hozni. A biztonsági információval a jogosultságokat is eldobtuk, ennek az újralétrehozása kihívást jelenthet, ha a biztonság nem megfelelően van dokumentálva. A statisztikai információk alapján a teljesítményt lehet javítani. Ha eldobjuk, akkor a teljesítmény érdekében azokat újra kell generálni.
Ha egy olyan adatbázis-objektumot dobunk el, amelyre egy alkalmazásprogram hivatkozik, akkor az alkalmazás program érvénytelenné válik. Az adatbázis-kezelő rendszertől és a program típusától függően további lépések szükségesek, hogy az alkalmazások ismét érvényesek legyenek az adatbázis-objektumok újralétrehozása után.
Számtalan más feladatai is lehet az adatbázis-adminisztrátornak, az adatbázis-struktúrájának a módosításával és migrálásával kapcsolatban a fizikai változtatásokon kívül. Az egyik nagy kihívás, hogy a tesztadatbázist szinkronban tartsa, hogy az az alkalmazásprogram teszteléséhez elérhető legyen. Az adatbázis-adminisztrátornak robusztus eljárásokat kell fejleszteni, hogy létrehozza az új teszt környezetet a fő teszt struktúrák duplázásával. Az adatbázis-adminisztrátornak szkripteket kell létrehoznia, amelyek a tesztadatbázisban lévő adatokat generálják, mielőtt az egyes tesztek elindulnának. Ha a szkriptek létrejöttek, az alkalmazásfejlesztők annyiszor futtatják, ahányszor szükséges.
A másik kihívás, ha egy nem megfelelően meghatározott adatbázis-változást kell visszaállítania, vagy egy migrációt vissza kell vonni. Ezek a feladatok nagyon bonyolultak. A feladat végrehajtásához ismerni kell az adatbázis-környezet változás vagy migráció előtti és utáni állapotát is.
Mindezek alapján belátható, hogy érdemes egy adatbázisváltozás-kezelő eszközt használni, hogy az adatbázisváltozás-kezelés egyszerűsödjön és automatizálódjon. Léteznek olyan adatbázis-adminisztrátori eszközök a piacon, amelyek a változási folyamatokat kezelik és lehetővé teszik az adatbázis-adminisztrátor számára, hogy egy változást rámutatással és kattintással határozzon meg. Az eszköz ezután az összes részletet kezeli. Egy ilyen eszköz megkönnyíti az adatbázis-

adminisztrátor feladatát abban is, hogy az adatbázis-objektum változása biztosan nem okoz más implicit változásokat. Az adatbázisváltozás-kezelő eszközök a következő tulajdonságokkal rendelkeznek:

- csökkentik az időt, amelyet az adatbázis-adminisztrátor a változáshoz szükségesek feladatok meghatározására fordít
- egyszerűbb és elegánsabb módszert biztosítanak az adatbázis-változás hatásának elemzésére
- kevesebb tudás szükséges az adatbázis-objektumok létrehozásához, módosításához, törléséhez
- minden változás nyomon követhető
- növekszik az alkalmazás elérhetősége azzal, hogy csökken a változás végrehajtásához szükséges idő.

A változáskezelő eszköz az egyik első olyan eszköz, amelyet a legtöbb cég beszerez, amikor adatbázist hoz létre. Egy ilyen eszköz csökkenti az időt, erőfeszítést és az emberi hibát az adatbázis-változások során. Az eszköz használatával járó sebesség- és pontosságnövekedés azonnal visszatérülő beruházást jelent a cég számára.

Az adatbázis-változás forgatókönyve

Egy adatbázis-adminisztrátornak az adatbázis élete során sok különböző típusú változtatást kell végeznie. Néhányat egyszerű és könnyű megvalósítani, másokat sokkal bonyolultabb.
Az ALTER utasítás sok típusú változás elvégzésére alkalmas. Más típusú változásokhoz esetleg további lépések szükségesek. Az adatbázis-adminisztrátor feladata megtalálni a legjobb módszert a különböző típusú adatbázis-változások megvalósítására. Gyakran az egyszerű változásokat bonyolultabb megvalósítani. Például egy egyszerű adatbázis-változás egyáltalán nem egyszerű, ha több különböző szerver több adatbázisán kell végrehajtani.
Egy bonyolultabb változás, mint egy oszlop átnevezése órákat vehet igénybe, ha kézzel valósítjuk meg. Egy oszlop nevének megváltoztatása több száz változást eredményezhet: ütemezni, futtatni, érvényesíteni kell a különböző környezetekben: fejlesztési, tesztelési és a termelési környezetben. Az adatbázis-adminisztrátor feladata megbirkózni az ilyen kihívással.

Néhány adatbázis-változás példa

Adjunk egy új oszlopot egy tábla végére. Általában ez egy nagyon egyszerű változás. A változás megvalósításához a következő ALTER utasítás szükséges:
ALTER TABLE tablanev ADD COLUMN ujoszlop típus;
A változás végrehajtásához egy egyszerű SQL utasítást adtunk ki. A változás végrehajtása egyszerű, nyomon követni viszont annál nehezebb. És minél nagyobb az adatbázis-környezetünk, a nyomkövetés annál nehezebb. Más szóval ha egy egyszerű változtatást 20 különböző adatbázisban kell elvégezni 3 hónap alatt, akkor a feladat bonyolulttá válik. Az adatbázis-adminisztrátornak képesnek kell lennie nyomon követni, hogy melyik környezetben mi változott. És még csak egy változásról beszéltünk, természetesen a 3 hónap alatt sokkal több változásnak is történnie kell az adatbázis-környezetben.
Bonyolultabb változás ha egy táblatér adatlapjain lévő szabad hely mennyiséget szeretnénk megváltoztatni. Ez a változtatás általában egy ALTER utasítással történik, de az ALTER utasítás után további feladatok vannak. Az ALTER utasítás:
ALTER TABLESPACE TS1 PCTFREE 25;
Ez az utasítás a TS1 nevű táblatér adatlapjain a teljes adatlap területének a 25%-a szabad kell, hogy maradjon. Az utasítás hatása csak az utasítás kiadása után létrejövő adatlapokra lesz érvényes. Ha az adatbázis-adminisztrátor a táblatér minden adatlapján ezt el szeretné érni, akkor a teljes táblateret újra kell szerveznie. És van még egy feladata, meg kell bizonyosodnia róla, hogy az így előálló új táblatérhez tartozó állományoknak van-e elegendő hely lefoglalva, és az állományokhoz elegendő

tárterület van-e rendelve a lemezen. Azaz az adatbázis-adminisztrátornak értenie kell, hogy az ALTER utasítás egyes paramétereinek milyen hatása van. Az adatbázis-adminisztrátornak tudnia kell, hogy milyen további feladatai vannak, hogy az adott változást valóban teljesen megvalósítsa.
Az összetett változás megfelelő megvalósítása nagy odafigyelést igényel. A folyamat tele van az emberi hiba lehetőségével és nagyon időigényes. A hatékony adatbázis-változáshoz az adatbázis-adminisztrátornak ismernie kell az adatbázis-környezetbeli bonyolult kapcsolatokat, illetve tudnia kell, hogy a használt adatbázis-kezelő rendszer milyen típusú változásokat támogat.

Az adatbázis-struktúrák összehasonlítása

Ha többszörös adatbázis-környezetet kezelünk, az adatbázis-adminisztrátornak gyakori feladata, hogy az egyik környezetet össze kell hasonlítania a másikkal. Általában a változások az egyik adatbázis-környezetben, a tesztkörnyezetben készülnek. Ha a változásokat sikeresen tesztelték, átvezetik őket a következő környezetbe, például a minőségbiztosítási tesztelésre szolgáló környezetbe. A változás megfelelő migrálásához az adatbázis-adminisztrátornak be kell tudnia azonosítani az összes tesztkörnyezetben létrejött változást.
Az egyik megközelítés szerint az adatbázis-adminisztrátor nyomon követi a változásokat és azokat egy az egyben duplikálja az új adatbázis-környezetbe. Ez a közelítés nem tűnik hatékonynak, időigényes és sok hibalehetőséget rejt.
Egy másik megközelítés szerint egy adatbázis-adminisztrátori eszközt használunk az adatbázis komponenseinek az összehasonlítására. A környezetek közötti összes különbséget az eszköz egy riportba írja, vagy az eszköz automatikusan átmásolja az adatbázis-környezet struktúráját egy másik adatbázis-környezetbe. Ez utóbbi végrehajtásához az eszköz a rendszerkatalógus, az adatszótár vagy a létrehozó DDL szkript használatával összehasonlítja a fizikai adatbázisokat. Összetett adatbázis-megvalósítások esetén egy ilyen összehasonlító eszköz létfontosságú, mert a különböző környezetekben történő változásokat nagyon bonyolult nyomon követni. Minél több környezet létezik, annál bonyolultabb a változáskezelés.
Ha a cégünknek nincs adatbázisváltozás-kezelő eszköze, akkor az adatbázis létrehozásához használt szkripteket mentsük el és tartsuk naprakészen. Az adatbázis minden változását át kell vezetni a DDL szkriptekbe is. Az ALTER utasítások megváltoztatják az adatbázist, de a szkripteket nem. Az adatbázis-adminisztrátornak kell a szkriptekbe az ALTER utasítást beleírni vagy az ALTER utasítás hatásának megfelelően kell a szkripteket módosítani. Egyik megközelítés sem ideális: az első esetén az ALTER utasításon kívül esetleg más műveletek is voltak, akkor azokat is fel kell vinni a szkriptekbe, a második esetén nagy a hibázási lehetőség, mert a változás kétféleképp történik, egyféleképp az adatbázisban, másképp a szkriptekben.
Ha nem tároljuk az adatbázis-objektumok DDL szkriptjeit, akkor az adatszótárbeli vagy rendszerkatalógusbeli információk alapján kell kézzel újralétrehoznunk a DDL utasításokat. Mindkét utóbbi megközelítésre, azaz a DDL elmentésére, illetve a DDL újralétrehozása is igaz, hogy időigényes és sok hibalehetőséget rejt magában.
Ha az adatbázis-adminisztrátornak nincs olyan eszköze, amely a különböző adatbázis-környezeteket össze tudná hasonlítani, akkor nyomon kell követnie minden változást és pontosan kell vezetnie, hogy melyik környezetben milyen változások történtek meg és milyen változások nem. Ez a folyamat is sok hibalehetőséget rejt magában, de ebben az esetben az adatbázis-adminisztrátornak naprakész, kigyűjtött információi vannak az adatbázisstruktúráról. Ha az adatbázis-adminisztrátor nem vezeti pontosan a változásokat, akkor rendszeresen vizsgálnia kell az adatbázisstruktúrákat, hogy mi változhatott az egyes adatbázis-környezetekben. Ez is időigényes folyamat, sok hibalehetőséget rejt, és nagyon unalmas tud lenni.

Adatbázis-változások kérése

Az adatbázis elsődlegesen az üzleti felhasználókért van, akik alkalmazásokon keresztül érik el az

adatbázist. A változás az ő igényük alapján történik. A változáskérést az alkalmazásfejlesztő csapat kapja meg. Az alkalmazásfejlesztők ezt az igényt feldolgozzák, majd az alkalmazás változásához szükséges adatbázis-változásokat továbbítják az adatbázis-adminisztrátor felé.
Az adatbázis-adminisztrátor csoportnak meg kell vizsgálnia a kérést vagy kéréseket, hogy a kért változás milyen hatással lesz az adatbázisra és az adatbázis-alkalmazásokra. Ezen információ kiértékelése után lehet csak az adatbázis-változásokat megvalósítani.
Egy alkalmazásfejlesztői kérés csak akkor történhet meg, ha az valóban megvalósítható. Előfordulhat, hogy a kérést abban a formájában nem lehet megvalósítani. Ekkor az alkalmazásfejlesztőnek és az adatbázis-adminisztrátornak meg kell vitatni a kérést, és átalakítani úgy, hogy a felhasználói igényeknek is megfeleljen és az adatbázisban is megvalósítható legyen.

A változáskérések szabványosítása

Az adatbázis-adminisztrátor csoportnak kell megszerveznie a vezérelveket és a módszereket arra, hogy a változások hogyan kérhetőek és valósíthatóak meg.
Az adatbázis-adminisztrátoroknak ki kell fejlesztenie egy olyan rendszert, amellyel a változáskéréseket egyszerűen kezelhető, dokumentált formában lehet kérni, majd a teljesítést visszaigazolni. A kérések létrehozásához szabványosított űrlapot kell létrehozni, amely megfelel a cég feltételeinek: mint a környezet, fejlesztési elvárások, tudás, adatbázis-adminisztrátor tapasztalat, termelési munkaterhelés, szolgáltatási szintről szóló megállapodások, platformok, adatbázis-kezelő rendszerek és névkonvenciók. Az űrlapoknak a változáshoz kapcsolódó minden információt tartalmaznia kell, de legalább az operációs rendszer, az adatbázis-alrendszer vagy a példány nevét, az objektum tulajdonosát, az objektum típusát, a kért változtatást és a kért adatokat. A kérést a megvalósítás előtt az illetékes személyeknek, mint az alkalmazásfejlesztő csoport vezetője, szenior adatbázis-adminisztrátor, adatadminisztrátor, rendszer-adminisztrátor, stb. támogatnia kell. Ezt a támogatást a rendszerben ugyancsak rögzíteni kell.
Ha az adatbázis-változás megtörtént, a rendszerben ezt a változást megvalósító adatbázis-adminisztrátor rögzíti, amelyet a kezdeményező azonnal láthat, vagy értesítést kaphat róla. A kezdeményező ezután kérheti a változást a termelési környezetben is.
A szabványosított változás kérési rendszerrel megelőzhetjük a félre kommunikációt amely a változáskezelési folyamat alatt történhet.

Az ellenőrző listák

Sok adatbázis-adminisztrátornak van ellenőrző listája, amellyel követik az egyes adatbázis-változásokat. Ez az ellenőrző lista esetleg beleolvadhat a változáskérési rendszerbe.

Kommunikáció

Szükséges, hogy a változást kezelő rendszert a változást kérők is ismerjék, illetve megfelelő módon használják. Az adatbázis-adminisztrátor feladata, hogy megtanítsa a fejlesztőket a rendszer használatára. Ahhoz, hogy a rendszert megfelelően használni lehessen, el kell készíteni a szükséges dokumentációt, amely tartalmazza, hogyan lehet változást kérni, mit hogyan kell kitölteni, illetve leírást ad az egyes változás kérési szolgáltatásokhoz is. (Például kb. mennyi időt vehet igénybe.)
A leggyakoribb probléma a változáskérések esetén az idő. A változást kérő elvárja, hogy az adatbázis-adminisztrátor azonnal végezze el a változáshoz szükséges teendőket, nem számolva azzal, hogy arra valójában mennyi időt kell szánni, vagy hogy az adatbázis-adminisztrátornak más feladatai is vannak. Ezt el lehet kerülni, ha az adatbázis-adminisztrátor tisztázza, hogy az egyes szolgáltatásokat mennyi idő alatt tud elvégezni. Az adatbázis-adminisztrátor ehhez figyelembe veszi a változás elvégzéséhez szükséges időt, az elérhetőséget, a teljesítménykövetelményeket, és azt is, hogy a saját munkájába ezt a feladatot hogyan tudja beütemezni. Ha az egyes szolgáltatásokhoz meghatároztuk, és dokumentáltuk, hogy mennyi időt vesz igénybe, akkor a változást kérő ezzel tud

számolni és a munkájába beépíti ezt a várakozási időt.
A változások kezelésénél igen fontos a változások egymásra hatásának, a függőségeknek a dokumentálása. Ezáltal egy változás esetén azonnal láthatjuk és ellenőrizhetjük annak a hatását. Azt is meg tudjuk adni, hogy milyen sorrendben kell a változásokat végrehajtani.

Összefoglalás

A fejezetben megnéztük, hogy a cég életében mivel jár a változás. Példákat soroltunk fel a változásra, majd megnéztük, hogy az adatbázis-adminisztrátornak az adatbázissal kapcsolatban milyen változásokkal kell számolnia. Felsoroltuk a változás sikeres megvalósításához szükséges alapvető követelményeket, és megnéztük ezeknek a jelentését: proaktivitás, intelligencia, tervezési elemzés, kihatás elemzés, automatizálás, az eljárás szabványosítása, megbízható és megjósolható folyamat, elérhetőség, gyors és hatékony szállítás. Az adatbázis szempontjából a legfontosabb változási típusokat részleteztük: az adatbázis-kezelő rendszer szoftverének változása, a hardver konfiguráció változása, a logikai és a fizikai terv változása, az alkalmazások változása, és a fizikai adatbázis-struktúra változása. Ezek közül részleteztük az adatbázis-adminisztrátor számára a leggyakoribb és legösszetettebb feladatot, az adatbázis-struktúra változásának a hatását. Erre adtunk példát, és megnéztük, hogyan lehet a különböző környezetek struktúráit összehasonlítani. Végül áttekintettük, hogy egy cégnél hogyan lehet az adatbázis-változási kéréseket hatékonyan, dokumentált formában feldolgozni, és a kéréseket megvalósítani, majd visszaigazolni.

Ellenőrző kérdések

1. Soroljunk fel példákat egy cégnél történő lehetséges változásokra!
2. Milyen okok miatt kell változtatni az adatbázisstruktúrát?
3. Melyek a szükséges követelmények a változás sikeres megvalósításához? Pontosan mit jelentenek ezek?
4. Adatbázis-adminisztrátori szemszögből milyen változástípusokra kell számítanunk?
5. Milyen következményei vannak annak, ha egy adatbázis-objektumot eldobunk? Mit jelent a kaszkádolt eldobás?
6. Milyen eszközök segítik az adatbázis-adminisztrátor munkáját a változások kezelésében?
7. Milyen lehetőségeink vannak a különböző adatbázis-környezetek összehasonlítására?
8. Hogyan érdemes egy cégnél a változáskéréseket kezelni?

Irodalomjegyzék

1. Craig S. Mullins: Database Administration, The Complete Guide to Practices and Procedures, Addison-Wesley, 2002
2. Ramez Elmasri – Shamkant B. Navathe: Fundamentals of Database Systems, Addison-Wesley, 2011
3. Hector Garcia-Molina – Jeffrey D. Ullman – Jenifer Widom: Adatbázis-rendszerek megvalósítása, Panem, 2001
4. Dr. Barna István: Osztott adatbázisok, Számítástechnika-alkalmazási Vállalat, 1984
5. Kevin Loney: Oracle Database 10g, Teljes Referencia, Panem, 2004
6. Oracle 11g Documentation: http://www.oracle.com/pls/db112/homepage
7. IBM DB2 9.7 Documentation: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp
8. Microsoft SQL Server 2008 R2 Documentation: http://msdn.microsoft.com/en-us/library/ms130214.aspx