Ez a jegyzet 150 oldalas hivatalos jegyzet ellenőrző kérdéseinek a válaszát tartalmazza. Ha valamelyik kérdésre, a hivatalos vagy órai jegyzet alapján jobb választ tudnál, vagy hibásnak véled a válaszd, akkor javítsd és jegyezd be a verziókezelőbe.

http://dbareactions.com/ dba 4ever 頑張って下さい。

Leírás	Verzió
Az összes ellenőrző kérdésre való válasz kikeresése a jegyzetből.	1.0

DBA

(Dumb Big Assbite**)**

1. fejezet
Az adatbázis-adminisztrátor

1. **Mi az az adatbázis?**

Az adatbázis egy szervezett adattár, amelyben az adatokat adatelemekként (például mezőként, rekordokként, és állományokként) érhetőek el.

2. **Mi az az adatbázis-kezelő rendszer?**

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.

3. **Ki az az adatbázis-adminisztrátor?**

Az adatbázis-adminisztrátor (Database Administrator, DBA) felelős egy cég adatbázisainak tervezéséért és karbantartásáért, azaz a cég adatbázisainak folyamatban lévő működésének biztosításáért és hatékonyságért, és azért, hogy az alkalmazások elérjék az adatbázisokat.

4. **Mit jelent a reaktív közelítés az adatbázis-adminisztrációban?**

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.

5. **Mit jelent a proaktív közelítés az adatbázis-adminisztrációban?**

A proaktív adatbázis-adminisztrátor a lehetséges problémák megelőzésén dolgozik. Tehát még a problémák bekövetkezte előtt látja, hogy mi okozhat majd problémákat és olyan megelőző lépéseket tesz, amelyek

segítségével elkerüli a problémákat. Nem lehet minden lehetséges problémát proaktív módon kezelni, lesznek tűzoltás jellegű feladatok. pl.: váratlan leállás, váratlan újraindítás, futási idejű kivételek kezelése.

6. Milyen feladatai vannak az adatbázis-adminisztrátornak az alkalmazásfejlesztés életciklusának a fázisai alatt?

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 tervezési fázis alatt az adatbázis-adminisztrátor a koncepcionális adatmodellt egy logikai adatmodellbe transzformálja. A fejlesztés megkezdése előtt a logikai adatmodellből fizikai adatbázistervet készít az adatbázis-adminisztrátor. Az adatbázis-adminisztrátornak az alkalmazás teszteléséhez példaadatokat kell a fizikai adatbázisba felvennie. Ha az alkalmazás fejlesztése a működési státuszba kerül, az adatbázis-adminisztrátor előkészíti az adatbázis-kezelő rendszert a munkaterhelésre. 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. 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.

7. Mi a koncepcionális adatmodell?

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 elérhető, és fel lehet használni a felhasználókkal való kommunikációban. Segítségével az adatbázis-adminisztrátor ellenőrizheti, hogy minden felhasználói követelményt figyelembe vett-e, illetve felfedezheti az esetlegesen konfliktusban álló adatkövetelményeket. A modell az 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.

8. Mi a logikai adatmodell?

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. A logikai modell létrehozása abból áll, hogy a koncepcionális adatmodellt az adatbázis-adminisztrátor a cég adatbázis-kezelő rendszere által használt modellbe transzformálja.

9. Mi a fizikai adatbázisterv?

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.

10. **Mi a feladata az adatadminisztrátornak? Miben különbözik az adatbázis-adminisztrátortól?**

Az adatok üzleti szempontjai tartoznak az adatadminisztrátorhoz, a technikai oldalt az adatbázis-adminisztrátor kezeli. Az adatadminisztráció az adat, mint erőforrás kezelésének üzleti oldalát elválasztja az adatokat kezelő technológiától. 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. 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 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. Az adatadminisztráció egy üzletorientált szakterület, amely a cég adatvagyonáért felelős. 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.

11. **Mi a feladata a rendszer-adminisztrátornak? Miben különbözik az adatbázis-adminisztrátortól?**

A rendszer-adminisztrátor felelős az adatbázis-kezelő rendszer telepítéséért és frissítéséért és 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. 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 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.

12. **Soroljuk fel röviden az adatbázis-adminisztrátor feladatait!**

– adatbázisterv(meg kell tanulnia relációs tervezést és ragaszkodnia kell a bevált gyakorlathoz, egy logikai adatmodellt fizikai adatbázisba leképezni)
– Teljesítménymonitorozás és hangolás,
– Elérhetőség,
– Adatbázis-biztonság és feljogosítás,
– Mentés és helyreállítás,
– Adatbázis-kezelő rendszer verziómigrációja.

13. **Mit jelent a fizikai integritás? Mit jelent a szemantikai integritás? Mit jelent a belső integritás?**

A *fizikai integritást* 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. Hivatkozási megszorítás, amelyet arra használ az adatbázis-adminisztrátor, hogy megadja azokat az oszlopokat, amelyekkel a táblák közötti kapcsolatot definiálja. 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.
A *szemantikus integritásra* példa az adatok minősége az adatbázisban. Az adatminőséget megfelelő alkalmazáskódokkal, üzleti fogásokkal, és különleges adatpolitikával, illetve adattisztítással és jól definiált folyamatokkal lehet támogatni. pl.: rossz cím vagy telefonszám, redundáns tárolás(szinkronizáltan tartás)
A *belső integritást* az adatbázis-kezelő rendszer a belső struktúrákban és kódokban lévő hivatkozások és mutatók konzisztenciájának biztosításával tartja fönn.
Az adatbázis-kezelő rendszerbeli belső integritás létfontosságú a következő területeken:
– *Indexkonzisztencia*: egy index igazából nem más, mint adatbázistáblák adataira mutató mutatók rendezett listája. 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.
– *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.

14. **Milyen IT infrastruktúrabeli komponensekkel áll kapcsolatban az adatbázis? Soroljuk fel őket!**

– 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

15. **Mitől függ az, hogy egy cégnek hány adatbázis-adminisztrátorra van szükséges?**

– 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.
– 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.
– 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.

16. **Mik a különbségek és a hasonlóságok a teszt, a fejlesztői és a termelési rendszerek között?**

Az új fejlesztés és a karbantartási munkálatok először mindig a teszt környezetben valósulnak meg. A működő alkalmazások pedig a termelési környezetben futnak. 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. 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.

2. fejezet
Az adatbázis-környezet létrehozása

1. Milyen szempontok alapján választunk adatbázis-kezelő rendszert?

– 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.

2. Soroljuk fel a napjainkban ismert nagy adatbázis-kezelő rendszert szállító cégeket és a termékeiket!

IBM: DB2
Microsoft: Microsoft SQL Server
Oracle: SQL Server

Nyílt kódú szoftverek: PostgreSQL, MySQL, SQLLite

3. **Milyen adatbáziskezelőrendszer-architektúrák vannak, és melyiknek mi a jellemzője?**

Vállalati (enterprise): 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 nagy-gépeken fut, több processzort, párhuzamos lekérdezéseket támogat, stb.
személyi (personal): A személyi adatbázis-kezelő rendszer egyfelhasználós, általában PC-n fut. Ilyen például a Microsoft Access. Ezek általában sokkal kevesebbe kerülnek, mint a vállalati adatbázis-kezelő rendszerek, nem skálázhatóak, és nem alkalmasak többfelhasználós alkalmazásokhoz.
*mobil:*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.

4. **Mit jelent a klaszterezés (fürtözés)? Milyen két domináns architektúrát említettünk? Melyiknek mi a jellemzője?**

A klaszterezés több független számítógép összekapcsolása, melyeknek az együtt végzett munkája a felhasználó számára úgy tűnik, mintha egy önálló, magas elérhetőségű rendszert használnánk. Két domináns architektúra létezik: a megosztás nélküli (shared-nothing) és a megosztott-lemez (shared-disk) 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. A fő előnye a skálázhatóság. Akár ezer processzorig skálázható a rendszer, mivel nem akadályozzák egymást.
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.

5. **Az adatbázis-adminisztrátornak milyen feladatai vannak az adatbázis-kezelő rendszer telepítése előtt és után? / Telepítéskor a DBMS telepítése mellett még milyen feladatai vannak a DBA-nak?**

Az adatbázis-adminisztrátornak ismernie kell a telepített adatbázis-kezelő rendszernek a követelményeit és elő kell készítene a környezetet az új adatbázis-kezelő rendszer fogadására. 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 és telepítéséhez mennyi tárterület szükséges. Tekintsük át, hogyan működik a telepítőprogram és kövessük az explicit utasításokat, amelyeket a telepítési útmutató ad.

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?**

– 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.
– Naplóállományok, amelyek az adatbázisban végrehajtott változásokat tartalmazzák. Ez aktív és archív naplókat, visszagörgető szegmenseket, és más naplókat tartalmaz.
– 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.
– 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.

7. **Miért olyan fontos az adatbázis-kezelő rendszer számára a memória?**

Jelentős mennyiségű memóriára van szüksége a gyorsítótárhoz (cache), hogy elkerülje a gyakori input-output műveleteket (I/O-t). Az adatok mellett az adatbázis-kezelő rendszer más struktúrákat is beolvas a memóriába, mint például programstruktúrákat. 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. Ha az adatbázis-kezelő rendszernek az elegendőnél több memóriát biztosítunk, akkor már sokat tettünk az adatbázis- folyamatok optimalizálásáért és a lehetséges teljesítményproblémák minimalizálásáért.

8. **Mik a rendszerparaméterek? Mire szolgálnak?**

Rendszerparaméterek konfigurálásával válnak az adatbázis-kezelő rendszer funkciói és erőforrásai elérhetővé a felhasználók számára. 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.

9. **Hogyan lehet az adatbázis-kezelő rendszer telepítését felülvizsgálni?**

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.

10. **Az alkalmazásoknak milyen szoftverrétegei vannak?**

– 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.

11. **Milyen előnyei és veszélyei vannak az adatbázis-kezelő rendszer frissítésének? Hogyan kell elvégezni a frissítést?**

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 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.

3. fejezet
Metaadatok kezelése

1. **Mi a metaadat?**

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.

2. **Mikor válik az adat információvá?**

Az információhoz szükség van a környezetre, amely meghatározza az adatok közötti kapcsolatot és más információkat. Az adat, ha metaadat környezetbe kerül, információvá válik.
pl.: adat: 27 -> a 27-es szám tízes vagy 8 számrendszerbeli?

3. **Mit tartalmaz a metaadat-stratégia?**

– 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

4. **Mi a különbség a technológiai és az üzleti metaadat között?**

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. Üzleti metaadat például az, hogy a SZEMELYISZAM valójában egy személyi számot takar, hogy egy személyi szám csak egy adott személyhez tartozik.

5. **Mit tartalmaz a rendszerkatalógus?**

– 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

6. **Soroljunk fel olyan metaadatokat, melyek nem találhatóak meg a rendszerkatalógusban!**

– 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

7. **Milyen tulajdonságai vannak a rendszerkatalógusnak? Melyik mit jelent?**

– *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.

8. **Mi a repository?**

A repository napjaink informatikai megnevezése szerint általános tárolót jelent. A jegyzetünk szerint a repository-ban az adatbázis-környezethez tartozó metaadatok kerülnek tárolásra.

9. **Általában mit tud a repository? Mit várhatunk még ezen felül el tőle?**

– 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.

10. **Milyen előnyei vannak a repository-nak?**

A repository-ban lévő metaadatok egyesített információkat tartalmaznak a cég különböző rendszereiről. Az egyesítés segítségével olyan kapcsolatokat lehet felfedezni az adatok között, amelyek még esetleg nem ismertek a cégen belül. 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. A repository megkönnyíti az alkalmazáskomponensek újrafelhasználását.

11. **Milyen nehézségekkel kell szembenézni a repository-val kapcsolatban?**

Az egyik legnagyobb nehézség 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.

12. Milyen forrásokból származnak a repository-k metaadatai?

– 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

4. fejezet
Adat- és tároláskezelés

1. Milyen alapvető tényezőket kell figyelembe venni a megfelelő tárolási technológia kiválasztásához?

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.

2. Milyen modern tárolási megoldásokról hallottunk?

SAN (Storage Area Network): A SAN egy speciális hálózat tárolási eszközök összekapcsolására. Segítségével a szerverek ezt a háttértárhálózatot egy normál háttértárolónak látják, és a megszokott módon tudják használni.
NAS (Network-attached Storage): 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.)

3. A tárolási rendszer kialakításakor milyen célokat érdemes szem előtt tartani?

– 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.

4. Mi az a táblatér?

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 táblatér az adatbázisban ugyancsak egy adatbázis-objektumként jelenik meg.

5. Mi az az adatlap? Hogyan épül fel?

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.

– *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.

6. Mi az adatrekord? Hogyan épül fel?

A tábla egy-egy sorát az adatbázis-kezelő rendszer adatrekordba szervezi. Ahhoz, hogy tudjuk, hogy az adott adatrekord melyik táblának az adatait vagy milyen felépítésű rekordot tárol, szükség van a rekord szerkezetét leíró információkra is.

– *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.

7. Egy adatlapra hány darab adatrekordot lehet elhelyezni?

Ez függ az adatrekord méretétől. Ha az adatlap mérete nagyobb, mint az adatrekord mérete akkor tegyük fel, hogy az adatlap mérete L bájt, az adatrekord mérete R bájt. Vegyük L/R egész részét, ami megadja, hogy egy adatlapra mennyi adatrekord kerülhet. Természetesen ekkor marad az adatlapon kihasználatlan terület. Ez nagy problémát nem okoz. 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.

8. Mi az az index? Miért hasznos?

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.

9. Hogyan épül fel az indexrekord?

– 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.

10. Mi az a kiterjesztés? Mi a másodlagos kiterjesztés?

A több adatlaphoz tartozó, a lemezen folyamatosan elhelyezkedő blokkokat nevezzük együtt kiterjesztésnek (extent). Később, ahogy az adatbázis-objektum nő, egyre több tárterületre van szüksége, ezért újabb és újabb kiterjesztéseket foglal le ezeket az újonnan lefoglalt kiterjesztéseket nevezzük másodlagos kiterjesztéseknek.

11. Mit jelent az 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.

12. Táblamódosítások esetén hogyan változik a lemezen lévő adatlapok tartalma?

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.
Egy *táblabeli sor törlése* esetén, 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. 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.

13. Hogyan lehet kiszámolni, hogy egy táblának mennyi tárterületre van szüksége?

A rekord hossza tartalmazza a fejlécet és az offset táblát is. A sor adatrészhosszának(a rekord része) a kiszámításához az adatbázis-adminisztrátornak dokumentációra van szüksége, amely leírja az adatbázis-kezelő rendszer által támogatott adattípusok aktuális fizikai hosszának a meghatározását. A rekordhossz kiszámítása után a következő lépés meghatározni, hogy mennyi sor fér el egy adatlapra. 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.

14. Hogyan lehet kiszámolni, hogy egy indexnek mennyi tárterületre van szüksége?

Egy index tárolásához szükséges hely kiszámításának első lépése az, hogy kiszámítjuk az index sorméretét. 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.

15. **Mi az a tranzakciónapló vagy adatbázisnapló?**

Az adatbázis egyik legfontosabb állománya amelyben az adatbázisban végrehajtott tranzakciók bejegyzései vannak. pl.: COMMIT, INSERT, UPDATE, DELETE

16. **Mik a tranzakciók ACID tulajdonságai?**

– Atomosság (atomicity): a tranzakció „mindent vagy semmit” jellegű végrehajtása (vagy teljesen végrehajtjuk, vagy egyáltalán nem hajtjuk végre).
– Konzisztenciamegőrzés (consistency 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 (isolation): az a tény, hogy minden tranzakciónak látszólag úgy kell lefutnia, mintha ez alatt az idő alatt semmilyen másik tranzakciót sem hajtanánk végre.
– Tartósság (durability): az a feltétel, hogy ha egyszer egy tranzakció befejeződött, akkor már soha többé nem veszhet el a tranzakciónak az adatbázison kifejtett hatása.

17. **Milyen naplóbejegyzések vannak az adatbázis-naplóban?**
– 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

18. **Mennyi állomány tartozik az adatbázisnaplóhoz?**

Az adatbázis-adminisztrátornak 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.

19. **Mit jelent a többszörös redundáns adatbázis-naplózás?**

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.

20. **Mi az ellenőrzőpont?**

Az adatbázisnaplóban található ellenőrzőpont-bejegyzés alapján az adatbázis-kezelő rendszer biztosan tudhatja, hogy az adatbázisnaplóban az az ellenőrzőpont előtti naplóbejegyzésekre már nincs szükség, mert azok már bekerült az adatbázisba is. Ilyenkor az adatbázisnaplót nem szükséges az ellenőrzőpont-bejegyzés elé visszagörgetni.

21. **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. 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.

22. **Mi alapján választja meg az adatbázis-adminisztrátor a naplóállományok méretét?**

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.

23. **Miért szükséges több háttértárat használni egy adatbázis tárolásához?**

– 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.

24. **Milyen szempontokat vegyünk figyelembe, amikor az állományokat különböző lemezekre helyezzük el?**

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. A tranzakciónapló a táblák adatait tartalmazó állományoktól mindig elkülönített eszközön legyen, és hogy a tranzakciónapló mentése független legyen az adatbázis mentésétől. Az SMS-sel (system-managed storage) az adatbázis-adminisztrátor helyett a rendszer határozza meg az állományok aktuális helyét.

25. **Mi az a raw partíció? Milyen előnyei és hátrányai vannak?**

A raw partíció egy egyszerű lemezeszköz, amelyen nincs operációs rendszer vagy állományrendszer telepítve. 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. A raw eszköz hátránya, hogy nehéz nyomon követni az adatbázis-állományokat.

26. **A tömörítésnek milyen előnyei és milyen hátrányai vannak?**

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.

27. **A jövőbeli adatbázis-növekedéssel járó tárigény kezelésére az adatbázis-adminisztrátor hogyan tud felkészülni?**

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.

5. fejezet
Adatok mozgatása

1. **Mire szolgál a LOAD segédprogram?**

A LOAD segédprogrammal nagy mennyiségű adathalmazt vihetünk be egy adatbázis táblájába.

2. **Mi történik, ha olyan táblába töltünk be adatokat, amely nem üres?**

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.

3. **Az adatbázis-adminisztrátornak milyen tényezőket kell figyelembe vennie a betöltés megvalósításakor? / A LOAD segédprogram használatánál milyen tényezőkre kell ügyelni?**

A betöltés újraindíthatósága: A betöltés külső, a betöltő folyamaton kívüli hiba esetén leállhat. Ekkor a betöltést újra kell indítani. 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.
Triggerek: A legtöbb adatbázisban a LOAD segédprogram nem indítja el a triggereket, ami adatintegritási problémákhoz vezethet. 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. Ekkor valamilyen más módon kell a megszorításoknak nem megfelelő adatokat kiszűrni, és megfelelően kezelni.
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.

4. **Milyen állományokra van szükség a betöltéshez?**

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.

5. **Mit tartalmaz az input állomány leírása?**

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 táblákba töltendő lebegőpontos adatok formátumáról a LOAD segédprogram általában valamilyen speciális információt vár.

6. **Hogyan lehet hatékonyabbá tenni a betöltést?**

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. 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. A LOAD segédprogram opciót biztosíthat a naplózás kikapcsolására. Ha a naplózást kikapcsoljuk, a betöltési folyamat felgyorsul, és minimalizáljuk a napló túlcsordulását. Általában a LOAD segédprogrammal sokkal hatékonyabb lehet egy tábla kezdő adatait betölteni, mint többszörös SQL INSERT paranccsal vagy alkalmazói programmal végrehajtani ugyanazt a feladatot.

7. **Lehet-e más alkalmazásokat futtatni a betöltés alatt?**
Az adatbázis-adminisztrátor már a betöltés folyamata alatt képmásolati mentést vagy adatbázis-statisztikát készíthet az aktuálisan betöltött adatokról.

8. **Mire szolgál az UNLOAD segédprogram?**

Az UNLOAD segédprogram célja, hogy adatokat olvasson ki az adatbázisból és a kiolvasott adatokat egy output adatállományba írja.

9. **Az adatkimentéssel párhuzamosan végezhető-e valamilyen tevékenység azon a táblán, amelyből adatot mentünk ki? Ha igen, milyen? Ha nem, miért nem?**

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.

10. **Mi az előnye és a hátránya annak, ha a képmásolati mentéseket arra használjuk, hogy például egy tábla adatait egy másik állományba mentsük?**

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. 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.

11. **Mit jelentenek adatkimentésnél a LIMIT, SAMPLE, WHEN opciók?**

A LIMIT paraméter arra szolgál, hogy korlátozza az UNLOAD segédprogram által kimentendő sorok számát. 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. A WHEN záradék azt határozza meg, hogy csak bizonyos, a WHEN kulcsszó után szereplő feltételnek megfelelő adatok szerepeljenek a kimentett adatok között. + pl.:

12. **Tesztelésnél hogyan lehet használni a LOAD és az UNLOAD segédprogramot?**

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.

13. **Mi az az 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 szoftvert be lehet úgy állítani, hogy felismerje és kinyerje az adatokat különböző forrásokból. 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 szoftver könnyedén automatizálja ezeket a transzformációkat.

14. **Mit jelent a replikáció és a 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 propagáció csak a megváltozott adatokat migrálja. A propagációt meg lehet valósítani úgy, hogy a tranzakciós naplót vizsgáljuk és az adatváltozásokat alkalmazzuk a másik adattárban.

15. **Mi az az üzenetküldő szoftver?**

Az üzenetküldő szoftverek 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 szoftvereknek az adatmozgatást tekintve abban van a legnagyobb előnye, hogy egyszerű, heterogén, bármihez hozzácsatlakoztatható kapcsolatokat hozhatunk létre velük.

6. fejezet
Adatok elosztása

1. **Mi az elosztott adatbázis?**

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.

2. **Van-e kitüntetett csomópont az elosztott adatbázisban?**

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.

3. **Összesen hány adatbázis-kezelő rendszert kell telepíteni egy elosztott adatbázis megvalósításhoz?**

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.

4. **Egy elosztott adatbázisban csak egyforma típusú adatbázis-kezelő rendszerek kapcsolódhatnak?**

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.

5. **Elosztott adatbázisban az adatok redundánsan vannak tárolva?**

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.

6. **Milyen jellemzőket használnak elosztott adatbázisok esetén?**

– *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.

7. **Milyen tényezőket kell figyelembe venni az elosztott adatbázisok tervezése és létrehozása esetén?**

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 az is fontos, hogy az egyes csomópontok mennyire lesznek függetlenek.

8. **Mitől függ az, hogy melyik csomópontra helyezzük el az adatokat?**

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.

9. **Elosztott adatbázisra fejlesztett alkalmazások esetén a fejlesztőket milyen információkkal segíti az adatbázis-adminisztrátor?**

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.

10. **Elosztott rendszerekkel kapcsolatosan milyen szabványokról hallottunk?**

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.
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. 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 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.

11. **A távoli oldalon lévő adatok lekérdezésére és módosítására milyen elérési módok léteznek? Mi a különbség közöttük?**

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.

12. Mi a kétfázisú COMMIT? Mire szolgál? Hogyan megy végbe?

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 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.

13. Elosztott rendszerek esetén milyen extra teljesítményproblémával kell az adatbázis-adminisztrátornak szembe néznie?

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. Elosztott rendszerekben a kérést küldő és a kérést kiszolgáló csomópontok teljesítményét külön-külön is vizsgálni kell.

14. Mi az az átereszt ő lánc? Milyen komponensek vesznek benne részt?

Egy elosztott adatbáziskérés áteresztőképességének analizálásához, az adatbázis-adminisztrátornak az egész kérést teljesítő áteresztő láncot meg kell vizsgálnia. 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. Ha csak egy komponens teljesítménye nem megfelelő, akkor az teljesítménybeli romláshoz vezethet. 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.

7. fejezet
Adatbázis-biztonság

1. **Ha a céghez új fejlesztő érkezik, akkor az adatbázis-adminisztrátornak milyen biztonsági feladatai vannak az új alkalmazottal?**

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.

2. **Az adatbázis-kezelő rendszerben milyen információkat szükséges/lehet egy felhasználói névhez rendelni?**

– 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.

3. **Mit jelent az, ha az adatbázis-kezelő rendszerben egy felhasználói nevet zárolunk? Miért jó ez?**

A felhasználói név zárolása megakadályozza, hogy a felhasználó elérje az adatbázis-kezelő rendszert, azonban a felhasználói nevet nem dobja el a rendszerből.
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 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.

4. **Miért szükséges bizonyos időközönként megváltoztatni a felhasználónak a jelszavát? Az adatbázis-adminisztrátor milyen adatbázis-kezelő rendszerbeli eszközöket használhat a jelszóváltoztatás kikényszerítésére?**

A jelszót szabályos időközönként meg kell változtatni, hogy a betörőknek nehezebb legyen elérni az adatbázist. Azokat a felhasználókat, akik nem változtatják meg a jelszavukat, az adatbázis-adminisztrátor letilthatja vagy zárolhatja a felhasználói nevet.

5. **Milyen utasítással lehet jogosultságokat adni, illetve visszavonni?**

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

6. **Mit jelent a WITH GRANT OPTION záradék?**

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.

7. **Hogyan csoportosítottuk a jogokat? Melyik csoport milyen jogokat tartalmaz? Hozzunk példát az egyes típusokra!**

A *táblajogok* felügyelik, hogy a felhasználók mely táblákat, nézeteket és táblabeli oszlopokat érhetnek el, illetve módosíthatnak.
– 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
A következő utasítás lehetővé teszi, hogy a FELHASZNALO7 nevű felhasználó a CIMEK táblából töröljön: GRANT DELETE ON CIMEK TO FELHASZNALO7;

Az *adatbázisobjektum-jogok* felügyelik, hogy mely felhasználóknak van joguk létrehozni adatbázis-objektumokat, mint például indexeket, táblákat, nézeteket, tárolt eljárásokat, triggereket. 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;

Tárolt-eljárás jogok vagy programjogok: Felügyeli, hogy ki futtathatja a tárolt függvényt, vagy a tárolt eljárást, illetve az adatbázisbeli programokat. 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 *rendszerjogok felügyelik*, hogy a felhasználók melyik adatbázis-kezelő rendszerbeli eszközöket használhatják és milyen adatbázis-kezelő rendszerbeli parancsokat futtathatnak. A legtöbb adatbázis-kezelő rendszer 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;

8. **Ki az a PUBLIC?**

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.

9. **Mit jelent a kaszkádolt jogosultság 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.

10. **Mik azok a biztonsági riportok? Miért hasznosak?**

Az adatbázis-adminisztrátornak monitorozni és riportolni kell a felhasználók jogait. 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. Legyünk biztosak, hogy a rendszerkatalógus megfelelő védelmet kap a termelési rendszerben.

11. **Mik azok a szerepkörök? Miért hasznosak? Hogyan lehet őket használni?**

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.

12. **Az adminisztrációs feladatok elvégzésére milyen csoportokat különböztettünk meg?**

– *Rendszer-adminisztrátor* (a rövidítése SA vagy SYSADM): A rendszer-adminisztrációs csoport a legerősebb az adatbázis-kezelő rendszerben, 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.
– *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.

13. **Hogyan használhatjuk a nézeteket az adatbázis-biztonsághoz?**

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. Vertikális megszorításnak nevezzük, ha egy nézet kihagyja az alaptábla bizonyos oszlopait. Horizontális megszorításnak nevezzük, amikor a nézet az alaptáblának csak bizonyos sorait kérdezi le. Ha megadjuk a

WITH CHECK OPTION záradékot, akkor frissítéskor és törléskor biztosak lehetünk abban, hogy csak azok az adatok módosulnak, amelyek a nézetben szerepelnek.

14. **Hogyan használhatjuk a tárolt eljárásokat az adatbázis-biztonsághoz?**

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. A tárolt eljárás elindításakor az eljárás ellenőrizhet bizonyos plusz feltételeket(pl.: időpont), hogy az eljárást futtató felhasználónak van-e joga az adatokkal dolgozni.

15. **Mi az az audit?**

Az audit egy olyan adatbázis-kezelő rendszerbeli eszköz, amely lehetővé teszi az adatbázis-adminisztrátornak, hogy kövesse az adatbázis erőforrásainak a használatát. 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. Az audit utólagos tevékenység, azaz nem tesz semmit az adatbázisbeli objektum elérésének megakadályozására.

16. **Milyen problémák merülnek fel az audittal kapcsolatban?**

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 másik probléma az auditeszközökkel, hogy az auditsorokat valahol tárolni kell.

17. **Az adatbázis-kezelő rendszerek milyen auditopciókat ajánlanak fel?**

– 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

18. **Milyen tanácsokkal találkoztunk az audittal kapcsolatban?**

– 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.

19. **Operációs rendszer szinten milyen állományok védelmére kell odafigyelnie az adatbázis-adminisztrátornak?**

– 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)

20. **A feladatütemező szoftverek számára milyen jogosultságot rendeljen az adatbázis-adminisztrátor?**

Ne adjunk az ütemezőnek rendszer-adminisztrátori vagy adatbázis-adminisztrátori csoportfeljogosítást. 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.

21. **Az adatbázis-adminisztrátornak az operációs rendszerhez milyen jogosultságra van szüksége?**

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.

8. fejezet
Adatbázismentés

1. **Hogyan csoportosítottuk az adatbázishibákat?**

– *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.

2. **Mi az a 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 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.

3. A képmásolati mentés készítésének gyakoriságához milyen két legfontosabb tényezőt kell figyelembe venni?

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.

4. Mit jelent a teljes és az inkrementális mentés? Mi a különbség közöttük?

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.

5. Milyen szemcsézettséggel menthetjük az adatbázis-objektumait?

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.

6. Szükséges-e menteni az indexeket? Miért?

Az indexeket az adatok alapján az adatbázis-kezelő rendszer újra fel tudja építeni. 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.

7. El lehet-e érni az adatbázist a mentés alatt? / Milyen előnyei és hátrányai vannak annak, ha az adatbázist a mentési folyamattal párhuzamosan el lehet érni?

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.

8. Mit jelentenek a forró mentés és a hideg mentés kifejezések?

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.

9. **Miért problémásabb a forró mentés?**

– 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

10. **Miért szükséges és hogyan tudunk konzisztens helyreállítási pontot létrehozni?**

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. 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.

11. **Mikor készítsünk konzisztenciapontot?**

– 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.

12. **Szükséges-e a naplót menteni? Miért? Hogyan?**

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. 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.

13. **Hogyan lehet az adatbázis-naplózás segítségével támogatni a helyreállítást?**

A helyreállítást, illetve a hibák miatti leállás megelőzését támogatja ha az aktív adatbázisnapló-állományból több, teljesen egyforma példány létezik, amelyeket az adatbázis-kezelő rendszer párhuzamosan ír. 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.

14. **A mentés ütemezéséhez milyen kérdéseket kell feltenni a cégünknél?**

– 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?

15. **Mi befolyásolja azt, hogy milyen gyakran kell képmásolati mentést készítenünk? Hogyan?**

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 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

16. **A táblabeli adatokon kívül mi az amit még mentenünk kell? Miért?**

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

17. **Az adatbázis-kezelő rendszer mentési segédprogramján kívül még milyen más eszközökkel tudunk adatbázismentést készíteni?**

Az adatbázis-kezelő rendszer UNLOAD segédprogramját használhatjuk logikai mentés készítésére.
Tároláskezelő szoftver használata mentési másolat készítésére: Ebben az esetben az adatbázis-kezelő rendszer vezérlésén kívül történik a mentés.

18. **Miért szükséges dokumentálni az adatbázis mentési-helyreállítási stratégiáját?**

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.

19. **Milyen vezérelvek vannak az adatbázis mentési-helyreállítási stratégiájának elkészítéséhez?**

– 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.

20. **Milyen alternatívák léteznek az adatbázis mentésére és helyreállítására?**

Tartalék (Standby) adatbázisok: Az online adatbázissal azonos másolat, amely majdnem naprakész az adattartalmat tekintve. Ha hiba történik a vezérlés átkerül a tartalék adatbázisba, és a normál működés ott folytatódik.
Másolatok (Replication): Az adatmásolat azt jelenti, hogy az adatbázisban egy elszeparált adatbázis-objektumban redundáns adatokat tárolunk és tartunk karban. A pillanatfelvétel-másolat adatbázistáblák másolata a forrásadatbázis egy lekérdezésén alapszik. A lekérdezés eredménye egy pillanatfelvétel-táblába kerül. A szimmetrikus másolat sokkal robusztusabb megvalósítás, mert a másolatokat automatikusan naprakészen tartja. 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.
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.

9. fejezet
Adatbázis-helyreállítás

1. **Milyen kérdéseket tegyünk fel, amikor az adatbázisbeli hiba okát és terjedelmét keressük, amelyből megállapíthatjuk, hogy milyen lépéseket kell tennünk a helyreállításhoz?**

– 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?

2. **Milyen általános lépései vannak a helyreállításnak?**

a. 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.
b. 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.
c. 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.
d. 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.
e. 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.
f. Helyreállítás a képmásolati mentésekből: az adatbázis-helyreállító segédprogramját vagy az állományrendszer parancsát használhatjuk.
g. 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.

3. **Milyen helyreállítási típusokról beszéltünk?**

– helyreállítás az utolsó lehetséges időpontra
– point-in-time (PIT) helyreállítás (adott időpontra)
– tranzakció-helyreállítás
– hagyományos tranzakció-helyreállítás
– tároláskezelő szoftvert használata

– offsite katasztrófa-helyreállítás

4. **Mit jelent a point-in-time helyreállítás?**

A point-in-time helyreállítás minden olyan tranzakciónak a hatását eltávolítja, amely az adott időpont óta történt. 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. 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.

5. **Mit jelent a tranzakció-helyreállítás?**

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.

6. **Mit jelent a hagyományos tranzakció-helyreállítás?**

A hagyományos helyreállítási akkor van szükség, amikor az adatbázis nem normál módon áll le. 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.

7. **Mit jelent az alkalmazás-helyreállításhoz szükséges tranzakció-helyreállítás? Hogyan lehet megvalósítani?**

◦ 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.
◦ 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.

8. **Mit jelent az, hogy az utolsó lehetséges időpontra állítunk helyre?**

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 helyreállítási folyamat egy korábbi teljes mentésről indul, alkalmazza az inkrementális másolatokat, aztán előregörgeti az archív adatbázisnapló-állományokat majd az aktív naplóállományt.

9. **Napjainkban milyen típusú hibák fordulnak elő a leggyakrabban?**

– felhasználói hibák
– alkalmazáshibák/alkalmazásproblémák

10. **Milyen kérdéseket tegyünk fel, hogy megtaláljuk az optimális helyreállítási módot?**

– 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?

11. **Milyen tényezők rövidíthetik le a helyreállítás időtartamát?**

– 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.

12. **Hogyan állítsuk helyre az indexeket?**

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.

13. **A helyreállítási tervet milyen szempontok alapján hozzuk létre?**

– 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.

10. fejezet
Katasztrófa-helyreállítási terv

1. **Mit nevezünk egy cég életében katasztrófának?**

A katasztrófa egy nem tervezett, kiterjedt veszteség az üzletkritikus alkalmazások területén, mely a számítógép feldolgozó kapacitásának hiánya miatt történik.

2. **Mi a legfontosabb célja a katasztrófa-helyreállítási tervnek?**

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. 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.

3. **Kockázat szempontjából milyen alkalmazáskategóriákat határoztunk meg?**

– 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.

4. **Miért szükséges a távoli hely a cég életében? Egy cég mire használja a távoli helyet?**

Szükség van egy olyan távoli helyre, ahol a cég adatainak másolatát, mentését tárolni lehet. Egy katasztrófa esetén az itt található mentéseket használják.

5. **Milyen feltételeknek kell megfelelnie a katasztrófa-helyreállítási tervnek?**

– 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.

6. Mit kell tartalmaznia a katasztrófa-helyreállítási tervnek?

– 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.

7. Az egyszer jól megírt katasztrófa-helyreállítási tervet kell változtatni? Miért?

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.

8. Mikor kell tesztelni a katasztrófa-helyreállítási tervet?

– 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

9. Milyen mentési technikákból származó mentéseket lehet használni a katasztrófa-helyreállításhoz?

– képmásolati mentés
– tároláskezelő szoftver segítségével

10. Hogyan kell a képmásolati mentéseket előkészíteni, hogy alkalmasak legyenek a katasztrófa-helyreállításra?

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. A riportról tartsunk egyet a helyi és a távoli oldalon is. Az adatbázisnaplókat is menteni kell és át kell őket küldeni.

11. **Mi a hátránya annak, ha a tároláskezelő szoftver segítségével történik az adatbázis mentése?**

A következő lépésekből áll:
a. Állítsuk le az adatbázis-kezelő rendszert, hogy a helyreállításhoz egy rendszerszintű szilárd pontot kapjunk.
b. Másoljunk le minden adatbázis-objektumot a tároláskezelő szoftver használatával azaz mentsük el a lemez teljes tartalmát egy másik tárolóeszközre, ami lehet pl. Szalag.
c. Ha a másolás sikerült, indítsuk újra az adatbázis-kezelő rendszert.
d. 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.

12. **A képmásolati mentésen és a tároláskezelő szoftver segítségével történő mentésen kívül milyen más lehetősége van egy cégnek a katasztrófa utáni helyreállításra felkészülni?**

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. 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.

13. **Milyen irányelveket soroltunk a katasztrófa utáni helyreállításhoz?**

– A helyreállítás sorrendje
– Adatlappangás:
– Alkalmazásokhoz kapcsolódó adatok archiválása
– Tömörítés
– Helyreállítás utáni képmásolat készítése

14. **Mit jelent az adatlappangás?**

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.

15. **Meg lehet-e előzni a katasztrófák okozta károkat?**

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.

11. fejezet
Adatok elérhetősége

1. Mit jelent az elérhetőség?

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.

2. Milyen értékekkel jellemezhetjük az elérhetőséget?

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.

3. Mi a különbség az elérhetőség és a teljesítmény 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.

4. Az elérhetőségnek milyen tényezői vannak? Melyik mit jelent?

– *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.

5. Milyen kapcsolata van egymással a karbantartásnak és az elérhetőségnek?

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.

6. Milyen hatással vannak az adattárházak az operatív adatbázisok elérhetőségére?

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.

7. Milyen költsége van a leállásnak?

Az állásidő költsége cégenként változik. Minden cégnek magának kell megbecsülnie az állásidő költségét. 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. 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.

8. **Milyen problémák okozhatják azt, hogy az adatbázisunk 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

9. **Milyen lehetőségeink vannak az elérhetőség növelésére?**

Rutinkarbantartás működő rendszereken: Ha a rendszerünk teljesítményén szeretnénk javítani, akkor mindenképp be kell szereznünk néhány olyan terméket, amelyek leegyszerűsítik és automatizálják a karbantartási funkciókat. Ezek az eszközök órákról percekre vagy egy percen belülire csökkentik a karbantartási időt és a karbantartási feladatokat és ezt úgy tudják ellátni, hogy közben a felhasználók a munkájuk elvégzéséhez szükséges adatokhoz folyamatosan hozzáférnek.
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 automatizálásnak az adatbázis mentésénél és helyreállításánál vehetjük még nagy hasznát.
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. Az adatbázis-kezelő rendszernek sok olyan eszköze van, amely az elérhetőséget támogatja. Két legalapvetőbb példa ezek közül: a segédprogramok és az adatbázis rendszerparaméterei.

10. **Milyen 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

11. **A klaszterezést hogyan lehet megvalósítani a szerverhibák kezelésének szempontja alapján?**

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.

12. fejezet
Teljesítmény

1. Mi a teljesítmény definíciója?

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. Az adatbázis-teljesítmény úgy is definiálható, mint az erőforrások optimalizálása, hogy növeljük az áteresztőképességet és csökkentsük a versengést, lehetővé téve a lehető legnagyobb munkaterhelést.

2. **Mit jelentenek a következő fogalmak: 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 *erőforrásaiként* 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.

3. **Kinek a feladata egy cégnél a teljesítményproblémák megoldása?**

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.

4. **Mit jelent a Pareto-elv vagy 80/20-as szabály?**

A 80/20 szabály, vagy más néven Pareto-elv egy régi alapelv, amely azt állítja, hogy a 80%-a az eredménynek 20% erőfeszítésből származik.

5. **Milyen problémák okozhatnak gyenge teljesítményű SQL utasításokat?**

– 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)

6. **Milyen eszköz segítségével lehet megtalálni a gyenge teljesítményű SQL utasításokat?**

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.

7. **Miket érdemes ellenőrizni a teljesítmény javítása érdekében az adatbázis-kezelő rendszer és az operációs rendszerhez kapcsolódóan?**

– 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

8. **Mit jelent a teljesítménykezelés? Milyen komponensei vannak?**

A teljesítmény kezelése különbözik a teljesítmény monitorozásától, mert a kezelés kombinálja a monitorozást egy részletes tervvel, amely a fellépő problémát feloldja.

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.

9. **Mi a különbség a teljesítményproblémák proaktív és reaktív kezelésében? Lehet minden teljesítményproblémát proaktívan kezelni?**

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.

10. **Mikor kell az alkalmazáskódok teljesítményével először foglalkozni? Miért?**

A teljesítményt az alkalmazásfejlesztés életciklusában korán, a tervezés és a létrehozás alatt figyelembe kell venniük. Mert elkerülhető a költséges újratervezés, legalábbis a teljesítmény problémák tekintetében és minél korábban történik a probléma azonosítása és javítása, annál kisebb a költsége.

11. **Miért érdemes az adatbázis-adminisztrátornak a lekérdezések teljesítményéhez modellt készítenie?**

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.

12. **Mire szolgálnak a történeti információk?**

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.

13. **Mit jelent a szolgáltatási szint kezelése?**

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 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.

14. **Mit jelent a szolgáltatási szintről szóló megállapodás?**

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.

15. **Sok cégnél miért nem tudnak megállapodást kötni a szolgáltatási szintről?**

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. 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. 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.

16. **Miért hasznos a szolgáltatási szintről szóló megállapodás?**

Megjósolhatóvá teszi a teljesítmény kezelését. 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.

17. **Milyen teljesítményhangoló eszközöket ismerünk?**

– *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.

13. fejezet
Rendszerteljesítmény

1. **Az adatbázis-adminisztrátor milyen kérdések megválaszolásával biztosíthatja az optimális hardverkörnyezetet az adatbázis-kezelő rendszer számára?**

– 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?

2. **Mi az az SSD? Miért hasznos az adatbázis-kezelő rendszerek számára?**

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 adatok olvasása az SSD-ről sokkal gyorsabb, mint a hagyományos merevlemezekről.

3. **Az adatbázis-adminisztrátor milyen kérdések megválaszolásával biztosíthatja az optimális operációsrendszer-környezetet az adatbázis-kezelő rendszer számára?**

– 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?

4. **Hogyan lehet az adatbázis-kezelő rendszer konfigurációs paramétereit módosítani?**

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.

5. **Az adatbázis-kezelő rendszerek általában milyen gyorsítókkal rendelkeznek? Melyiknek mi a funkciója?**

Az *adatpuffer* vagy *adatgyorsító* az input/output műveletek minimalizálását szolgálják. A memóriában lévő adat elérése lényegesen gyorsabb, mint a lemezen lévő adat elérése. 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 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-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.

6. Az adatbázis-kezelő rendszernek a gyorsítókon kívül milyen memóriaterületekre lehet szüksége?

– 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.

7. Az adatgyorsító milyen típusú adatlapokat tartalmaz? Melyik mit jelent?

– *Tiszta adatlapok*: azok az adatlapok, amelyeket az adatbázis-kezelő rendszer korábban olvasásra használt és olvasás konzisztens 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.

8. Mekkora méretűre válasszuk az adatgyorsítót?

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.

9. Mit jelent az olvasási hatékonyság? Mitől függ ez az érték? Az olvasási hatékonyságot milyen érték mellett tekintjük megfelelőnek?

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. Egy jó adatgyorsító olvasási hatékonysága legalább 80%-os. Természetesen az olvasási hatékonyság értéke a feldolgozás típusától is függ.

10. Mi az az adatbázis-napló? Mire használja az adatbázis-kezelő rendszer?

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. 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.

11. **Mit jelent a duál naplózás?**

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.

12. **Mi az az ellenőrzési pont?**

Az adatbázis-kezelő rendszer ellenőrzési pontot hoz létre, hogy garantálja, hogy minden módosított adatbázislap biztonságban a lemezre íródott.

13. **A naplózás támogatására milyen memóriaterületeket használ az adatbázis-kezelő rendszer?**

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 adatbázisnaplóhoz olvasási naplópuffer definiálásával optimalizálhatjuk azokat a műveleteket, amelyek az adatbázisnaplót olvassák, mint a ROLLBACK, RECOVER műveletek.

14. **Miért szükséges archiválni a naplót?**

A mentési-helyreállítási stratégia megvalósításához sok esetben az adatbázisnapló mentésére is szükség van.

15. **Milyen hatással vannak a megnyitott adatbázis-objektumok a teljesítményre?**

Az egyes adatbázis-objektumok megnyitásához memóriaterületre van szükség. 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.

16. **Az adatbázis-objektumok zárolása és az objektumokért a versengés milyen hatással van a teljesítményre?**

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.

17. **Hogyan kell elhelyezni a rendszerkatalógust a teljesítmény szempontjából?**

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.

14. fejezet
Adatbázis-teljesítmény

1. **Milyen technikákat soroltunk fel az adatbázis optimalizálására? / Milyen teljesítményoptimalizáló eszközök léteznek?**

– 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

2. **Mit jelent a particionálás? Hogyan támogatja a teljesítmény javulását?**

Az adatbázis-kezelő rendszerek a fizikai állományokat az adattáblákhoz rendelik. Egy adatbázistábla az adatok halmazának logikai megjelenése, amely fizikailag a tárolón helyezkedik el. A particionálás segíti a párhuzamos adatfeldolgozást. A párhuzamos adatfeldolgozás esetén az adatbázis-kezelő rendszer az adatbázis adatait több folyamat segítségével éri el.

– 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.

3. **Mi az a raw partíció?**

Állományrendszer esetén az írásokat az operációs rendszer is puffereli. 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.

4. **Az indexelés hogyan támogatja a teljesítményt? Milyen esetekben hasznos az index?**

Indexek nélkül az adatbázis adatainak elérése úgy menne végbe, hogy az adatbázis-kezelő rendszer egy tábla minden sorát végignézné.

– 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

5. **Mennyi indexet kell egy táblára 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.

6. **Csak pozitívan hat az index az adatbázis-teljesítményre?**

Egy index negatívan hat a teljesítményre a módosításnál, ha az adat módosításánál az indexet is módosítani kell.

7. **Új index létrehozásakor miket kell tesztelni?**

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.

8. **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. Kerüljük az indexelést azon tábláknál, ahol az elérésnél mindig a teljes táblát keressük, és az olyanoknál, ahol soha nem használunk WHERE utasításrészt. 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.

9. **Mit jelent az index túlterhelése?**

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.

10. **Mit jelent a denormalizálás? Milyen lehetőségei vannak?**

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.

– 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

11. **Miért szükséges szabad helyet biztosítani az adatbázisbeli táblatereken?**

A szabad helyre azért van szükség, hogy a táblatér adatlapjainak egy része üres és elérhető legyen, amikor új adatot akarunk benne tárolni.

12. **Hogyan lehet meghatározni, hogy egy táblatérben mennyi hely maradjon szabadon?**

Az adatlapokon szükséges szabad hely nagyságát táblaterenként vagy esetleg adatbázis-objektumonként határozhatja meg az adatbázis-adminisztrátor.

13. **Milyen előnyei és hátrányai vannak annak, hogy a táblatérben szabad helyek maradnak?**

Előny:
– 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
Hátrány:
– 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

14. **Mire alapozhatja az adatbázis-adminisztrátor azt, hogy mennyi szabad helyet hagyjon az egyes táblaterekhez vagy adatbázis-objektumokhoz?**

– a beszúrások és módosítások gyakorisága
– a sorláncolás, sormigráció, és az adatlaptörések valószínűsége
Statikus táblához ne definiáljunk szabad helyet, mert nem lesz szüksége több helyre.

15. **A tömörítés milyen hatással van a teljesítményre? Miért?**

A tömörítés használható az adatbázis méretének csökkentésére. 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. 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. Az egyik oldalról lemezterületet mentünk meg és csökkenhet az input/output költség. Másrészt nagyobb lesz a processzor terhelése az adatok be- és kitömörítése miatt.

16. **Milyen szempontok szerint kell az adatbázis-adminisztrátornak az adatbázis-állományait a lemezen elhelyezni?**

Először is az indexeket és az adatokat tartalmazó állományokat különítsük el, ha lehet. Ha mindkettő ugyanazon a lemezen van, akkor az valószínűleg negatív hatással van a teljesítményre. 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.

17. **Hogyan befolyásolja az adatlap mérete a teljesítményt?**

A megfelelő adatlapméret kiválasztása fontos feladat, mert az adatbázis-adminisztrátornak ez az egyik eszköze arra, hogy az adatbázis input/output műveleteinek a teljesítményét optimalizálja. Ha az adatrekord nagyobb, mint az adatlap, akkor az adatbázis-adminisztrátor a telepítésnél nem jól választotta meg az adatlap méretét. Ez pedig teljesítményproblémákhoz vezethet.

18. **Miért szükséges újraszervezni az adatbázist?**

– *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.

19. **Hogyan lehet újraszervezni az adatbázist?**

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.

20. **Mikor kell újraszervezni az adatbázist?**

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.

21. **Lehet-e automatizálni az újraszervezést?**

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.

15. fejezet
Adatbázis-változáskezelés

1. **Soroljunk fel példákat egy cégnél történő lehetséges változásokra!**

– 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.

2. **Milyen okok miatt kell változtatni az adatbázisstruktúrát?**

– 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.

3. **Melyek a szükséges követelmények a változás sikeres megvalósításához? Pontosan mit jelentenek ezek?**

Proaktivitás: a proaktív változás jövőbeli probléma megelőzését szolgálja.
Intelligencia: 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.
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. Az egyes megvalósítások hatása meglehetősen különböző lehet.
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.
Elérhetőség: a legtöbb változás megvalósításához leállás szükséges. Manapság a magas elérhetőség a legtöbb alkalmazás esetén követelmény. 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.

4. Adatbázis-adminisztrátori szemszögből milyen változástípusokra kell számítanunk?

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.
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.
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. Az adatadminisztrációban jártas cégek azt választanák, hogy először a koncepcionális és logikai szinten végezzük el a változásokat és aztán migráljuk azt a fizikai adatbázisba.
Alkalmazások: Az alkalmazásváltozásoknak és az adatbázis-változásoknak egymással szinkronban kell lenniük. Ha az adatbázis-változás a termelési környezetbe kerül, az alkalmazásváltozásnak is át kell kerülnie. Ha az alkalmazásváltozást visszavonják, az adatbázis-változást is vissza kell vonni, és fordítva.
Fizikai adatbázis-struktúra: A legtöbb adatbázis idővel változik. 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.

5. Milyen következményei vannak annak, ha egy adatbázis-objektumot eldobunk? Mit jelent a kaszkádolt eldobás?

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.
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.

6. Milyen eszközök segítik az adatbázis-adminisztrátor munkáját a változások kezelésében?

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.

7. **Milyen lehetőségeink vannak a különböző adatbázis-környezetek összehasonlítására?**

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. Ha a cégünknek nincs adatbázisváltozás-kezelő eszköze, akkor az adatbázis létrehozásához használt szkripteket mentsük el és tartsuk naprakészen. 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.

8. **Hogyan érdemes egy cégnél a változáskéréseket kezelni?**

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. A szabványosított változás kérési rendszerrel megelőzhetjük a félre kommunikációt amely a változáskezelési folyamat alatt történhet.