**Az adatmenedzsment alapelvei**

* Pontosság
* Konzisztencia
* Biztonság
* Hozzáférhetőség

**Pontosság:**

* Az adatoknak hibák és torzítások nélkül kell tükrözniük az általuk reprezentált valós entitásokat vagy folyamatokat.
* A pontatlan adatok hibás elemzésekhez, rossz üzleti döntésekhez és az adatokon alapúló rendszerekbe vetett bizalom csökkenéséhez vezethetnek.

**Konzisztencia:**

* Az adatokat egységes formátumban kell reprezentálni, és a különböző rendszerekben és folyamatokban meghatározott standardoknak kell megfelelniük.
* A nem standardizált adatok, például az eltérő dátumformátumok vagy eltérő formátumú termékkódok hibákat okozhatnak az integráció és az elemzés során.

**Biztonság:**

* Az adatok védelme az illetéktelen hozzáféréstől, lopástól vagy más incidensektől kritikus fontosságú a bizalmasság és a jogi követelményeknek való megfelelés szempontjából.
* **A biztonsági intézkedések közé** tartozik a titkosítás, a hitelesítés és a hozzáférés-kezelés.
* Az adatvédelmi incidensek pénzügyi veszteségeket, a hírnév romlását és jogi szankciókat is eredményezhetnek. A biztonság különösen fontos az olyan érzékeny információk, mint a személyes vagy pénzügyi adatok esetében.

**Hozzáférhetőség:**

* A hozzáférhetőség biztosítja, hogy a megfelelő emberek a megfelelő időben hozzáférjenek a megfelelő adatokhoz, az operatív és stratégiai döntések támogatásához.
* A késedelemmel vagy nem hozzáférhető adatok akadályozhatják az üzleti műveleteket és befolyásolhatják a döntéshozatalt.

**Az adatmenedzsment területei**

* **Adatarchitektúra:** Az adatvagyon struktúrájának és az adatáramlás folyamatainak átfogó tervezése és optimalizálása.
* **Adatbázis-adminisztráció**: Az adatbázisok zavartalan működésének biztosítása, különös tekintettel a teljesítményre, a biztonságra és az elérhetőségre.
* **Adatintegráció:** Különböző forrásokból származó adatok összehangolása és egységesítése az elemzés és felhasználás céljából.
* **Adatmodellezés:** Az adatok logikai és fizikai szerkezetének, valamint azok közötti kapcsolatoknak a meghatározása és megtervezése.
* **Adatminőség-menedzsment:** Az adatok pontosságának, konzisztenciájának és megbízhatóságának fenntartása, a hibák azonosítása és javítása által.
* **Adatgazdálkodás:** Az adatok kezelési és felhasználási szabályainak, irányelveinek és felelősségi körének meghatározása

**Adatarchitektúra**

* Az adatarchitektúra egy keretrendszer, amely megadja, hogy az adatok hogyan áramlanak és tárolódnak egy szervezeten belül.
* **Leírja az**
* **Adatvagyont:** Az összes adatforrás dokumentációja.
* **Adatáramlást:** Hogyan mozognak az adatok az egyes rendszerek között.
* **Technológiai keretrendszert:** Adatbázisok, tároló rendszerek és elemzési platformok

**Adatbázis-adminisztráció**

* Egy adatbázis adatok szervezett gyűjteménye, amelyet úgy terveztek, hogy lehetővé tegye az információk hatékony tárolását, visszakeresését, frissítését és kezelését különböző alkalmazások számára.
* Adatbázis-kezelő rendszerek (DBMS) az adatbázisok kezelésére használt szoftverek.

**Fő típusok:**

* **Relációs (RDBMS):** strukturált táblákat és SQL-t használ
* **NoSQL:** strukturálatlan és félig strukturált adatokat kezel
* **Speciális:** pl. idősoros, gráf és in-memory adatbázisok

**Az adatbázis-adminisztrátor (DBA) felelős a szervezet adatbázisainak kezeléséért és karbantartásáért.**

**A DBA-k főbb feladatai:**

* Adatbázisok tervezése.
* A teljesítmény monitorozása és az adatbázis-kezelő rendszer, és a lekérdezések optimalizálása.
* Biztonsági mentési és katasztrófa-helyreállítási megoldások megvalósítása.
* Biztonsági protokollok érvényesítése az érzékeny adatok védelme érdekében.

**Adatintegráció**

Az adatintegráció az adatok több forrásból történő kombinálásának folyamata, amely lehetővé teszi az átfogó elemzést, riportkészítést és operatív felhasználást.

* **ETL (Extract, Transform, Load):** Az adatok kinyerése a forrásrendszerekből, átalakítása konzisztens formátumokba, majd betöltése a célrendszerbe.
* **ELT (Extract, Load, Transform):** Az adatokat először kivonják és betöltik a célrendszerbe, ahol az átalakítás később történik, jellemzően a lekérdezéskor vagy az elemzés során.
* **Valós idejű integráció:** Az adatok integrálása és frissítése folyamatosan vagy azonnal, ami közel valós idejű elemzést tesz lehetővé

**Adatmodellezés**

Az adatmodellezés az adatelemek és azok közötti kapcsolatok leírásának folyamata, amely az adatok strukturálása és szervezése érdekében a hatékony felhasználás és elemzés érdekében.

**Modellek típusai:**

* **Koncepcionális modell:** Az adatok szerkezetének és céljának magas szintű, átfogó leírása.
* **Logikai modell:** Az adatelemek, kapcsolatok és attribútumok részletes leírása, technológiafüggetlen módon.
* **Fizikai modell:** A tényleges megvalósítás leírása egy adatbázis kezelő rendszerben, figyelembe véve a technikai megvalósítási szempontokat

**Adatminőség-menedzsment**

Az adatminőség az adatok pontossága, konzisztenciája és rendeltetésének való megfelelősége.

**Az adatminőség javításának módszerei:**

* **Profilozás:** A kiugró értékek, anomáliák és mintázatok azonosítása az adatokban.
* **Tisztítás:** Az adatkészletek hibáinak vagy ellentmondásainak kijavítása.
* **Validálás:** Annak biztosítása, hogy az adatok megfeleljenek az előre meghatározott szabályoknak és szabványoknak.

**Adatgazdálkodás**

Az adatgazdálkodás olyan irányelvek és eljárások rendszere, amelyek biztosítják az adatok megfelelő felhasználását.

**Főbb elemei:**

* **Szabályzatok:** Meghatározzák az adatok hozzáférésére és használatára vonatkozó szabványokat.
* **Bizottságok:** Az irányítást felügyelő tanácsok, amelyek biztosítják a szabályok betartását.
* **Adatgondnokság (data stewardship):** Olyan szerepkörök, amelyek felelősek a szabályzatoknak való megfelelés biztosításáért

**Szabályozási megfelelés céljai**

A szervezeteknek meg kell felelniük az adatvédelmi törvényeknek (pl. GDPR), hogy megvédjék az érzékeny információkat, fenntartsák a bizalmat és elkerüljék a bírságokat.

**A fogyasztói adatok védelme:**

* A személyes adatok bizalmas kezelésének, sértetlenségének és biztonságának biztosítása.
* Az adatkezeléssel kapcsolatos elszámoltathatóság megteremtése:
* Az adatgyűjtésre, -tárolásra és -megosztásra vonatkozó iránymutatások egyértelmű meghatározása.

**Az adatok megfigyelhetősége**

Az adatok megfigyelhetősége (data observability) az adatok és „csővezetékeik” állapotának, megbízhatóságának és teljesítményének ellenőrzésére összpontosít a zökkenőmentes működés biztosítása érdekében.

**Főbb elemei:**

* **Monitoring:** Az adatminőség és a csővezeték teljesítményének automatikus figyelése.
* **Riasztás:** Valós idejű értesítések az érintett személyeknek az adatokkal kapcsolatos problémákról vagy hibákról.
* **RCA (root cause analysis):** Azonosítja és meghatározza a csővezeték problémáinak forrását azok megoldásához.

**Információs rendszerek**

* **Egy információs rendszer (IR) komponensek egy olyan együttese, amely lehetővé teszi az adatok gyűjtését, tárolását és feldolgozását, valamint információk vagy digitális termékek biztosítását.**
* **Fő komponensek:**
  + **adatbázisok**
  + **alkalmazások**
  + **szoftverek**
  + **hardverek**
  + **emberek**

**IR – Komponensek**

**Alkalmazások**

* **Egy adott szervezet igényeire szabott szoftverek.**
* **Lehetnek dobozos szoftverek vagy teljesen egyedi fejlesztésű megoldások.**
* **Az alkalmazások adatbázisokból származó adatokat használnak**

**olyan funkciók megvalósításához, mint a tranzakciók feldolgozása,**

**munkafolyamatok kezelése, vagy elemzések és jelentések készítése.**

* **Példák: vállalatirányítási (ERP) rendszerek, ügyfélkapcsolat-kezelő (CRM) rendszerek, üzleti intelligencia eszközök, például jelentések és irányítópultok készítésére.**

**Szoftverek**

* **Az alkalmazásokon és a DBMS-en túl magába foglalja:**

**▫ *Operációs rendszerek:* Alapvető szoftverek, amelyek a hardverek erőforrásait kezelik, és szolgáltatásokat biztosítanak a programok számára.**

**▫ *Middleware:* „Köztes szoftver”, ami az alkalmazásokat vagy adatbázisokat köti össze, lehetővé téve azok interoperabilitását és az adataik kezelését.**

**▫ *Segédszoftverek:* Rendszerkarbantartási eszközök, biztonsági**

**szoftverek, fejlesztői eszközök stb.**

**Hardver**

* **Az információs rendszer működtetéséhez és támogatásához szükséges**

**fizikai ICT komponensek.**

* **Főbb példák:**

**▫ *Szerverek:* Számítógépek, amelyek az alkalmazásokat, a DMBS-t,**

**és a különféle hálózati szolgáltatásokat futtatják.**

**▫ *Tárolók:* A helyszíni (azaz „on-premises”; pl. SAN vagy NAS**

**rendszerek, helyi merevlemezek), és a felhőalapú megoldások is.**

**▫ *Hálózati infrastruktúra:* Router-ek, switch-ek, tűzfalak és egyéb**

**eszközök, amelyek lehetővé teszik a rendszer belső és külső**

**kommunikációját.**

**Emberek**

* ***Végfelhasználók:* Azok a személyek, akik használják a rendszert,**

**beleértve az alkalmazottakat is, az adatrögzítőktől a vezetőkig.**

* ***Adminisztrátorok:* A rendszer konfigurálásáért, karbantartásáért, biztonságáért és teljesítményéért felelős szakemberek.**
* ***Fejlesztők:* Az IR szoftveres komponenseit tervező, fejlesztő és**

**karbantartó informatikusok.**

* ***Elemzők:* Szakértők, akik az üzleti igényeket elemzik, a munkafolyamatokat optimalizálják, és biztosítják, hogy a rendszer megfeleljen a szervezeti céloknak.**

**SDLC**

* **A szoftverfejlesztési életciklus (software development life cycle, SDLC) egy keretrendszer, amelyet szoftverrendszerek tervezésére, fejlesztésére és karbantartására használnak.**

**Fázisok:**

1. **Követelmények feltárása**
2. **Elemzés**
3. **Tervezés**
4. **Implementáció**
5. **Tesztelés és integráció**
6. **Üzemeltetés és karbantartás**
7. **Követelmények feltárása (Planning)**
   * **A rendszer céljainak meghatározása, a meglévő rendszerek értékelése és a szükséges erőforrások azonosítása (magas szintű követelmények).**
   * **Megvalósíthatósági tanulmányok készítése (technikai, működési, pénzügyi szempontok figyelembevételével).**
8. **Elemzés (Requirement Analysis)**
   * **A felhasználói követelmények összegyűjtése és az üzleti folyamatok dokumentálása.**
   * **Funkcionális és nem funkcionális követelmények meghatározása.**

**▫ Adatfolyam-diagram (data flow diagram, DFD) és egyed-kapcsolat (entity-relationship, ER) modell.**

1. **Tervezés**
   * **A rendszer architektúrájának, adatbázis-struktúrájának, felhasználói**

**felületeinek és munkafolyamatainak műszaki specifikációja.**

* + **A rendszer moduláris, skálázható és biztonságos kialakításának biztosítása.**
  + **A rendszerterv elkészítése („tervrajz” az implementációhoz).**

**4. Implementáció**

* **A rendszer kódolása, tesztelése és telepítése iteratív vagy inkrementális módszertannal (pl. agilis, vízesés).**
* **Hardver/szoftver telepítése, adatbázisok létrehozása és adatmozgatás.**

1. **Tesztelés**
   * **A rendszer követelményeknek való megfelelésének ellenőrzése különböző tesztelési szinteken, pl. egység-, integrációs, rendszer-, és felhasználói elfogadási teszteléssel.**
   * **A teljesítmény, a biztonság, a megbízhatóság és használhatóság vizsgálatának elvégzése.**
2. **Üzemeltetés és karbantartás**
   * **A rendszerteljesítmény figyelése, hibajavítások és frissítések alkalmazása.**
   * **Rendszeres biztonsági mentések, auditok és optimalizálás a hosszú**

**távú megbízhatóság érdekében.**

* **1. Követelmények felmérése (megvalósíthatóság):**

**▫ *Adatigények azonosítása:* A rendszer működéséhez szükséges**

**kulcsfontosságú adatok meghatározása.**

**▫ *Meglévő adatforrások értékelése:* Az adatok már rendelkezésre állnak, beszerezhetők vagy újonnan létrehozandók?**

**▫ *Adatkövetelmények dokumentálása:* A *magas szintű* adatszerkezet**

**és adatáramlás meghatározása.**

**▫ *Megvalósíthatóság elemzése:* A meglévő adatbázisok**

**integrálhatóságának és új adatbázisok kialakításának vizsgálata.**

**▫ *Stratégiai tervezés:* Hosszú távú adatkezelési stratégiák,**

**megfelelőség és biztonsági követelmények figyelembevétele.**

* **2. Elemzés (követelmények, koncepcionális modellezés):**

**▫ *Adatigények meghatározása:* Annak dokumentálása a**

**felhasználók bevonásával, hogy milyen adatokra van szükség és**

**hogyan fogják azokat használni.**

**▫ *Koncepcionális modell létrehozása:* Egy modell készítése az**

**egyedek, attribútumok és a kapcsolatok definiálására (ER modell).**

**▫ *Konzisztencia és integritás biztosítása:* Az adatigények közötti potenciális ellentmondások azonosítása és feloldása.**

**▫ *Adatforrások elemzése:* Annak meghatározása, hogy a szükséges**

**adatok belső és/vagy külső forrásból kerülnek majd beszerzésre.**

**▫ *Validációs szabályok megadása:* Üzleti szabályok definiálása az adatok bevitelére és ellenőrzésére.**

* **3. Tervezés (logikai és fizikai modellezés):**

**▫ *Logikai modell létrehozása:* Az ER-modell átalakítása egy**

**strukturált sémává (pl. relációs modellé).**

**▫ *DBMS kiválasztása:* A megfelelő DBMS (pl. Oracle, MySQL, PostgreSQL) kiválasztása az üzleti igények alapján.**

**▫ *A séma optimalizálása:* Normalizálás a redundancia**

**minimalizálása és a hatékonyság növelése érdekében.**

**▫ *Fizikai tárolás meghatározása:* Az indexelési stratégiák, partíciók, tárhely kiosztás stb. megadása.**

**▫ *Biztonság és hozzáférés-szabályozás tervezése:* Felhasználói**

**szerepkörök, jogosultságok és titkosítási stratégiák definiálása.**

* **4. Implementáció (adatbázis létrehozása, adatbetöltés):**

**▫ *Adatbázis séma létrehozása:* A fizikai adatbázis szerkezetének**

**kialakítása (táblák, indexek, megszorítások).**

**▫ *Adatmigrációs szkriptek fejlesztése:* Az adatok kinyerése, átalakítása és betöltése (extract, transfer, load, ETL) régi rendszerekből vagy külső forrásokból.**

**▫ *Hozzáférési és biztonsági szabályok beállítása:* Felhasználói**

**szerepkörök és hozzáférési korlátozások megvalósítása.**

**▫ *Lekérdezési teljesítmény optimalizálása:* Indexek, gyorsítótárak és lekérdezés-végrehajtási tervek finomhangolása.**

**▫ *Adatmentési eljárások kialakítása:* Rendszeres biztonsági**

**mentések és vészhelyreállítási stratégiák kidolgozása.**

* **5. Tesztelés és integráció:**

**▫ *Tesztadatok betöltése:* Az adatbázis feltöltése tesztadatokkal, a**

**valós környezet szimulálásához.**

**▫ *Adatintegritás ellenőrzése:* Hivatkozási integritás, elsődleges kulcs megszorítások és üzleti szabályok érvényesítése.**

**▫ *Teljesítménytesztelés:* Lekérdezési válaszidők mérése és az**

**indexelés optimalizálás szükség szerint.**

**▫ *Biztonsági tesztelés:* Szerepkör-alapú hozzáférési szabályok ellenőrzése, titkosítási mechanizmusok tesztelése.**

**▫ *Integrációs tesztelés:* Annak biztosítása, hogy az alkalmazások és a**

**külső rendszerek megfelelően kapcsolódnak az adatbázishoz.**

* **6. Üzemeltetés és karbantartás**

**▫ *Monitorozás:* A lekérdezések teljesítményének, a tárhely**

**kihasználtságának és a terhelés rendszeres áttekintése.**

**▫ *Karbantartás:* Az indexek újraépítése, a statisztikák frissítése és a régi adatok archiválása.**

**▫ *Biztonság és megfelelőség:* A hozzáférési naplók áttekintése, a**

**biztonsági házirendek frissítése és auditok végrehajtása.**

**▫ *Biztonsági mentés és helyreállítás:* Az adatbázis pillanatképek és tranzakciónaplók karbantartása.**

**▫ *Az adatbázis fejlesztése:* Pl. a séma és az indexelési stratégiák**

**módosítása az üzleti igények változásaihoz igazítva.**

**Az Adatbázis életciklusa - DBLC**

**Az adatbázis életciklusa (database life cycle, DBLC) az SDLC-**

**be integrálódik:**

1. **Kezdeti vizsgálat – Az adatigények és korlátozások azonosítása.**
2. **Koncepcionális tervezés – Magas szintű adatmodell létrehozása.**
3. **Logikai tervezés – A koncepcionális modell átalakítása egy adatbázis sémává.**
4. **Fizikai tervezés – Az adatbázis fizikai szerkezetének meghatározása (tárolás, indexek stb.).**
5. **Implementáció és tesztelés – Az adatbázis létrehozása, kihelyezése**

**és validálása.**

1. **Üzemeltetés és karbantartás – Az adatbázis optimalizálása és frissítése az esetlegesen változó igényekhez.**

**Koncepcionális tervezés**

* **A koncepcionális tervezés az adatbázis-tervezés első szakasza, amely az adatok és kapcsolatok egy absztrakt leírását hozza létre.**
* ***Célja:* Egy szoftver- és hardverfüggetlen adatmodell kidolgozása,**

**amely tükrözi a valós világ releváns egyedeit és kapcsolatait.**

* ***Eredménye:* Egy olyan adatmodell, amely alapot biztosít a logikai és fizikai adatbázis-tervezéshez.**

**▫ Biztosítja az adatok pontosságát és teljességét a megvalósítás előtt.**

**▫ Meghatározza az egyedeket, attribútumokat, kapcsolatokat és megszorításokat.**

**▫ Hidat képez az üzleti követelmények és a megvalósítás között.**

**A DBMS kiválasztásának főbb szempontjai:**

* ***Költség:* beszerzési ár, licencdíj, a karbantartás, képzés és**

**üzemeltetés költsége stb.**

* ***Funkciók, eszközök:* példaalapú lekérdezés (Query by Example,**

**QBE), jelentésgenerálás, alkalmazásfejlesztési eszközök stb.**

* ***Adatmodell:* relációs, objektum-relációs, objektumorientált,**

**hierarchikus vagy hálózati modell**

* ***Hordozhatóság:* Képes-e a DBMS több platformon, operációs rendszeren stb. működni?**
* ***Hardverkövetelmények:* processzor-, memória- és tárhelyigény,**

**kompatibilitás**

**Logikai tervezés**

* **A logikai tervezés fókusza az adat szerkezetének definiálása, a fizikai tárolástól független módon.**
* ***Célja:* A koncepcionális modell átalakítása egy strukturált sémává**

**egy kiválasztott adatmodell alapján (pl. relációs modell).**

* ***Eredménye:* Egy logikai adatmodell, amely meghatározza pl. a táblákat, attribútumokat, kapcsolatokat és megszorításokat.**

**▫ Strukturált és normalizálva reprezentáció kialakítása.**

**▫ Az adatok integritásának és konzisztenciájának biztosítása a fizikai megvalósítás előtt.**

**▫ Szoftver- és platformfüggetlen szemléletben készül.**

**Fizikai tervezés**

* **A fizikai tervezés az adatbázis-tervezés utolsó szakasza.**
* ***Célja:* Az adatok integritásának, biztonságának és az elérés teljesítményének biztosítása a tárolás és elérés technikai részleteinek megadásával és hangolásával.**
* ***Eredménye:* A működő, optimalizált adatbázis.**

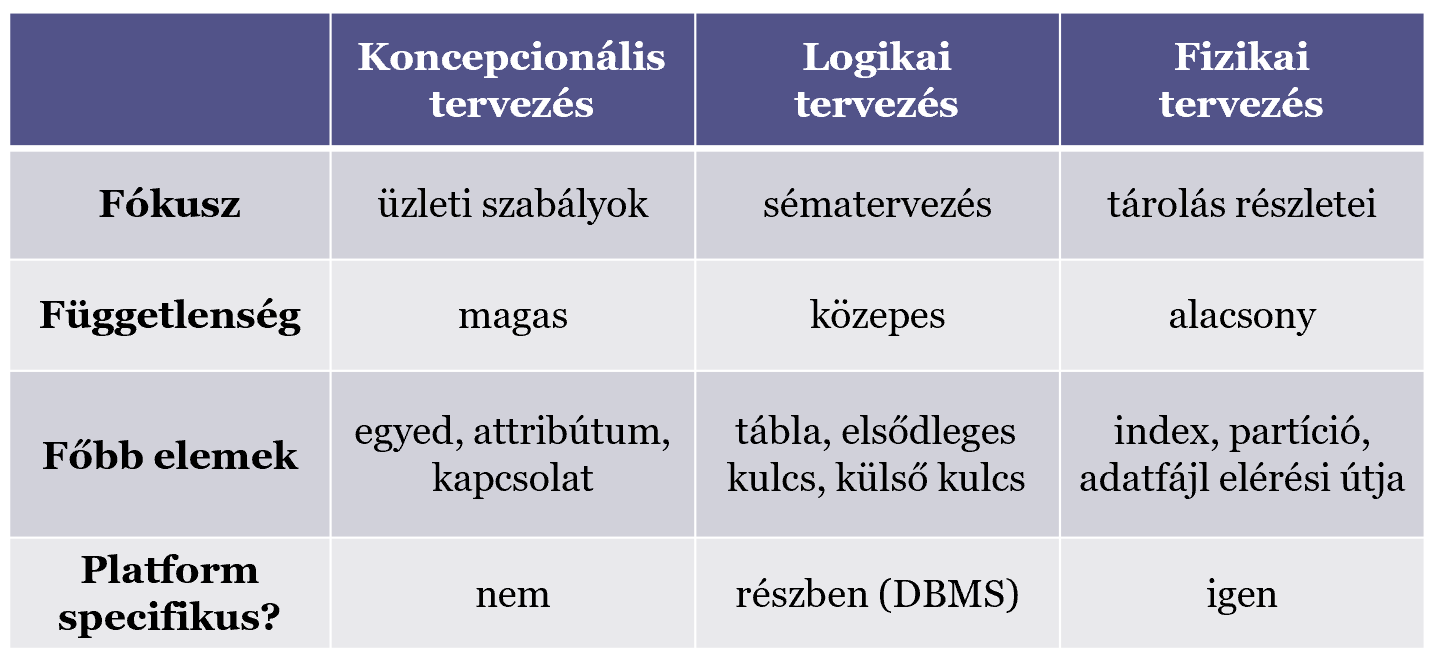
**▫ Az adatok fizikai tárolásának és elérésének meghatározása.**

**▫ A lekérdezések és tranzakciók megfelelő teljesítményének biztosítása.**

**▫ Az adatbiztonsági intézkedések implementálása.**

**▫ A tárhely allokáció és az indexelés hangolása a hatékonyság**

**növelése érdekében.**

****

**Adatmodellezés**

* **Mi az adatmodellezés?**

**▫** Egy strukturált megközelítés az egyedek, attribútumok, kapcsolatok

és megszorítások definiálására.

▫ Biztosítja az adatok konzisztenciáját, integritását, hatékony tárolását és lekérdezését.

* **Miért fontos az adatmodellezés?**

**▫** Kapcsolat az üzleti követelményeket az adatbázis-tervezés között.

▫ Csökkenti az adatok redundanciáját és javítja a lekérdezések teljesítményét.

▫ Biztosítja a skálázhatóságot a rendszer jövőbeli növekedésének

támogatását.

* **Az adatmodell egy formális leírás, amely meghatározza az adatelemek**

**struktúráját, egymáshoz való kapcsolatait és az adatok kezelését egy**

**adatbázis-rendszeren belül.**

* Egy keretrendszert biztosít az adatok hatékony szervezésére és kezelésére.
* **Főbb funkciói:**

**▫** Meghatározza az adatok szervezési módját (egyedek, attribútumok,

kapcsolatok).

▫ Biztosítja az egységes formátumokat és megszorításokat (adatintegritási szabályok).

▫ Csökkenti a redundanciát és az inkonzisztenciát (normalizálás,

megszorítások).

▫ Fizikai szinten támogatja az indexelést és a teljesítményoptimalizálást.

**Az adatmodellek értelmezése**

Fontos, hogy az „adatmodell” két különböző dolgot is jelenthet.

1. **Adatbázismodell:**

▫ Koncepcionális keretrendszer az adatok tárolására.

▫ Meghatározza, hogyan van az adatelem strukturálva és tárolva egy adatbázis-rendszerben.

▫ Különböző megközelítéseket képviselnek az adatbázis-tervezés és funkcionalitás szempontjából.

* + Példák:

▫ relációs modell, objektum orientált modell, hierarchikus modell,

hálózati modell, hibrid modellek

**Az adatmodellek megértése**

Fontos, hogy az „adatmodell” két különböző dolgot is jelenthet.

1. **Adatmodellezési szintek:**

**▫** Az adatbázis-tervezés során az egyes szintek leírása.

▫ Az absztrakt ábrázolástól a megvalósításig tartó átmenetet írják le.

* + Három tervezési szakasz:

▫ **koncepcionális tervezés** (magas szintű, üzleti fókuszú)

→ koncepcionális modell

▫ **logikai tervezés** (részletes szerkezet, platformfüggetlen)

→ logikai modell

▫ **fizikai tervezés** (megvalósítás-specifikus)

→ fizikai modell

**Adatmodellek (Adatbázismodellek)**

**Korai modellek:** Hierarchikus modell, hálózati modell

**Modern modellek:** relációs modell, objektumorientált modell

**Kiterjesztett/hibrid modellek:** Objektum-relációs modell, NoSQL modellek

**Miért használnak különböző modelleket?**

**Az alkalmazások eltérő követelményeket támasztanak.**

* Az iparágak eltérő igényekkel és felhasználási esetekkel rendelkeznek,

ami különböző adatszervezési és hozzáférési mintákhoz vezet.

* A megfelelő adatbázismodell javíthatja a hatékonyságot, a

skálázhatóságot és a konzisztenciát az adott alkalmazás számára.

**Folyamatos fejlődés az adott kor kihívásainak megfelelően.**

* Az adatbázismodellek fejlődése a merev struktúráktól (pl. hierarchikus és hálózati modellek) a rugalmasabb megoldások felé (pl. gráf adatbázisok).
* Minden modell célja az, hogy optimalizálja a teljesítményt, a

kapcsolatok kezelését és a lekérdezések hatékonyságát.

**Modellezés kulcsfogalmak**

* **Séma:** Az adatbázis teljes (koncepcionális/logikai) felépítése.
* **Alséma:** Meghatározza az adatbázis azon részeit, amelyek egy adott

alkalmazás számára láthatóak.

* **Adatmanipulációs nyelv (DML):** Adatok hozzáadására, törlésére és módosítására szolgál az adatbázisban.
* **Adatdefiníciós nyelv (DDL):** Az adatbázis-adminisztrátorok által használt nyelv a sémaelemek meghatározására.
* **Alséma DDL:** Lehetővé teszi az alkalmazások számára, hogy saját igényeikhez igazodó adatszerkezeteket definiáljanak.

**Hierarchikus modell**

* **Kapcsolatok:**

▫ Minden **szülőnek** több gyermeke lehet.

▫ Minden **gyermeknek** csak egy szülője van.

* **Hierarchikus útvonal:**

▫ Az adatok fa struktúrában vannak szervezve.

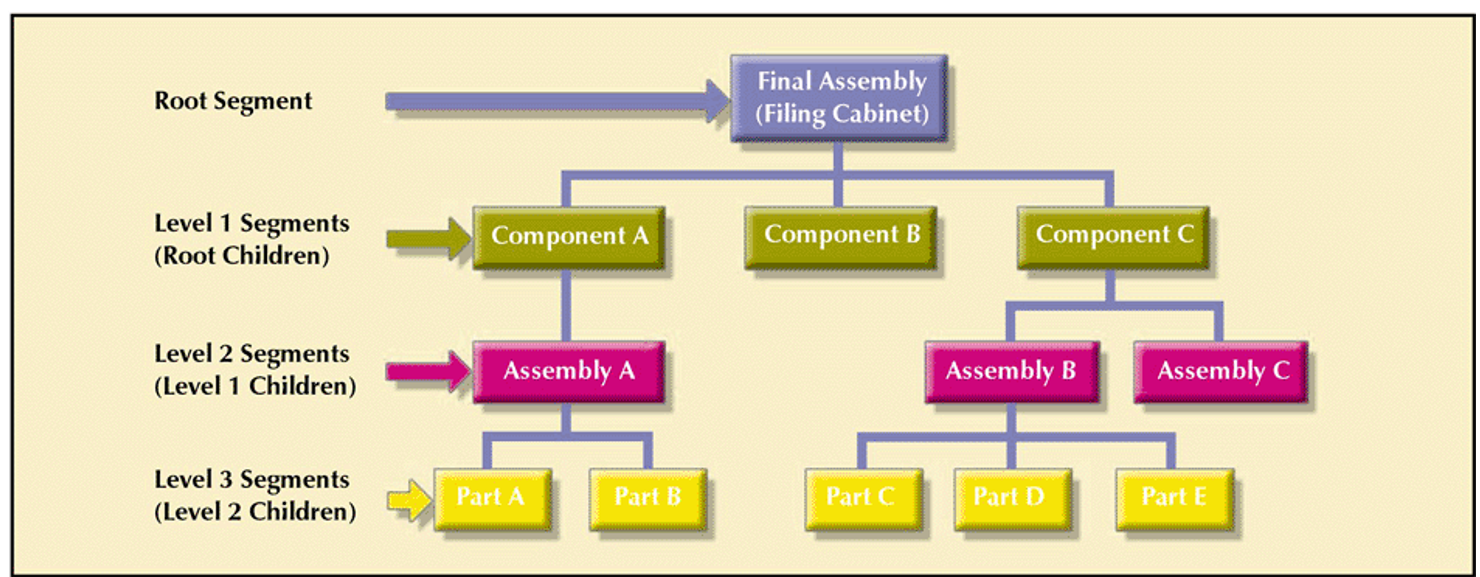
▫ A szegmensek rendezett sorrendben követik egymást, preorder

bejárás szerint (felüről lefelé, balról jobbra).

* **Optimalizálás:**

▫ A gyakran elért adatok kerüljenek közelebb a gyökérhez.

▫ *Példa:* Ha a D alkatrész a leggyakrabban használt, akkor azt balra célszerű helyezni a fában.



**Hierarchikus modell – előnyök**

* **Koncepcionális egyszerűség:**

▫ A fa struktúra intuitív és könnyen érthető.

▫ Az egyértelmű szülő-gyermek kapcsolatok egyszerűsítik az adatok szervezését.

* **Adatfüggetlenség (alapszintű):**

▫ Ha egy adatelem típusa megváltozik, a DBMS automatikusan

propagálja a módosítást az egész adatbázisban.

▫ Csökkenti a manuális módosítások szükségességét azokban az

alkalmazásokban, amelyek a megváltozott adatelemre hivatkoznak.

* **Adatbázis-integritás:**

▫ Erős szülő-gyermek kapcsolatok biztosítják a hivatkozási integritást.

▫ Megakadályozza az árva rekordok létrejöttét (minden gyermekhez kötelezően tartoznia kell egy szülőhöz).

* **Hatékonyság:**

▫ 1:N kapcsolatokhoz optimalizált, ahol minden szülőhöz több

gyermek tartozik.

▫ Gyors lekérdezés a meghatározott hierarchia mentén történő keresések esetében.

▫ Kiemelkedően hatékony nagy méretű adathalmazoknál, ahol a

kapcsolatok idővel nem változnak.

**Hierarchikus modell – hátrányok**

* **Összetett megvalósítás:**

▫ Részletes ismereteket igényel a fizikai adattárolásról.

▫ A fejlesztőknek alacsony szintű tárolási részletekkel is foglalkozniuk

kell.

* **Nehezen kezelhető és módosítható:**

▫ A lekérdezésekhez és frissítésekhez előre meghatározott útvonalakra

van szükség, ami bonyolítja az adatkezelést.

▫ A hierarchia módosítása (például új szülőszint beillesztése) hatással van az alkalmazás logikára is.

▫ A szerkezeti változtatások időigényesek és hibalehetőségeket rejtenek

magukban.

* **Implementációs korlátok:**

▫ Nehéz M:N kapcsolatokat kezelni, mivel a modell alapvetően 1:N

kapcsolatokra épül.

▫ A megkerülő megoldások (pl. redundáns adatok vagy „köztes táblák”) használata komplexebbé tehetik az adatbázist.

* **Szabványosítás hiánya:**

▫ Nincs általános adatdefiníciós vagy adatmanipulációs nyelve.

▫ Minden hierarchikus adatbázis-rendszer saját, egyedi elérési módszereket és lekérdezőnyelveket alkalmazhat.

**Hálózati modell**

* Hasonlít a hierarchikus modellre, de egy rekord több szülővel is

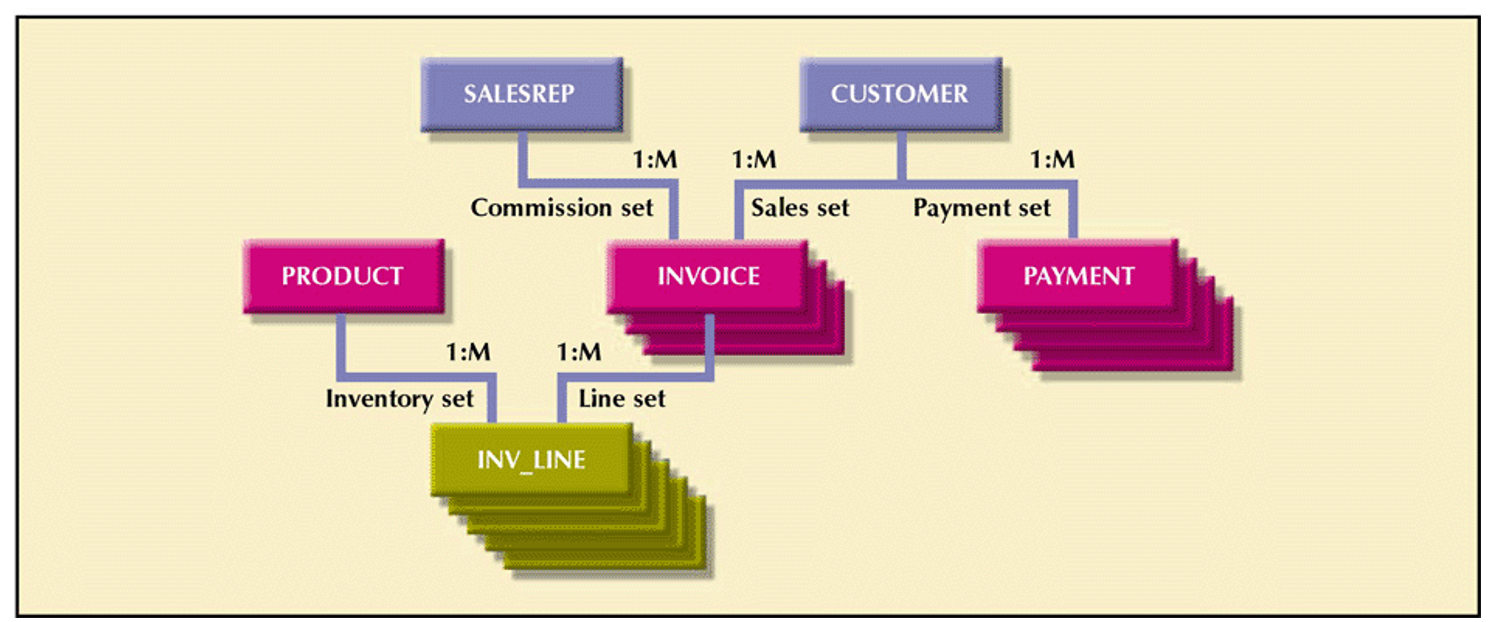
rendelkezhet.

* Az adatok rekordokban vannak szervezve, amelyeket **halmaz**okon

keresztül kötnek össze.

* Minden halmazban van:
  + **tulajdonos** (owner) (hasonló a szülőhöz a hierarchikus modellben)
  + **tag** (member) (hasonló a gyermekhez a hierarchikus modellben)
* Egy tag több halmazhoz is tartozhat, ami

összetettebb kapcsolatokat tesz lehetővé.



**Hálózati modell – előnyök**

* **Rugalmasabb kapcsolatok:**

▫ Hatékonyan modellezi az M:N kapcsolatokat, ellentétben a hierarchikus adatmodellel.

* + Lehetővé teszi a többszörös szülő-gyermek kapcsolatokat.

▫ Ideális összetett kapcsolatokkal rendelkező alkalmazásokhoz (pl.: ERP rendszerek, telekommunikáció).

* **Hatékony hozzáférés az adatokhoz:**

▫ Gyorsabb lekérdezéseket tesz lehetővé, mivel közvetlen hozzáférést biztosít a kapcsolódó rekordokhoz.

* + Nem igényel speciális bejárási műveleteket, mint a hierarchikus

modell.

* **Adatbázis-integritás elősegítése:**

▫ A rekordok közötti kapcsolatokat előre meg kell határozni az adatok

bevitele előtt.

▫ Biztosítja a hivatkozási integritást azáltal, hogy a tulajdonos-tag kapcsolatok kötelezőek.

* **Adatfüggetlenség:**

▫ Az alkalmazások működése nem igényli a fizikai tárolási struktúra ismeretét, így rugalmasabb adatkezelést biztosítanak.

* **Szabványosítás:**

▫ Adatmanipulációs nyelv és séma szabványok adaptálása.

▫ Előre meghatározott szabályok vannak az adatbázis szerkezetére és műveleteire, amelyek javítják a hordozhatóságot és a konzisztenciát.

**Hálózati modell – hátrányok**

* **Strukturális függetlenség hiánya:**

▫ A kapcsolatok módosítása alkalmazáson belüli változtatásokat is

igényelhet, és hatással lehet a teljesítményre.

* **Összetett lekérdezések:**

▫ Az adatok elérése gyakran előre meghatározott útvonalakon történik, ami rugalmatlanná teszi a lekérdezéseket.

▫ A lekérdezések során explicit módon meg kell adni a tulajdonos-tag

kapcsolatokat, ami megnehezíti az optimalizálást.

* **Korlátozott skálázhatóság:**

▫ Nagyon nagy adathalmazok esetén a modell kevésbé hatékony lehet.

▫ Ahogy az adatok összetettsége növekszik, a teljesítmény csökkenhet.

* *Megjegyzés: A hálózati modell ma már elsősorban csak legacy rendszerek használják.*

**Relációs modell - alapszerkezet**

* **Tábla (reláció):**

▫ Az adatokat mátrixszerűen tárolja sorokban és oszlopokban.

▫ Minden sor (tuple) egyedi adatrekordot tárol.

▫ Minden oszlop (attribútum) az adatelem egy tulajdonságát jelöli.

▫ A táblák közös attribútum(ok) (elsődleges és külső kulcsok) alapján

kapcsolódnak egymáshoz.

* **Relációs séma:**

▫ Meghatározza a táblákat, azok attribútumait és a táblák közötti

kapcsolatokat.

▫ Leírja az adatok szervezését és az integritási megszorításokat.

**Relációs modell - táblák**

* A **relációs táblák** hatékony lekérdezéseket és kapcsolatokat tesznek

lehetővé.

* Egy tábla koncepcionálisan hasonlít egy strukturált fájlhoz, ahol minden

sor egy rekordnak felel meg és minden oszlop egy attribútumot jelöl.

* Azonban:

▫ A tábla egy *absztrakt adatreprezentáció*, független a fizikai tárolástól.

▫ A táblák közötti kapcsolatokat kulcsok (elsődleges és külső kulcsok)

kezelik, nem pedig fizikai kapcsolatok.

▫ A felhasználóknak és a DBA-knak nem kell ismerniük az adatok fizikai tárolási módját.

▫ A DBMS kezeli az adatok lekérdezését és optimalizálását.

**Relációs modell - előnyök**

* **Strukturális függetlenség:**

▫ Az adatszerkezet módosítása nem befolyásolja a DBMS adatelérési

módszereit.

▫ Rugalmas adatbázis-tervezést tesz lehetővé, mivel a módosítások nem igényelnek alapvető módosításokat az alkalmazás kódjában.

▫ Egységesített és szabványosított módszerekkel segíti elő az adatok

kezelését és konzisztenciáját.

* **Egyszerűbb logikai modellezés:**

▫ Az adatok logikai nézetére helyezi a hangsúlyt, így könnyebben érthető és kezelhető.

▫ A kapcsolatokra és az adatintegritásra fókuszál, nem pedig a fizikai tárolás részleteire.

* **Ad hoc lekérdezési képesség:**

▫ Támogatja a rugalmas lekérdezéseket SQL használatával.

▫ Lehetővé teszi a felhasználók számára az adatok lekérdezését és

elemzését anélkül, hogy előre meghatározott lekérdezési struktúrákra lenne szükség.

**Relációs modell - hátrányok**

* **Magas hardverigény és szoftveres „overhead”:**

▫ Jelentős hardveres erőforrásokat igényel az adattárolás, feldolgozás

és indexelés miatt.

▫ Nagy méretű adatbázisok esetén a teljesítmény csökkenhet, ha nincs megfelelő optimalizálás.

* **Lehetséges tervezési és megvalósítási problémák:**

▫ Nem megfelelő sématervezés esetén a lekérdezések lassúvá válhatnak, és adatredundancia alakulhat ki.

▫ A teljesítmény optimalizálásához szükséges normalizálás és indexelés szaktudást igényel.

* **Adatsilók kialakulásának lehetősége**

**OO modell - alapszerkezet**

* **Objektumok:** A valós világ entitásait reprezentálják.
* **Osztályok:** Az objektumok szerkezetét és viselkedését írja le.
* **Öröklődés:** Az objektumok szülőosztályoktól örökölhetnek

attribútumokat és metódusokat.

* **Enkapszuláció:** Az adatok és metódusok együtt tárolódnak.
* **Polimorfizmus:** Ugyanaz az interfész, különböző megvalósításokkal.
* **Az OO modell sajátosságai:**

▫ Az objektumokat egyedi azonosítóval (OID) tárolja az adatbázisban.

▫ A kapcsolatok objektumhivatkozásokkal valósulnak meg.

▫ Metódusokat is lehet tárolni és futtatni az adatbázison belül.

▫ Összetett adattípusok natívan támogatottak.

▫ Lekérdezések objektumhivatkozások navigálásán alapulnak.

**OO modell - előnyök**

* Megkönnyíti az objektumorientált programozási nyelvek és az adatbázis-tárolás integrációját.
* Nem szükséges objektum-relációs leképezést (ORM) végezni, mivel az

objektumok közvetlenül tárolhatók és kezelhetők.

* Az objektumok örökölhetik a tulajdonságokat és metódusaikat a szülőosztályokból, ami könnyíti az újrafelhasználhatóságot.
* Hatékonyan kezeli az M:N és a hierarchikus kapcsolatokat is, anélkül,

hogy join műveletekre lenne szükség.

**OO modell - hátrányok**

* Nincsenek univerzális, elfogadott szabványok az OO modellhez, ellentétben a relációs adatbázisokkal (pl. SQL szabvány).
* Az objektumok tárolása és visszakeresése lassabb lehet, mivel az

adatokat és a viselkedést együtt tárolja.

* További feldolgozást igényel a metódusok végrehajtása és az objektumkapcsolatok navigálása, ami teljesítményproblémákhoz vezethet nagyobb adathalmazoknál.

**NoSQL adatmodellek**

* **Kulcs-érték tárolók (key-value stores):**

▫ Egyszerű adatmodell, amely kulcs-érték párokon alapul.

▫ Magas skálázhatóságú és elosztott működésre tervezett.

▫ Felhasználási területek: gyorsítótárazás, munkamenet-kezelés, valós idejű alkalmazások

▫ *Példák:* Redis, Amazon DynamoDB, Riak

* **Dokumentum-orientált adatbázisok (document stores):**

▫ Félig strukturált adatokat tárol (JSON, BSON, XML).

▫ Rugalmas séma, amely lehetővé teszi a dinamikus adatszerkezeteket.

▫ Felhasználási területek: tartalomkezelő rendszerek, felhasználói profilok kezelése, e-kereskedelmi alkalmazások

▫ *Példák:* MongoDB, CouchDB, RavenDB

* **Oszlopcsalád-alapú adatbázisok (column-family stores):**

▫ Olvasási/írási műveletek optimalizálása nagy adatmennyiség esetén.

▫ Az adatokat oszlopokban tárolja sorok helyett, ami javítja az analitikai

lekérdezések teljesítményét.

▫ *Példák:* Apache Cassandra, HBase, Google Bigtable

* **Gráf-adatbázisok (graph databases):**

▫ Összetett kapcsolatok és hálózatok kezelésére tervezték.

▫ Az adatokat csomópontok, élek és tulajdonságok használatával tárolja.

▫ Felhasználási területek: hálózatok elemzése, csalások detektálása, ajánlórendszerek stb.

▫ *Példák:* Neo4j, Amazon Neptune, ArangoDB

* **Memóriában futó adatbázisok (in-memory databases):**

▫ Bármely adatmodellel alkalmazható megoldás.

▫ Minden adatot a RAM-ban tárol, így rendkívül gyors hozzáférést biztosít.

▫ Felhasználási területek: gyorsítótárazás, valós idejű analitika, mesterséges intelligencia alkalmazások

▫ Bizonyos memóriában futó DBMS-ek biztosítják az adatok perzisztálását is, hogy azok pl. áramkimaradás esetén is megmaradjanak.

A megfelelő adatbázis kiválasztása

1. **Az adatok komplexitása**
   * **Strukturált adatok:** Relációs adatbázisok

▫ Olyan alkalmazásokhoz, amelyek meghatározott sémát és strukturált adatokat igényelnek (pl. pénzügyi rendszerek, készletkezelés stb.)

* + **Félig-strukturált/strukturálatlan adatok:** NoSQL adatbázisok

▫ Rugalmas vagy strukturálatlan adatformátumok kezelésére (közösségi

média tartalmak, IoT adatok stb.)

1. **Kapcsolatok**
   * **Egyszerű kapcsolatok:** Relációs adatbázisok
   * **Erősen összekapcsolt adatok:** Gráf-adatbázisok
2. **Üzleti igények**
   * **Tranzakciós integritás:** Relációs adatbázisok

▫ Kritikus alkalmazásokhoz, amelyek erős konzisztenciát és ACID tulajdonságokat igényelnek (banki rendszerek, vállalatirányítási rendszerek).

* + **Skálázhatóság és teljesítmény:** NoSQL adatbázisok

▫ Nagy méretű vagy magas terhelésű feladatokhoz, ahol az elsődleges cél a horizontális skálázás (valós idejű webszolgáltatások, analitikai rendszerek).

▫ Megjegyzés: A modern relációs adatbázisok is kínálnak elosztott

architektúrákat, amelyek biztosítják a skálázhatóságot.

**Metaadat**

* A **metaadatok** "adatok az adatokról".

▫ Míg az *adatok* az eseményeket, objektumokat és kapcsolatokat írják le, a

*metaadatok* kontextust biztosítanak az adatokhoz.

* A **metaadatkezelés** magában foglalja a metaadatok létrehozásához,

tárolásához, integrálásához és ellenőrzéséhez szükséges folyamatokat, hogy biztosítsa azok pontosságát és hasznosságát.

* Analógia: könyvtári katalógus

▫ Képzeljünk el egy könyvtárat katalógusrendszer nélkül - a megfelelő könyv megtalálása nehéz és időigényes lenne.

▫ A könyvtári katalógus (azaz a „metaadat”) strukturált információkat nyújt

a könyvekről, segítve a felhasználóknak a könyvek hatékony megtalálását.

▫ Hasonlóképpen, a metaadatok segítenek a digitális adatok szervezésében és kezelésében egy szervezeten belül.

**Miért fontosak?**

* A metaadatok javítják a szervezetek adatkezelését és döntéshozatalát. Az előnyök közé tartoznak:

▫ *Gyorsabb információkeresés*: Segít az elemzőknek gyorsan megtalálni

a releváns adatokat.

▫ *Alacsonyabb képzési költségek:* Dokumentálja az adatdefiníciókat, forrásokat és előzményeket, csökkentve a személyzet képzési idejét.

▫ *Jobb együttműködés:* Áthidalja az üzleti felhasználók és a fejlesztők

közötti szakadékot.

▫ *Gyorsabb rendszerfejlesztés:* A fejlesztési életciklus rövidítésével csökken a piacra kerülési idő.

▫ *A felesleges adatok eltávolítása:* Megakadályozza az elavult vagy

helytelen információk használatát.

**Metaadatok – fogalmak és tevékenységek**

* A metaadatok adják a „könyvtári katalógust” a kezelt adatkörnyezetben.
* Az adatok (a tartalom) leíró címkéi vagy kontextusa.
* Megmutatja az üzleti és technikai felhasználóknak, hogy hol találják

meg a megfelelő információkat az adattárakban.

* Válaszokat ad olyan kérdésekre, mint:

▫ Honnan származnak az adatok?

▫ Hogyan került oda?

▫ Milyen átalakításokat végeztek rajtuk?

▫ Mi az adatok minőségi szintje?

▫ ...

* Információt nyújt arról, hogy mit jelentenek ténylegesen az adatok, és hogyan kell értelmezni azokat.

**Metaadat definíció**

* A **metaadatok** olyan információk, amelyek leírják az adatok szerkezetét, szabályait és kapcsolatait egy szervezeten belül.
* **Információt ad a(z)**

▫ fizikai és logikai adatszerkezetekről,

▫ technikai és üzleti folyamatokról,

▫ adatokra vonatkozó szabályokról és megszorításokról.

* **A metaadatok főbb szerepei:**

▫ Leírja az adatvagyont (adatbázisok, adatelemek és adatmodellek).

▫ Meghatározza az üzleti folyamatokat, az alkalmazott rendszereket,

a szoftverkódot és a technológiai infrastruktúrát is.

▫ Megadja az adatok és fogalmak közötti kapcsolatokat.

**A metaadatok főbb típusai**

* **Üzleti metaadat:** Az adatokat az üzleti kontextus és a felhasználás szempontjából határozza meg, a szervezeti célok alapján.

▫ *Példák:* fogalommeghatározások, adatminőségi mérőszámok,

szabályozási megfelelés részletei

* **Leíró metaadat:** Információkat nyújt az adatvagyonról a feltárás, azonosítás és megértés megkönnyítése érdekében.

▫ *Példák:* címek, szerzők, létrehozás dátuma, kulcsszavak,

összefoglalók

* **Adminisztratív metaadat:** Az adatgazdálkodás támogatására szolgál (hozzáférési jogok, felhasználási irányelvek, adatutak stb. kezelése).

▫ *Példák:* tulajdonjog, verziókezelés, hozzáférés-szabályozás,

adatmegőrzési irányelvek

* **Technikai metaadat:** Az adatok formátumának, feldolgozásának és megszorításainak leírása az informatikai rendszerekben.

▫ *Példák:* adattípusok, kódolási szabványok, átalakítási szabályok,

tárolási részletek

* **Szerkezeti metaadat:** Leírja, hogy az adatok hogyan vannak szervezve, tárolva és összekapcsolva egy rendszeren belül.

▫ *Példák:* adatbázis sémák, fájlformátumok, adatmodellek, táblák

közötti kapcsolatok

* **Operatív metaadat:** Az adatok feldolgozására és a folyamatokban való felhasználására vonatkozó, futásidejű információk.

▫ *Példák:* ETL-naplók, végrehajtási időbélyegek, feldolgozási állapot,

adatfrissítési ütemezések

**Metaadatok kezelése**

A metaadatok kezelése több területen is megjelenik:

* **Üzlet és irányítás:** Az üzleti definíciók, szabályok és irányítási

irányelvek a konzisztencia és a megfelelés biztosítása érdekében.

* **Adatszerkezet és adatminőség:** Az adatok modellezéséhez,

integrációjához, átalakításához és az adatminőség menedzsmentjéhez.

* **Rendszerirányítás:** Az informatikai infrastruktúra, rendszertervezés

és rendszerbiztonság kezeléséhez.

* **Folyamat- és tartalomkezelés:** Az üzleti- és munkafolyamatok dokumentálásához és a tartalom kezeléséhez.
* **Architektúra és integráció:** A metaadattárakkal és a metaadatok

származásával foglalkozik az integráció biztosításához.

**Metaadatok létrehozása és forrásai**

* Metaadatokat minden adatkezelési tevékenység során hoznak létre és használnak fel.

Alapvető kontextust biztosítanak az adatok azonosításához, elemzéséhez

és irányításához.

* **Metaadatok létrehozása és azonosítása**

▫ Az üzleti metaadatok felhasználói interakciók, definíciók és az adatok

elemzése során jönnek létre.

▫ A metaadatok különböző szinteken léteznek:

🞄 *Aggregált szint:* Átfogó, összevont jellemzőket tartalmaz, például tématerületeket és közös tulajdonságokat.

🞄 *Részletes szint:* Pl. az adatbázis oszlopainak tulajdonságai, kód feloldási értékek, műszaki specifikációk stb.

* **A metaadatok forrása**

▫ *Elsődleges források:* A szervezeten belüli bármely (megnevezett) entitás, beleértve az adatbázisokat, üzleti szabályokat, adatmodelleket és alkalmazásokat.

▫ *Másodlagos források:* Más metaadattárak, amelyekhez integrációs eszközökön vagy köztes szoftvereken keresztül lehet hozzáférni.

* A metaadatok hatékony kezelése és a kapcsolódó metaadatforrások közötti hatékony navigáció kritikus fontosságú az adatmenedzsment, a konzisztencia és a használhatóság szempontjából.

**Metaadat-követelmények megadása**

* **A metaadatokra vonatkozó követelmények célja:**

▫ A metaadatkörnyezet szükségességének meghatározása.

▫ A hatókör és a prioritások kijelölése.

▫ Az érintettek tájékoztatása és a kommunikáció elősegítése.

▫ Az eszközök értékelésének, implementálásának, modellezésének és

a kapcsolódó szolgáltatások irányítása.

* **A metaadat-követelmények forrásai:**

▫ Üzleti és technikai felhasználóktól származnak:

🞄 A szerepkörök és felelősségek elemzése.

🞄 A felhasználók előtt álló kihívások azonosítása.

🞄 A kiválasztott személyek információs igényeinek megértése.

**Üzleti felhasználói követelmények**

* Az üzleti felhasználóknak pontosan dokumentált és könnyen elérhető adatokra van szükségük a döntéshozatal és a működési hatékonyság támogatásához.
* Főbb követelmények:

▫ *Egyértelmű adatértelmezés:* A metaadatoknak kontextust kell

biztosítaniuk az operatív és analitikai rendszerek számára.

▫ *Bizalom az adatokban:* Javítja a jelentésekbe, dashboard-okba és

elemzésekbe vetett bizalmat.

▫ *Információhozzáférés:* Támogatja a jelentések készítését, lekérdezéseket és ad hoc elemzéseket.

▫ *Konzisztens terminológia:* Szabványosított üzleti fogalmak alkalmazása.

* *Az adatirányítás* biztosítja a metaadatok kezelésének konzisztenciáját, minőségét és megfelelőségét.

**Technikai felhasználói követelmények**

* A technikai felhasználóknak részletes metaadatokra van szükségük az adatstruktúrák, az adatfeldolgozás és az integráció hatékony kezeléséhez.
* Főbb területek:

▫ *Adatfeldolgozás és átbocsátóképesség:* Pl. a napi adatmennyiség és a feldolgozási idő kezelése.

▫ *Metaadatok és adatáramlás:* A források, célrendszerek,

transzformációk és az architektúra (logikai és fizikai) megértése.

▫ *Integráció és szabványok:* A nem szabványos metaadatok és az adatleképezések kezelése.

* Kik használják a technikai metaadatokat?

▫ adatbázis-adminisztrátorok (DBA-k), adatadminisztrátorok (DA-k), fejlesztők, architektek stb.

**Metaadat - architektúra**

**Főbb rétegek:**

1. **Létrehozás/származtatás:** Metaadatok kinyerése különböző

forrásokból.

1. **Integráció:** Metaadatok összevonása különböző rendszerekből.
2. **Tár (repozitórium):** Egy vagy több tároló, ami központilag vagy elosztott módon kezeli a metaadatokat.
3. **Kiszolgálás:** Metaadatok hozzáférésének biztosítása felhasználók és rendszerek számára.
4. **Felhasználás:** Az adatirányítás, az analitika és az operatív folyamatok támogatása.
5. **Ellenőrzés és kezelés:** A metaadatok konzisztenciájának, biztonságának és életciklusának biztosítása.

**Központosított metaadat - architektúra**

* **Egyetlen metaadattár** tárolja a különböző forrásokból származó élő metaadatok másolatait. Ideális, ha a konzisztencia és az egységesség a legfontosabb, de több erőforrást igényel.
* **Előnyök**

▫ Magasabb rendelkezésre állás, mivel független a forrásrendszerektől

▫ Stabil struktúra, mert nem befolyásolják a külső rendszerek formátumai

▫ Forrásnál tárolt, jobb minőségű metaadatok

* **Limitációk**

▫ Komplex replikációt és folyamatos frissítéseket igényel

▫ Nagy karbantartási igény, erőforrás-igényes kezelés

▫ További modulokra vagy middleware-re lehet szükség a kezeléshez

▫ Költségesebb a metaadatok validálása és a karbantartása

**Elosztott metaadat - architektúra**

* **Valós idejű megközelítés**, ahol a metaadatok közvetlenül a forrásokból kerülnek lekérdezésre. Egy hozzáférési pont dolgozza fel a lekérdezéseket middleware protokollok segítségével (pl. Object Request Broker).
* **Előnyök**

▫ A metaadatok naprakészek és érvényesek maradnak

▫ Kevesebb kötegelt feldolgozásra és a replikációra van szükség

▫ A lekérdezések végrehajtására összpontosít

* **Limitációk**

▫ A rendszerek közötti konzisztencia nincs kikényszerítve

▫ A lekérdezések sikertelenek lesznek, ha a forrásrendszer nem érhető el

**Metaadattárak**

* Egy **metaadattár (metaadat-repozitórium)** fizikai táblákból áll, amelyeket jellemzően relációs adatbázisban tárolnak, és amelyek

különböző interfészeket (hozzáférési módokat) valósítanak meg.

* Általános kialakításúnak és nagymértékben integráltnak kell lennie az alkalmazkodóképesség biztosítása érdekében.
* *Példa:* üzleti metaadatok változása
  + Kezdetben: A vásárlót úgy határozzák meg, mint "bárki, aki egy

üzletben vásárolt vagy egy katalógusból rendelt".

* + Később: A webáruház hozzáadása után a definíció úgy változik, hogy az online vásárlókra is kiterjedjen.
  + A metaadattárak biztosítják ezen változások nyomon követését és

következetes kezelését.

**Metaadattár - típusok**

* Egy jól strukturált metaadattár támogatja az adatok konzisztenciájának fenntartását, az adatirányítást és az adatok elérhetőségét.
* **Címtár (directory):** Az adatok helyét tárolja a szervezeten belül; a

fejlesztők és a haladó felhasználók számára hasznos.

**Fogalomtár (glossary):** Az üzleti fogalmak meghatározása, egy tezaurusz segítségével, amely végigvezeti a felhasználókat az ekvivalencia-, hierarchia- és asszociációs kapcsolatokon.

* **Egyéb metaadat-tárolók:**
  + Speciális listák (pl. forráslisták, interfészlista, kódkészlet-lista)
  + Lexikonok
  + Térbeli/időbeli sémák
  + Adattárak adattárai, üzleti szabály adattárak

**Adatminőség - menedzsment**

* Az **adatminőség-menedzsment (data quality management, DQM)** olyan kritikus folyamat, amely biztosítja a különböző forrásokból származó adatok összevonását, tisztítását és integrálását megbízható és jó minőségű adatkészleteket létrehozásához.
* A DQM nem csak a hibák javítása!
* A DQM megjelenik az adatok teljes életciklusában. Például:

▫ *Adatok létrehozása:* Az adatok helyes bevitelének biztosítása a

forrásnál.

▫ *Adattranszformáció:* Az adatok egységesítése és formázása a konzisztencia érdekében.

▫ *Adattovábbítás és integráció:* A zökkenőmentes adatáramlás

biztosítása a rendszerek között.

**Adatminőség - menedzsment**

**A DQM folyamata (magas szintű áttekintés):**

1. Az adatminőséggel kapcsolatos üzleti igények azonosítása.
2. Mérőszámok meghatározása az adatminőség mérésére, nyomon követésére és ellenőrzésére.
3. A minőségi szabványoknak való megfelelés biztosítása (jelentések, adatgazdálkodási lépések végrehajtása).
4. Az adatfelelős (data steward) értesítése a problémák megoldása és az adatminőség fenntartása érdekében.
5. Az adatminőségi folyamatok finomítása és javítása.

**DQM – fogalmak és tevékenységek**

* Az **adatminőségi elvárások** fő szempontjai:
  + *Üzleti szabályok meghatározása:* Annak egyértelmű dokumentálása, hogy mi minősül "jó minőségű" adatnak a struktúra, a konzisztencia és az integritás szempontjából.
  + *Mérési módszerek:* Mérőszámok megállapítása az adatminőség értékelésére különböző dimenziókban (pl. pontosság, teljesség, időszerűség).
  + *Elfogadhatósági küszöbértékek:* Olyan minimális minőségi szintek meghatározása, amelyeknek az adatoknak meg kell felelniük ahhoz, hogy üzleti felhasználásra érvényesnek tekinthetők legyenek.
* Az adatminőségi elvárások szolgálnak egy strukturált **adatminőségi**

**keretrendszer** felépítésének alapjául.

* Egy adatminőségi keretrendszerhez szükséges:
  + *A követelmények meghatározása:* Az adatok pontosságára, teljességére, konzisztenciájára és megbízhatóságára vonatkozó üzleti elvárások meghatározása.
  + *Ellenőrzési szabályok és nyomon követés megadása:* Érvényesítési szabályok, ellenőrzési mechanizmusok és automatizált ellenőrzések bevezetése az adatminőségi változások nyomon követése érdekében.
  + *Mérés és riportolás:* Az adatminőség értékelésének meghatározása

és annak biztosítása, hogy az megfeleljen az üzleti igényeknek.

**A DQM ciklus**

* A **DQM ciklus** egy strukturált folyamat, amelynek célja az adatminőség meghatározása, kezelése és folyamatos javítása az üzleti követelmények teljesítése érdekében.

1. **Az adatproblémák azonosítása:**

▫ Az üzleti célokat befolyásoló kritikus adatproblémák felismerése.

▫ A legfontosabb adatminőségi dimenziók meghatározása (pl.

pontosság, teljesség, konzisztencia).

▫ Üzleti szabályok megállapítása a jó minőségű adatok biztosításához.

1. **Tervezés:**

▫ Az azonosított adatproblémák körének felmérése.

▫ A gyenge adatminőség üzleti hatásainak meghatározása.

▫ A lehetséges megoldásokat értékelése a problémák kezelésére.

1. **Bevezetés:**

▫ Az adatok profilozása és vizsgálata az inkonzisztenciák azonosítására.

▫ Automatizált felügyeleti folyamatok bevezetése a hibák észlelésére.

▫ Az adatkezelési hibák javítása a forrásnál.

1. **Monitorozás:**

▫ Az adatok minőségének nyomon követése (üzleti szabályok alapján).

▫ A minőségi küszöbértékek meghatározása, az eltérések észlelése.

▫ Az adatminőségi szabványoknak való megfelelés figyelése.

1. **Cselekvés:**

▫ Korrekciós intézkedések az azonosított problémák megoldására.

▫ Az üzleti szabályokat és folyamatokat frissítése a hasonló jövőbeli hibák megelőzése érdekében.

**Adatminőségi dimenziók**

* **Pontosság:** Azt méri, hogy az adatok mennyire jól tükrözik a valós világ entitásait.

▫ *Példa:* Az ügyfél telefonszámának meg kell egyeznie a tárolt számmal.

* **Teljesség:** Biztosítja, hogy minden szükséges attribútumnak legyen értéke, és minden szükséges rekord létezzen.

▫ *Példa:* Egy rendelésnek mindig tartalmaznia kell az ügyfél azonosítóját

és a termék adatait.

* **Konzisztencia:** Biztosítja, hogy az adatok egységesek legyenek a különböző forrásokban, és ne legyenek ellentmondásban önmagukkal.

▫ *Példa:* A vevő címének meg kell egyeznie a számlázási és a szállítási

adatbázisban.

* **Aktualitás:** Az adatok naprakészségét és a közelmúltbeli változások tükrözését méri.

▫ *Példa:* A CRM-rendszerben nem szabad, hogy elavult legyen az ügyfél

utolsó vásárlási dátuma.

* **Részletesség:** Az üzleti igények alapján értékeli az adatértékek részletességének szintjét.

▫ *Példa:* A termék árát tizedesjegy pontossággal kell tárolni.

* **Adatvédelem:** Biztosítja az adatokhoz való hozzáférés ellenőrzését és a biztonsági irányelvek betartását.

▫ *Példa:* Az egészségügyi nyilvántartásokhoz csak az arra jogosult

személyzet férhet hozzá.

* **Hivatkozási integritás:** Biztosítja a rekordok közötti kapcsolatok érvényességét.

▫ *Példa:* A számlának hivatkoznia kell egy, az adatbázisban már létező

ügyfél azonosítóra.

* **Időszerűség:** Meghatározza, hogy az adatoknak milyen gyorsan kell rendelkezésre állniuk és hozzáférhetővé válniuk.

▫ *Példa:* A pénzügyi tranzakciós adatokat valós időben kell frissíteni a

csalások felderítéséhez.

* **Egyediség:** Biztosítja, hogy minden rekord csak egyszer jelenjen meg az adatok között.

▫ *Példa:* Egy ügyfélnek nem lehetnek duplikált bejegyzései a rendszerben.

* **Érvényesség:** Biztosítja, hogy az adatok megfeleljenek az előre meghatározott formátumoknak és üzleti szabályoknak.

**Adatprofilozás**

* **Az adatprofilozás** segíti a hibák és ellentmondások felderítését, a teljesség értékelését és az üzleti szabályoknak való megfelelést.
* **Statisztikai elemzés:**

▫ A legfontosabb adatjellemzők mérése (pl.: min, max, átlag, szórás).

▫ Segít azonosítani a kiugró értékeket, hiányzó értékeket stb.

* **Adatkapcsolatok feltárása:**

▫ Függőségek, duplikációk és inkonzisztencia feltárása az adatokban.

▫ Biztosítja a hivatkozási integritást a kapcsolódó adatok, táblák között.

* **Üzleti szabályok validálása:**

▫ Annak ellenőrzése, hogy az adatok megfelelnek-e az üzleti szabályoknak.

**Felbontás és szabványosítás**

* Az **felbontás (parsing)** során az adatok elemzése és komponensekre bontása történik, hogy megfeleljenek egy megadott mintázatnak.

▫ Nyers adatok vizsgálata és a releváns részek kinyerése.

▫ Szabálymotorok használata az érvényes és érvénytelen értékek azonosítására.

▫ Segít a strukturálatlan adatelemek átrendezésében és javításában.

* A **szabványosítás (standardization)** biztosítja, hogy az elemzett adatok

az üzleti szabályoknak megfelelő, konzisztens formátumot kövessenek.

▫ Az adatok szabványos formátumba alakítása az elemzést követően.

▫ Az eltérő vagy érvénytelen adatok átalakítása elfogadható formátumra.

▫ Növeli az adatok pontosságát, konzisztenciáját és használhatóságát a különböző rendszerekben.

* **Miért fontos a felbontás és a szabványosítás?**

▫ Megszünteti a többértelműséget és következetlenségeket a bevitt

adatokban.

▫ Elősegíti a különböző rendszerek közötti integrációt.

▫ Biztosítja az adatkezelési irányelveknek való megfelelést.

▫ Csökkenti a hibákat az adatelemzés és a riportok készítése során.

* *Példa:*

▫ Felbontás: +36 (52) 512-900 / 12345

országkód: 36, körzetszám: 52, állomásszám: 512900, mellék: 12345

▫ Szabványosítás: +36(52)512900-12345, 06.52.512900.12345,

52 512 512 900 12345 stb. → +36-52-521-900/12345

**Azonosság egyeztetése**

* **Az azonosság egyeztetése** során kerülnek azonosításra és

összekapcsolásra az azonos entitásra utaló rekordok (inkonzisztens,

duplikált adatelemek).

▫ **Determinisztikus párosítás**: Előre meghatározott szabályokat és pontos mintaillesztést használ a rekordok összekapcsolására.

🞄 Jól strukturált, szabványosított formátumú adatokat igényel.

🞄 *Példa:* TB vagy útlevélszámok egyeztetése a rekordok között.

▫ **Sztochasztikus párosítás**: Statisztikai modelleket használ az egyezés valószínűségének meghatározására.

🞄 Hatékonyabb a strukturálatlan és hiányos adatok esetében.

🞄 *Példa:* Ügyfelek nevének, e-mail címének és telefonszámának

összevetése fuzzy logika segítségével.

**Adatbázis - adminisztráció**

**Mi az adatbázis-adminisztráció?**

* Az adatbázisrendszerek kezelésének, karbantartásának és optimalizálásának

gyakorlata.

* Biztosítja az adatok rendelkezésre állását, biztonságát és teljesítményét.
* Támogatja az üzleti folyamatokat a strukturált adatok hatékony kezelésével.
* Magában foglalja a proaktív felügyeletet és hibaelhárítást is.

**Miért fontos az adatbázis-adminisztráció?**

* Az adatbázisok jellemzően kritikus üzleti információkat tárolnak.
* A hatékony adminisztráció biztosítja az adatok integritását, rendelkezésre állását és biztonságát.
* Javítja az alkalmazások teljesítményét és a felhasználói élményt.
* Minimalizálja az adatvesztéssel és a rendszerhibákkal kapcsolatos kockázatokat.

**Adatbázis - adminisztrátor**

**Adatbázis-adminisztrátor (database administrator, DBA):**

* Az adatbázisok tervezéséért, megvalósításáért és karbantartásáért felelős.
* Biztosítja az adatbázisok hatékony és biztonságos működését.
* Együttműködik az IT csapatokkal, a fejlesztőkkel és az üzleti résztvevőkkel.

**DBA kompetenciák:**

* Az adatbázis-technológiák erős ismerete.
* Az adatmodellezési és normalizálási módszerek ismerete.
* Jártasság a biztonsági mentési, helyreállítási és biztonsági protokollokban.
* Jártasság a teljesítményhangolásban és lekérdezés-optimalizálásban.
* Problémamegoldó és hibaelhárító készség.

**A DBA fő feldatai**

1. **Adatbázis-tervezés és architektúra** – adatbázisok tervezése és strukturálása
2. **Telepítés és konfiguráció** – a DBMS beállítása és üzembe helyezése
3. **Teljesítményhangolás** – az adatbázis sebességének és hatékonyságának optimalizálása
4. **Biztonsági mentés és helyreállítás** – adatvédelem biztosítása meghibásodás esetén
5. **Biztonsági menedzsment** – hozzáférések szabályozása és visszaélések

megelőzése

1. **Monitorozás és karbantartás** – az adatbázisok egészségének és működésének fenntartása
2. **Fejlesztőcsapatok támogatása** – adatbázisokkal kapcsolatos

alkalmazásfejlesztési igények segítése

1. **Frissítések és javítások** – az adatbázis-rendszerek naprakészen tartása

**Az adatbázis – adminisztrátorok típusai**

* **Rendszer DBA:** Az adatbázis fizikai és infrastrukturális aspektusainak kezelése, beleértve a telepítést, konfigurációt, javítást és frissítést.
* **Alkalmazás DBA:** Együttműködik a fejlesztőkkel a hatékony sémák és lekérdezések megtervezésében, biztosítva az alkalmazás és az adatbázis hatékony együttműködését.
* **Teljesítmény DBA:** Az adatbázis szűk keresztmetszeteinek azonosítására és megoldására szakosodott. A lekérdezések optimalizálására, az indexelési stratégiákra és az erőforrás-kezelésre összpontosít.
* **Biztonsági DBA**: Hozzáférés-vezérlési mechanizmusokat vezet be, figyeli a biztonsági irányelveket, és biztosítja a szabályozásoknak való megfelelést az adatok védelme érdekében.
* A szervezet méretétől és összetettségétől függően ezek a DBA-szerepeket

egyetlen személy is elláthatja, vagy megoszthatók egy csapat tagjai között.

**Adatbázis – adminisztráció - kihívások**

* **Elérhetőség és a leállások megelőzése:** A kritikus alkalmazások folyamatos működésének biztosítása.
* **Teljesítményoptimalizálás:** A lekérdezések és indexelés hatékonyságának

fenntartása növekvő terhelés mellett.

* **Biztonság és megfelelőség:** Az adatok védelme a biztonsági incidensek ellen, valamint a szabályozási előírások betartása.
* **Biztonsági mentés és helyreállítás:** Megbízható mentési és helyreállítási

tervek fenntartása.

* **Erőforráskezelés:** A rendszererőforrások elosztása több alkalmazás és felhasználó támogatására.
* **Lépést tartani a technológiai fejlődéssel** – Az adatbázis-rendszerek,

felhőplatformok és automatizáció változásaihoz való alkalmazkodás.

**DBA megközelítések**

* Az adatbázis-adminisztráció különböző megközelítéseket követhet.
* Két elsődleges megközelítés:
  + **Reaktív**
  + **Proaktív**
* A választott megközelítés hatással van az adatbázis teljesítményére,

rendelkezésre állására és biztonságára.

* De: Egyetlen DBA sem lehet 100%-ban reaktív vagy 100%-ban proaktív.
* Szükség van a felügyeletre és a megelőző intézkedésekre, de erős hibaelhárítási képességeket is biztosítani kell.
* A jó DBA-k a proaktív stratégiák alkalmazása mellett reaktív tudással is

rendelkeznek az adatbázis működésének biztosításához.

**Reaktív DBA:**

* Más néven "tűzoltó" megközelítés.
* A problémák kijavítására összpontosít**, miután** azok felmerültek.
  + Hibaelhárítás, vészhelyzeti javítások, adatbázisok helyreállítása.
* Gyakoribb olyan környezetben, ahol a monitorozás korlátozott, vagy ahol nem hajtanak végre rendszeresen proaktív intézkedéseket.

**A reaktív DBA jellemzői:**

* Adatbázis-összeomlásokra és meghibásodásokra reagál.
* Teljesítményproblémákat a felmerülésükkör oldja meg.
* Felhasználói panaszokat és hibákat vizsgálja és javítja.
* A gyors, rövid távú megoldásokra fókuszál inkább, mint a hosszú távú megelőzésre.

**Proaktív DBA:**

* Más néven megelőző megközelítés.
* A problémák elkerülésére összpontosít, mielőtt azok felmerülnének.
  + Monitorozást, automatikus riasztásokat és kapacitástervezést használ.
* Célja a stabilitás, a hatékonyság és az optimalizált adatbázis-műveletek.

**A proaktív DBA jellemzői:**

* Automatizált monitorozó eszközöket alkalmaz.
* Rendszeres állapotellenőrzést és teljesítményhangolást végez.
* Katasztrófa-helyreállítási és biztonsági mentési stratégiákat tervez.
* Együttműködik a fejlesztőkkel a lekérdezések és indexelés optimalizálásában.
* A hosszú távú adatbázis-stabilitásra és hatékonyságra összpontosít.

**A DBA szerepe az SDLC-ben**

1. **Követelmények feltárása:**

Ebben a szakaszban a projekt általános célkitűzései vannak meghatározva, felmérik a megvalósíthatóságot, és meghatározzák az erőforrásigényeket – beleértve az adatbázis-követelményeket is – az alap biztosítása érdekében.

1. **Elemzés:**

Az DBA-k és az érdekeltek együttműködnek a konkrét adatkövetelmények összegyűjtésében és dokumentálásában, tisztázva, hogy az alkalmazás hogyan fogja tárolni, feldolgozni és elérni az információkat.

1. **Tervezés:**

Ezután elkészül az adatbázis sémája és architektúrája, amely felvázolja az entitások közötti kapcsolatokat, az indexelési stratégiákat és a tárolást a teljesítmény, a skálázhatóság és a biztonsági célok elérése érdekében.

1. **Implementáció:**

Az adatbázis felépítése és integrálása az alkalmazási környezetbe. Ez magában foglalja a táblák, nézetek és egyéb objektumok létrehozását a meghatározott séma szerint, valamint ezek összekapcsolását az alkalmazási réteggel.

1. **Tesztelés és integrálás:**

Az adatbázis validáláson megy keresztül, beleértve a funkcionális tesztelést, a teljesítmény kiértékelését és a biztonsági vizsgálatokat, annak érdekében, hogy valós körülmények között is megfeleljen a követelményeknek.

1. **Üzemeltetés és karbantartás:**

Ezután a DBA felügyeli a teljesítményt, javításokat alkalmaz, és gondoskodik a biztonsági mentési és helyreállítási folyamatokról, biztosítva az adatbázis biztonságát, elérhetőségét és hatékonyságát az egész életciklusa alatt.

**Teljesítménymonitorozás és hangolás**

* A **monitorozás** biztosítja az adatbázis optimális működését különböző terhelések alatt, és segít az esetleges problémák azonosításában.
* A **monitorozó eszközök** figyelik a rendszer állapotát, teljesítményét és erőforrás-felhasználását, valós idejű betekintést nyújtva a rendellenességek és szűk keresztmetszetek észleléséhez.
  + Főbb funkciók: lekérdezéselemzés, teljesítményhangolás, hibafigyelés és diagnosztika.
  + *Példák:* Oracle Enterprise Manager, Microsoft SQL Server Profiler, MySQL Performance Schema.
* **Naplóelemzés:** Az ismétlődő problémák, sikertelen tranzakciók és hibák azonosítása a naplók alapján.
* **Lekérdezés-profilozás:** A lassú lekérdezések felderítése, a végrehajtási tervek optimalizálása és a válaszidők csökkentése.
* A **teljesítményhangolás** biztosítja a hatékony működést terhelés alatt.
* **Főbb teljesítménymutatók:** lekérdezési válaszidő, CPU-használat, I/O

műveletek száma, memóriahasználat stb.

* A **teljesítményoptimalizálási technikák** közé tartoznak például:

▫ *SQL lekérdezések optimalizálása:* A lekérdezések egyszerűsítése a végrehajtási idő és az erőforrás-felhasználás csökkentése érdekében.

▫ *Indexelés és particionálás:* A hozzáférés sebességének növelése indexek

létrehozásával és a nagy adathalmazok szegmensekre osztásával.

▫ *Gyorsítótárazás:* A gyakran használt adatok tárolása a memóriában, a lekérdezési idő minimalizálásához.

▫ *Csatlakozás-összevonás:* Az adatbázis-kapcsolatok újrafelhasználása az

újak létrehozásával járó az erőforrások jobb kihasználása érdekében.

**Biztonság és hozzáférés - kezelés**

* Az adatbázisok gyakran tartalmaznak érzékeny (üzleti és ügyfél) adatokat, amelyek magasabb szintű védelmet igényelnek.
* **Felhasználói hozzáférés kezelése:**

▫ A felhasználók csak a szerepükhöz minimálisan szükséges jogosultságokkal rendelkezzenek (legkevesebb jogosultság elve).

▫ Biztonságos hitelesítési mechanizmusokat kell használni (pl. MFA, SSO).

▫ A hozzáférés és módosítások részletes naplózása szükséges a gyanús

tevékenységek felderítése és az azokra való reagálás érdekében.

* **Néhány biztonsági gyakorlat:**

▫ Szerepek használata a hozzáférési jogosultságok hatékony kezeléséhez.

▫ Az adatok titkosítása mind nyugalmi állapotban, mind az átvitel közben.

▫ Rendszeres biztonsági ellenőrzések és sebezhetőségi értékelések elvégzése.

▫ A biztonsági javítások és frissítések rendszeres alkalmazása.

**Biztonsági mentés és helyreállítás**

* Az adatvesztés megelőzése a DBA egyik legfontosabb feladata.
* **A biztonsági mentések típusai:**

▫ *Teljes mentés:* Teljes adatbázis másolat.

▫ *Inkrementális mentés:* Változások az utolsó biztonsági mentés óta.

▫ *Differenciális mentés:* Változások az utolsó *teljes* biztonsági mentés óta.

* **Helyreállítási technikák:**

▫ point-in-time helyreállítás, naplóalapú helyreállítás, replikáció, failover

fürtözés, …

* **A biztonsági mentés és helyreállítás gyakorlatai:**

▫ Automatizált biztonsági mentéseket alkalmazása.

▫ A biztonsági mentések on-site és off-site (felhő, külső tároló) tárolása.

▫ A biztonsági mentések helyreállítási folyamatainak rendszeres tesztelése.

▫ A katasztrófa utáni helyreállítási tervek karbantartása.

**DBA – személyzet mérete**

**Miért fontos a DBA-k száma?**

* A helyes adatbázis-kezelés elengedhetetlen az adatok integritása,

biztonsága és rendelkezésre állása szempontjából.

* Előfordulhat, hogy egy személy nem képes az összes adatbázissal kapcsolatos feladatot és felelősséget kezelni.
  + Túl kevés DBA → Leállásokhoz, gyenge teljesítményhez és biztonsági

rések kockázatához vezethez.

* + Túl sok DBA → A hatékonyság csökkenése és felesleges költségek.
* A megfelelő egyensúly megtalálása biztosítja az adatbázis teljesítményét és költséghatékonyságát, és így az arra épülő alkalmazások megfelelő működését is.

**DBA – munkaterhelési tényezők**

1. **Adatbázisok száma:** több adatbázis több kezelési erőfeszítést is igényel
2. **Adatbázis mérete**: a nagyobb adatbázisok több tárolási, biztonsági,

mentési és teljesítményhangolási feladatot jelentenek

1. **A felhasználók száma:** több egyidejű felhasználó megnövekedett igényt jelent a DBA felé
2. **Alkalmazás összetettsége**: az összetett alkalmazások jellemzően több

adatbázis-hangolást és összetettebb felügyeletet igényelnek

1. **Automatizálás és nyomon követés:** a fejlett monitoring eszközök

használata csökkentheti a kézi munkát, de további szakértelmet is igényel

1. **Biztonsági és megfelelési követelmények:** a szigorú megfelelés (pl.

GDPR, HIPAA) külön biztonsági irányítást igényel

1. **Helyreállítási és biztonság**: a gyakori biztonsági mentések és a magas szintű rendelkezésre állás nagyobb figyelmet igényel a DBA-tól

**DBA – adatbázis-méret és komplexitás**

**Hogyan befolyásolja a méret a DBA feladatait?**

* **Kis adatbázisok (<100 GB) → Kevesebb teljesítmény-optimalizálás,**

**egyszerűbb mentési és biztonsági stratégiák, alapvető indexelés.**

* **Közepes adatbázisok (100 GB – 1 TB) → Összetettebb indexelési stratégiák, lekérdezés-optimalizálás, és rendszeres karbantartás szükséges.**
* **Nagy adatbázisok (>1 TB) → Nagyobb kihívások, például particionálás, tárolóhely-kezelés, elosztott rendszerek, és teljesítményhangolás.**

**Miért nem csak a méret számít?**

* **OLTP → Egy kisebb adatbázis is komoly DBA felügyeletet igényelhet, ha másodpercenként több ezer tranzakció történik.**
* **Big Data rendszerek → Egy nagy adattárházon inkább batch-feldolgozást**

**végeznek, így más típusú optimalizálás és karbantartás szükséges.**

* **Elosztott és replikált rendszerek → Nem feltétlenül nagy az adatbázis mérete, de az összetettség miatt több DBA-munkát igényelhet.**

**DBA – SLA követelmények**

* A **szolgáltatási szint megállapodás (service level agreement, SLA)** meghatározza az elérhetőséget, a válaszidőt és a teljesítménnyel kapcsolatos elvárásokat.
* **Hatása a DBA munkaterhelésére:**
  + **99.9% elérhetőségi SLA** → Proaktív monitorozást és rendkívül gyors reakcióidőt igényel.
  + **24/7 rendelkezésre állás** → Több DBA szakembert vagy műszakos

munkarendet tesz szükségessé.

* + **Nagy teljesítményű SLA-k** → Az indexelésre, a lekérdezések optimalizálásra és a teljesítményhangolásra kell fókuszálni.
  + **Katasztrófa-helyreállítási SLA-k** → Kiterjedt biztonsági mentési

stratégiákat és átfogó failover-tervezést igényel.

**DBA – Hány DBA szükséges?**

**Általános iránymutatások:**

▫ 1 DBA 10-20 alacsony/közepes összetettségű és méretű adatbázisra.

▫ 1 DBA 5 nagyobb összetettségű, kritikus fontosságú adatbázisra.

▫ 100-200 fejlesztőre legalább 1 DBA jusson a fejlesztőcsapatokban.

* **Egyéb megfontolások:**

▫ Felhőalapú vagy helyi adatbázisokat kell kezelni?

▫ Használnak automatizált DBA eszközöket?

▫ Van külön biztonsági DBA, vagy a biztonság az általános DBA- feladatok része?

▫ Mennyire változékony az adatbázis-környezet?

**Adatbázis környezetek**

* A szervezetek többféle környezetet használnak a szoftverfejlesztés és az üzemeltetés során, hogy biztosítsák az alkalmazások stabilitását és megbízhatóságát.
* A három fő környezet:
  + **Fejlesztési környezet ("Dev"):** Kódolásra, új funkciók fejlesztésére és alkalmazások létrehozására szolgál.
  + **Tesztelési környezet ("Test"/"Staging"):** A funkcionalitás,

teljesítmény és hibamentesség ellenőrzésére.

* + **Éles/Termelési környezet ("Prod"):** Az a környezet, ahol az alkalmazást a végfelhasználók valós körülmények között használják.
* A környezetek megfelelő kezelése kulcsfontosságú az alkalmazások

stabilitásának, biztonságának és hatékonyságának fenntartásához.

**A különálló környezetek fontossága:**

* **Megakadályozza a nem szándékos változásokat** – A fejlesztők nem

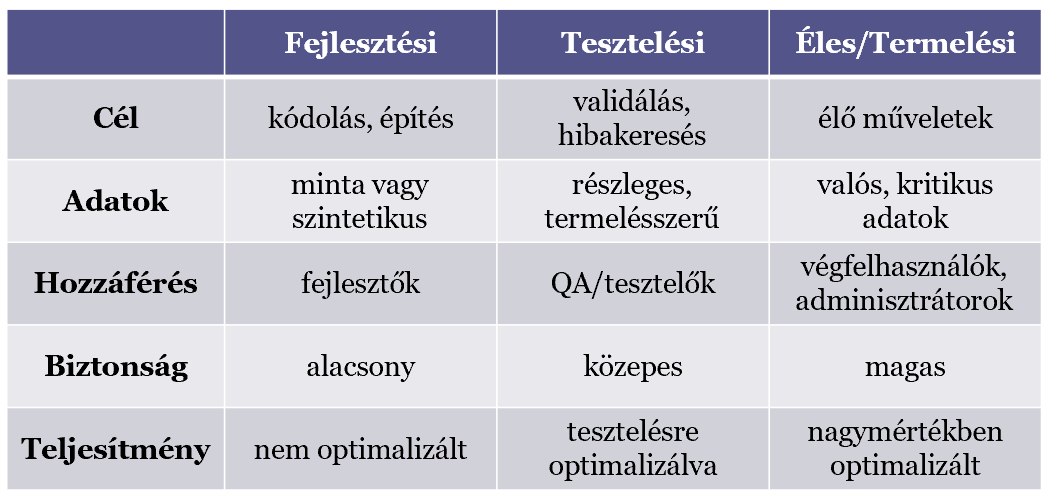
módosíthatják véletlenül az éles termelési adatokat.

* **Fokozza a biztonságot** – Az érzékeny éles adatok védettek maradnak, minimalizálva az illetéktelen hozzáférés és adatvesztés kockázatát.
* **Támogatja a tesztelést és a hibakeresést** – A változtatások

ellenőrizhetők és finomhangolhatók anélkül, hogy a végfelhasználókat

érintenék.

* **Optimalizálja a teljesítményt** – A termelési adatbázisok továbbra is a valós munkaterheléshez igazodnak, elkerülve a fejlesztési és tesztelési folyamatok okozta terhelést.
* **Megkönnyíti a megfelelést** – Segíti a szabályozási előírások betartását azáltal, hogy a tesztelési és éles adatok elkülönülnek.



**Az adatok kezelése különböző környezetekben:**

* **Konzisztencia** – Biztosítani kell, hogy a tesztkörnyezet tükrözze az éles struktúrát, így a fejlesztés és tesztelés eredményei használhatóak és érvényesek lesznek.
* **Adatfrissítési stratégiák** – A tesztadatok relevanciájának fenntartása érdekében rendszeres, ütemezett frissítéseket kell végrehajtani.
* **Verzióellenőrzés** – A séma- és adatbázismódosítások nyomon követése elengedhetetlen a környezetek közötti konzisztencia és kompatibilitás megőrzéséhez.
* **Automatizálási eszközök** – Telepítési és migrációs eszközökkel a frissítések gyorsabbá, biztonságosabbá és hatékonyabbá tehetők.

**Az adatok szinkronizálásának stratégiái (Prod és Test/Dev között):**

* **A termelési adatok részhalmaza:** A teljes adatbázis replikálása helyett csak egy releváns részhalmaz kerül felhasználásra a tesztelési és fejlesztési környezetekben, csökkentve az adatkezelési terhelést.
  + **Anonimizált adatok:** Az érzékeny termelési adatok *nem valós, de értelmes* értékekkel történő helyettesítése, biztosítva az adatvédelem és a megfelelőség fenntartását.
* **Adatbázis pillanatfelvételek (snapshots)** – Időhöz kötött másolatok létrehozása a termelési adatbázisról tesztelési célokra, amelyeket szükség esetén vissza lehet állítani.
* **Szintetikus adatok generálása:** Automatikusan létrehozott tesztadatkészletek alkalmazása, ha valós adatok nem állnak rendelkezésre, különböző eszközök segítségével.

**Adatbiztonság és megfelelőség tesztkörnyezetekben:**

* A tesztkörnyezet gyakran kevésbé biztonságos, mint az éles rendszer, így az

érzékeny termelési adatok védelme kulcsfontosságú.

* Számos szabályozás, például a **GDPR, HIPAA és PCI-DSS**, tiltja a valós ügyféladatok védtelen felhasználását tesztelési célokra.

**Biztonsági és megfelelőségi intézkedések:**

* **Maszkolás és anonimizálás:**

A valós adatok helyettesítése anonimizált vagy generált értékekkel a személyes azonosítás elkerülése érdekében.

* **Feladatok szétválasztása:**

A fejlesztők és tesztelők nem férhetnek hozzá közvetlenül az éles adatokhoz.

* **Hozzáférés-kezelés:**

Csak engedélyezett személyek férhetnek hozzá a tesztadatbázisokhoz, minimalizálva a kockázatokat.

**Biztonsági és megfelelőségi intézkedések (folyt.):**

* **Titkosítás:** Az érzékeny adatok átalakítása egy visszafejthető kódolt

formába, így. Ez megakadályozza az illetéktelen hozzáférést az adatokhoz,

még akkor is, ha azok kiszivárognának.

* **Tokenizálás:** Az érzékeny adatokat helyettesítő azonosító (token) segítségével cseréli le, amelynek nincs visszafejthető kapcsolata az eredeti adattal.
* **Rendszeres ellenőrzések és naplózás:**

Minden adathozzáférés és módosítás nyomon követése, beleértve az audit naplók rendszeres felülvizsgálatát.

* **Megfelelőségi jelentések készítése:**

Az adatkezelési tevékenységekről részletes dokumentációk készítése a

szabályozási követelmények teljesítése és a biztonság fenntartásához.

**Tároláskezelés és adatintegritás**

* Ez az előadás az adattárolási technológiákra fókuszál, elsősorban relációs adatbázis-kezelő rendszerek kontextusában, és az adatbázis- adminisztrátor szemszögéből.
* A következő témákat fedi le:

▫ Hagyományos és korszerű tárolási megoldások: fizikai lemezek, SAN/NAS, felhőalapú megoldások, elosztott rendszerek, …

▫ A tárolási megoldás hatása a teljesítményre, skálázhatóságra,

megbízhatóságra és költségekre.

▫ Adatbázis-specifikus aspektusok: táblaterek, indexelés, tranzakciónaplók, állományok elhelyezése, …

* Alapvető háttérismereteket nyújt a tárolással kapcsolatban a

megalapozott és hatékony döntéshozatalhoz.

**A megfelelő tárolási megoldás kiválasztásának fontossága**

* A DBMS-ek nem minden tárolási technológiát támogatnak.
* A DBA feladata, hogy különböző tárolási megoldásokat kiértékelje, és kiválassza azt, amely a szervezet számára a legmegfelelőbb.
* A döntés során figyelembe veendő főbb szempontok:

▫ *Teljesítmény:* az olvasási és írási műveletek sebessége

▫ *Megbízhatóság:* az adatvesztés és a rendszerleállások kockázatának minimalizálása

▫ *Használhatóság:* az integráció és a kezelés egyszerűsége

▫ *Költségek:* a költségvetés és az elvárt funkciók közötti egyensúly

megtalálása

**Alapvető tárolási technológiák**

* A **merevlemezek (HDD-k)** továbbra is az egyik leggyakrabban alkalmazott, alapértelmezett tárolási megoldást jelentik.

▫ Nagy adatmennyiség esetén költséghatékony megoldást kínálnak.

▫ A mechanikus alkatrészek miatt sérülékenyebbek.

* **RAID (Redundant Array of Independent Disks, független lemezek redundáns tömbje):**

▫ Adatredundanciával csökkenti a hardverhibák hatását.

▫ Különböző konfigurációk a teljesítmény és a hibatűrés optimalizálására:

🞄 RAID 0 – csíkozás, RAID 1 – tükrözés, RAID 5 – elosztott paritás,

RAID 6 – kettős paritás

▫ Egyes RAID szintek kombinálhatók vagy egymásba ágyazhatók, például:

RAID 1+0 („tükrözött csíkozás”), RAID 0+1 („csíkozott tükrözés”).

* A **solid-state drive-ok (SSD-k)** mozgó alkatrészek híján gyorsabb elérést biztosítanak, kisebb késleltetéssel, ellenállóbbak a fizikai behatásokkal szemben, energiatakarékosabbak és halkabbak is.
* NVMe (Non-Volatile Memory Express):

▫ PCIe-alapú flash tárolókhoz optimalizált protokoll

▫ lényegesen nagyobb IOPS-t (I/O művelet másodpercenként) kínál

* Előnyök adatbázisok esetében:

▫ jobb véletlenszerű I/O teljesítményt nyújt

▫ csökkenti a szűk keresztmetszeteket a nagy terhelésű, online

tranzakciófeldolgozás (OLTP) során

* **SAN (Storage Area Network, tárolóhálózat):**

▫ Speciális hálózat, amely tárolóeszközök összekapcsolására szolgál.

▫ A szerverek a SAN-t egy logikai lemezként látják.

▫ Gyakran tárolószerverek rack-jeként kerül implementálásra.

* **NAS (Network-Attached Storage, hálózatra kapcsolt tároló):**

▫ Egy megosztott tárolóeszköz, amely hálózaton keresztül érhető el.

▫ A felhasználók számára hálózati meghajtóként jelenik meg.

▫ Könnyen integrálható, azonban a hálózati forgalmat megosztja más adatforgalommal, így teljesítménybeli kompromisszumokkal járhat nagy terhelés esetén.

**Megfontolandó szempontok**

* SAN/NAS rendszerek választásakor fontos mérlegelni a következőket:

▫ skálázhatóság (bővíthetőség),

▫ áteresztőképesség (adatátviteli sebesség),

▫ hálózati jellemzők (sávszélesség, késleltetés).

* A RAID tömbök egyaránt alkalmazhatók hagyományos lemezeken

alapuló tárolásnál, valamint SAN/NAS környezetekben is.

* Mindig mérlegelni kell a költségeket és az adminisztratív terheket a teljesítményre vonatkozó és a megbízhatósági követelményekkel szemben.

**Az I/O műveletek hatása a teljesítményre**

* Az adatbázis teljesítménye közvetlenül összefügg az I/O műveletek

sebességével.

* Az adatok felolvasása a háttértárolóról sokkal lassabb, mint a memóriából történő olvasás.
* A modern tárolórendszerek gyorsítótárakat (cache) alkalmaznak, amelyek előzetes adatbetöltéssel optimalizálják az adatok elérését, ezáltal csökkentik az I/O műveletekből adódó késleltetést.

**Felhőalapú tárolási technológiák**

**Nyilvános felhő**

* A tárolást külső szolgáltató végzi (pl. Amazon S3).
* Rugalmas, „pay-as-you-go” (használat alapú) díjszabás jellemzi.
* Az elosztott infrastruktúrának köszönhetően egyszerűsített katasztrófa- helyreállítási megoldásokat kínál.

**Privát felhő**

* A tárolás helyben, vagy dedikált erőforrásokon történik.
* Nagyobb kontrollt biztosít az adatok felett.
* Elősegíti az adatkezelési és megfelelőségi előírások betartását.

**Hibrid és multi-cloud megoldások**

* Hibrid felhő: nyilvános és privát felhő kombinációja
* Multi-cloud: több felhőszolgáltató egyidejű használata
* *Rugalmasság* – alkalmazások a legalkalmasabb környezetben futtathatók
* *Költséghatékonyság* – optimális erőforrás-kihasználás, alacsonyabb

költségek

* *Teljesítményoptimalizálás* – skálázhatóság és hatékony terheléselosztás
* *Szolgáltatói függetlenség* – vendor lock-in csökkentése

**Elosztott tárolás**

**Elosztott fájlrendszerek (például: HDFS, Ceph)**

* Az adatokat több csomópont között osztják el és replikálják, így nincs egy „single point of failure”.
* Párhuzamos adatelérést támogat, ezáltal jelentősen javítja a teljesítményt nagy adatmennyiségek esetén.
* Magas rendelkezésre állást és hibatűrést biztosít.

**Skálázható (scale-out) architektúra**

* A rendszer horizontálisan skálázható további csomópontok hozzáadásával.
* Ezáltal hatékonyan növelhető mind a tárolókapacitás, mind az adatátviteli sebesség.
* Különösen alkalmas:

▫ Big data elemzésekhez

▫ Adattó (data lake) architektúrákhoz

▫ Olyan munkafolyamatokhoz, amelyek párhuzamos feldolgozást és magas rendelkezésre állást igényelnek

**Konténeres környezetek és tárolás**

**Konténeres telepítés:**

* A konténerek az alkalmazásokat függőségeikkel együtt tárolják →

hordozhatóak és konzisztens működést biztosítanak minden környezetben.

* A konténerképek könnyen kezelhetőek és változatlanok (immutable).
* Segítik az egyes szolgáltatások skálázását és a telepítési ütközések csökkentését.

**Container Storage Interface (CSI):**

* Szabványos interfész, amely lehetővé teszi, hogy az orchesztrátorok (pl. Kubernetes) különböző tárolószolgáltatókhoz kapcsolódjanak.
* Dinamikus perzisztens tárhelyek kiosztását teszi lehetővé felhőből vagy helyi háttértárról.
* Moduláris és hordozható tárolókezelést biztosít.
* Támogatott rendszerek: pl. AWS EBS, Azure Disk, Ceph.

**Efemer (átmeneti) kötetek**

* Ideiglenes tárolók, amelyek csak a konténer élettartamáig léteznek.
* A bennük tárolt adatok elvesznek a konténer leállásakor vagy újraindításakor.
* Nem alkalmasak adatbázisokhoz vagy tartós adatokhoz.
* Ideális felhasználás:

▫ Gyorsítótárazás

▫ Munkamenet-adatok

▫ Futtatás közbeni ideiglenes fájlok

**Tárolási technológiák**

**Ajánlások tárolórendszerekhez**

* A modern tárolóplatformok többsége API-alapú kezelést, automatikus

erőforrás-kiosztást és dinamikus skálázást kínál → egyszerűbb

üzemeltetés, kevesebb manuális beavatkozás

* Felhőalapú vagy elosztott tárolás esetén kulcsfontosságú a titkosítás, hozzáférés-szabályozás, szabályozási megfelelőség (pl. GDPR) kiértékelése.
* Minden új tárolómegoldás esetén javasolt teljesítményteszteket végezni a késleltetés, átviteli sebesség és megbízhatóság kiértékeléséhez, főként, ha adatbázisokat szolgál ki.

**Tároláskezelés RDBMS-ekben**

**Táblaterek**

* Definíció: A táblatér olyan logikai struktúra, amely az adatbázis-

objektumokat (pl. táblák, indexek) fizikai állományokhoz rendeli hozzá.

* Célja:

▫ Kapcsolódó adatbázis-objektumok csoportosítása,

▫ A tárolási hely meghatározása, és ezzel a teljesítmény optimalizálása.

* Tipikus kialakítás:

▫ Egy táblatérhez egy vagy több állomány tartozik.

▫ Az adatbázis-adminisztrátor határozza meg, hogy az egyes

objektumokat (pl. táblák és indexek) mely táblaterekbe helyezi.

**Adatlapok és adatrekordok**

* **Adatlap (blokk)**

▫ Az adatlap az I/O alapegysége egy DBMS-ben.

▫ Mérete rögzített, például 4 KB vagy 8 KB – ez rendszerfüggő.

* **Adatrekord (sor)**

▫ Egy sor adatait, és metaadatokat tárol (fejléc, offset tábla).

▫ Egy lapon több rekord is elfér, kivéve ha egy rekord túl nagy méretű.

* **Adatlap szerkezete:**

▫ *fejléc:* lap metaadatai (pl. adatlap-azonosító, szabad terület mutatók)

▫ *adatrekordok:* a tábla sorainak tényleges adatai

▫ *offset tábla (opcionális):* a változó hosszúságú mezők gyors elérését segíti

**Az index fogalma**

* Szerepe: Lehetővé teszi a gyorsabb keresést azáltal, hogy mutatókat

tárol az adatsorokra bizonyos oszlop(ok) alapján.

* Felépítése:

▫ Az indexek általában kiegyensúlyozott fára épülnek, (B-fa, B⁺-fa).

▫ Az index kulcs-érték párokból áll:

🞄 a kulcs az indexelt oszlop(oka)t tartalmazza,

🞄 az érték pedig mutató a megfelelő adatrekord helyére.

* Jelentősen csökkenti a keresési időt, különösen nagy méretű táblákban.
* Javítja a lekérdezések teljesítményét, de tárolási többletigénnyel és magasabb karbantartási költséggel jár (pl. frissítések, beszúrások esetén).

**Indexrekord**

* Tartalma:

▫ *Fejléc:* néhány bájtnyi adminisztratív metaadat, amely az indexrekord szerkezetét és elrendezését írja le

▫ *Indexkulcs-értékek:* az indexelt oszlop(ok) aktuális értékei, a definíció sorrendjében

▫ *Adatlapmutató:* egy mutató, amely az indexelt tábla fizikai adatrekordjának helyét azonosítja

▫ *Opcionális offset tábla:* akkor szükséges, ha a rekord változó hosszúságú mezőket is tartalmaz

* Elhelyezés: Az indexrekordokat külön erre a célra fenntartott adatlapokon tárolja a rendszer jellemzően.

**Kiterjesztés (extent)**

* Definíció: A kiterjesztés a lemezen egymás után következő adatblokkokból

álló egység, amelyet a DBMS egyszerre foglal le.

* Célja:

▫ A gyakori kis méretű foglalások elkerülése, ami töredezettséghez vezetne.

▫ Lehetővé teszi, hogy az adatbázis-objektumok fokozatosan, nagyobb

egységekben növekedjenek.

* Elsődleges és másodlagos kiterjesztés:

▫ *Elsődleges kiterjesztés:* az első lefoglalt kiterjesztés, amikor egy új adatbázis-objektum létrejön

▫ *Másodlagos kiterjesztés(ek):* az objektum növekedésével utólag lefoglalt további kiterjesztések

**Állománykiterjesztés**

* Erre akkor van szükség, amikor egy táblatérhez tartozó állomány eléri a

számára lefoglalt maximális méretet, de további területre van szükség.

* Megvalósítás:

▫ A DBA új fizikai állományt adhat hozzá a táblatérhez, vagy növelheti a meglévő állomány méretét.

▫ Egyes DBMS-ek támogatják az automatikus állománynövelést

(*autoextend*), ha ez be van állítva.

* Fontos a megfelelő méretküszöbök meghatározása, hogy elkerüljük a lemez betelését, ami leállásokhoz vezethet.

**Adatok módosítása a táblákban**

* **INSERT (beszúrás):**

▫ A DBMS keres egy adatlapot, ahol van elegendő szabad hely.

▫ Ha nincs megfelelő lap, akkor újat foglal (akár egy új kiterjesztésből).

* **DELETE (törlés):**

▫ Szabad terület keletkezik az adatlapon, ami töredezettséghez vezethet.

▫ Időnként újraszervezés szükséges a tárhasználat optimalizálásához.

* **UPDATE (módosítás):**

▫ Ha a rekord mérete nő, akkor előfordulhat, hogy új adatlapon kell

elhelyezni, vagy szét kell osztani több adatlapra, mutatók segítségével.

▫ Ha a rekord mérete csökken, az ismét szabad területet eredményez.

**A táblaméret kiszámítása**

* **Rekordméret:** Az összes oszlop típusának mérete, valamint a

metaadatok mérete (rekordfejléc, offset tábla).

▫ Változó hosszúságú mezők esetén a rekordméret is változhat.

* **Sorok száma adatlaponként:** A szabad adatlap-területet osztva a rekordmérettel. A fejlécét és a töredezettséget is figyelembe kell venni.
* **A teljes táblaméret meghatározása:**

▫ Becsüljük meg a tábla várható sorainak számát.

▫ Számítsuk ki, hogy hány sor fér el egy adatlapon.

▫ Szorozzuk meg a lap méretét a szükséges lapok számával.

▫ A későbbi bővítéshez szükséges helyet is be kell tervezni.

**Az indexméret kiszámítása**

* **Indexrekord mérete:** Az indexrekord a kulcsoszlop(oka)t és

metaadatokat (pl. rekordfejléc, mutató, opcionális offsetek) tartalmazza.

* **A táblamérethez hasonló megközelítés:**

▫ Meg kell állapítani, hogy egy adatlapon hány indexrekord fér el.

▫ Az így kapott értéket szorozzuk meg az indexrekordok számával

(egyoszlopos indexnél ez a táblában lévő sorok száma).

▫ Vegyük figyelembe a B-fa szerkezet (szintek, elágazási tényező) többlethely-igényét, a töredezettséget és a jövőbeli növekedéshez szükséges területet is.

**További megfontolások**

* **Autoextend és monitorozás**

▫ Az automatikus állománykiterjesztés megkönnyíti a tárolási

kapacitás bővítését, azonban elrejtheti a mélyebb problémákat, ha nincs a lemezhasználat megfelelően monitorozva.

* **Indexek karbantartása**

▫ Az indexek javítják a lekérdezések sebességét, ugyanakkor

lassíthatják az írási műveleteket a folyamatos frissítési többlet miatt.

▫ Az indexek rendszeres újraépítése vagy újraszervezése szükséges,

különösen nagy mennyiségű adatmódosítás után, hogy megőrizzük

azok hatékonyságát.

**További megfontolások**

* **Táblatér-stratégiák**

▫ Külön táblaterekbe helyezhetjük a magas tranzakciószámú táblákat, az archivált adatokat és az indexeket.

▫ Ez javítja az I/O műveletek hatékonyságát, egyszerűsíti a biztonsági mentési feladatokat, és átláthatóbb lesz a tárhely kezelése is.

**Adatbázisnaplók**

* A **tranzakciónapló** egy dedikált állomány vagy állománycsoport, amely az adatbázison végrehajtott összes módosítást rögzíti.
* A változásokat naplóbejegyzések formájában tárolja, megjelölve, hogy

melyik tranzakció, mit, és mikor hajtott végre.

* **ACID-támogatás:**

▫ *Atomosság (atomicity):* A naplózott módosításokat teljes egészében

végrehajthatjuk (commit), vagy visszaállíthatjuk (rollback).

▫ *Konzisztencia (consistency):* A naplózott módosítások segítenek az adatbázis konzisztens állapotának helyreállításában.

▫ *Elkülönítés (isolation):* Támogatja az egyidejű tranzakciók kezelését.

▫ *Tartósság (durability):* A naplóbejegyzések alapján az elvégzett módosítások szükség esetén újra végrehajthatók.

* A **tranzakció** olyan végrehajtási egység, amely egy vagy több DML utasításból (INSERT, UPDATE, DELETE) áll.

▫ Explicit (pl. BEGIN TRANSACTION) vagy implicit módon kezdődik.

▫ A tranzakció COMMIT (véglegesítés), vagy ROLLBACK (módosítások visszavonása) utasítással fejeződik be.

* **Naplóbejegyzés-típusok:**

▫ A tranzakció kezdő- és végpontját jelölő bejegyzések.

▫ Melyik sorban történt változás,

▫ Milyen értékek változtak meg (régi és új adat).

▫ COMMIT és ROLLBACK események.

▫ Egyéb rendszerhez kapcsolódó események (például ellenőrzőpontok).

* **Aktív napló (active log):**

▫ Nyitott naplóállomány, amibe a DBMS a tranzakciók bejegyzéseit írja.

▫ Kritikus fontosságú a helyreállításhoz, ezért kiemelt védelmet igényel.

▫ A megbízhatóság miatt jellemzően tükrözve van (redundáns tárolás).

* **Naplóváltás (log switching):**

▫ Ha az aktív napló megtelik/elér egy adott méretet, a DBMS átvált egy új

naplóállományra.

▫ A korábbi naplóállomány inaktívvá válik, és archiválásra kerül(het).

* **Archivált napló (archived log):**

▫ Történeti naplóállomány, amely a már lezárt tranzakciókat tartalmazza.

▫ Biztonsági másolatként menthetők másodlagos tárolóra.

▫ Point-in-time helyreállításhoz használatóak.

**Ellenőrzőpontok:**

* Az ellenőrzőpontok olyan bejegyzések, amelyek azt jelzik, hogy az addig

történt tranzakciók teljes mértékben ki lettek írva a lemezre.

* A DBMS-nek az ellenőrzőpont előtti naplóbejegyzésekre már nincs szüksége egy normál összeomlás utáni helyreállításhoz.

**Naplókarbantartás:**

* A régebbi, inaktív naplóállományok biztonságosan archiválhatóak, ha már nincs rájuk szükség a biztonsági mentésekhez vagy elemzésekhez.
* A naplóállományok rendszeres karbantartásával („log housekeeping”) elkerülhető, hogy elfogyjon az allokált tárhely, és kezelhető szinten

tarthatóak a helyreállítási idők.

**Magas rendelkezésre állás biztosítása:**

* Használjunk több napló példányt (tükrözés) vagy RAID technológiát az adatbázisnaplók tárolásához.
* A tranzakciónapló állományait más fizikai tárolóeszközön helyezzük el, mint az adatok állományait, így csökkenthető az I/O versengés.

**Monitorozás és riasztások:**

* Valós időben kövessük nyomon a naplóállományok tárhelyhasználatát.
* Állítsunk be automatikus riasztásokat, hogy megelőzzük a naplóállományok

váratlan betelését.

**Archiválási stratégia:**

* Rendszeresen mozgassuk az inaktív naplóállományokat biztonságos, akár külső tárolóra (offsite storage).
* Igazítsuk az archiválást a szervezet általános biztonsági mentési és katasztrófa-helyreállítási irányelveihez.

**A tárolás skálázása**

* Folyamatos növekedés: több rekord, nagyobb fájlok, …
* **Rendszeres monitorozás:**

▫ Használjuk a DBMS beépített eszközeit vagy külső szoftvereket.

▫ Figyeljük a tárhely alakulását → előzzük meg a kritikus helyhiányt.

* **Tárhely visszanyerése:**

▫ Nem használt objektumok törlése a tárhely felszabadításához.

▫ A régi, szükségtelen adatok archiválása vagy végleges törlése.

* **Tárhely bővítése:**

▫ A táblaterek és adatfájlok méretének növelése (ALTER).

▫ Szükség esetén új fizikai lemezek hozzáadása, RAID-kötetek bővítése.

**További megfontolások**

* Szezonális vagy eseményekhez kötődő (pl. ünnepi értékesítés,

marketingkampányok) csúcsidőszakokban az adatbázis gyorsan nőhet.

* Ennek kezeléséhez elengedhetetlen a proaktív tervezés, mivel a hirtelen tárhelyhiány üzemzavarokat és kieséseket okozhat.
* A modern adatbázis-kezelő rendszerek általában két irányban is

skálázhatóak:

▫ horizontálisan – további csomópontok hozzáadásával

▫ vertikálisan – nagyobb lemezek/RAID-tömbök stb. alkalmazásával

**Horizontális és rugalmas skálázás**

**Adatparticionálás (sharding)**

* A nagy adathalmazokat több adatbázis-példányra vagy szerverre osztja
* Elosztja az I/O- és számítási terhelést → jobb teljesítmény és válaszidő
* Különösen hasznos nagy forgalmú alkalmazások esetén
* Csökkenti a szűk keresztmetszeteket, javítja a skálázhatóságot

**Rugalmas felhőalapú erőforrások**

* Dinamikus tárhely- és számítási kapacitás (pl. AWS EBS, Azure Disks)
* Igény szerinti skálázás → optimális erőforrás-kihasználás
* Pay-as-you-go modell: költséghatékony, elkerülhető a túlméretezés
* Jól illeszkedik a változó terhelésű munkafolyamatokhoz

**Particionálás és adatrétegezés**

**Particionálás:**

* Nagyméretű táblák (és indexek) kisebb egységekre bontása: tartomány,

lista vagy hash alapú módszerekkel.

* Javítja a lekérdezések teljesítményét az adatmennyiség csökkentésével.
* Egyszerűsíti a karbantartást, a biztonsági mentéseket és a történeti adatok archiválását.

**Adatrétegezés:**

* A ritkán használt („hideg”) adatokat automatikusan olcsóbb, lassabb tárolókra mozgatja (pl. HDD, objektumtároló).
* A gyakran elért („forró”) adatokat nagy teljesítményű tárolón tartja (pl.

NVMe SSD).

* Költséghatékony működést biztosít, a teljesítmény csorbítása nélkül.

**Archiválási irányelvek**

**Archiválási irányelvek:**

* Határozzuk meg a megőrzési szabályokat a történeti vagy ritkán használt

adatokra.

* Az elavult adatokat költséghatékony tárolókra migráljuk (pl. felhőalapú objektumtárolók, szalagos tárolók, „hideg” HDD-k).
* Csökkenti a tárolási költségeket, miközben szükség esetén elérhető marad

az adat.

**Életciklus-kezelő eszközök:**

* Automatizálják az adatok mozgatását a forró, meleg és hideg tárolási szintek között.
* Biztosítják az adatmegőrzési szabályok (pl. GDPR) betartását.
* Támogatják az adatkezelést: nyomon követik a hozzáférést, felhasználást és

a lejáratot.

**Tömörítés és deduplikáció**

**Tömörítés:**

* A modern adatbázisok támogatják a sor- vagy oldal szintű tömörítést.
* Transzparens a felhasználók felé, csökkenti a tárhelyigényt és az I/O műveletek számát.
* Nagy adatkészleteknél javítja a teljesítményt.
* Hátrány: nagyobb CPU-terhelés I/O közben a tömörítés miatt.

**Deduplikáció:**

* Azonos, ismétlődő adatok felismerése és eltávolítása, jellemzően a tárolási vagy mentési rétegben.
* Különösen hatékony mentések, virtuális gépek és naplók esetében.
* Javítja a tárhatékonyságot, csökkenti a mentési időt és méretet.

**A tárolás skálázása**

**Kapacitástervezés:**

* Azonosítsuk előre a jövőbeli tárolási igényeket az eddigi használati trendek alapján.
* Hangoljuk össze a tárolókapacitás bővítését az adatmegőrzési szabályokkal és az üzleti növekedés ütemével.

**Rendszeres karbantartás:**

* Rendszeresen szervezzük újra a táblákat és indexeket, ezzel csökkentve a

töredezettséget.

* Folyamatosan monitorozzuk a teljesítményt, hogy proaktívan megelőzzük a szűk keresztmetszetek kialakulását.

**Automatizálás és riasztások:**

* Állítsunk be riasztásokat kritikus küszöbértékek átlépése esetére (például

amikor a táblatér kihasználtsága eléri a 80%-ot).

* Automatizáljuk a rendszeres karbantartási feladatokat (például archiválás, adatok törlése).

-----------------------------------------------------------------------------------------------------------------

**Bevezetés az adatbázis-biztonságba**

* Az **adatbázis-biztonság** technikai, jogi és szabályozási intézkedések összessége, amelyek az adatbázis-rendszerekben tárolt adatokat védik.
* Ez nem csupán technikai probléma – a biztonság magában foglalja:

▫ a jogszabályok betartását, a felhasználói adatok etikus kezelését, szabályzatok megalkotását és betartatását, …

* Az adatvédelmi incidensek pénzügyi veszteséghez, hírnévromláshoz,

jogi következményekhez stb. vezethetnek.

* A cél az alábbiak megelőzése:

▫ illetéktelen hozzáférés,

▫ adatokkal történő visszaélés,

▫ információk kiszivárgása.

**Adatbázis-biztonsági fenyegetések**

* **Az integritás elvesztése**

▫ Akkor következik be, amikor az adatokat helytelenül módosítják, akár

véletlenül, akár szándékosan.

* **Az elérhetőség elvesztése**

▫ A jogosult felhasználók nem tudnak hozzáférni az adatokhoz.

▫ Okozhatja rendszerhiba, szolgáltatásmegtagadási (DoS) támadás,

zsarolóvírus, stb.

* **A bizalmasság elvesztése**

▫ Érzékeny adatok jogosulatlan nyilvánosságra hozatalára utal.

▫ Gyakori okai: belső fenyegetések, gyenge hozzáférés-vezérlés, SQL-

befecskendezéses támadások, stb.

**Adatbázisok hálózati környezetben**

* Az adatbázisok egy egymással összekapcsolt komponensekből álló hálózatba vannak integrálva. Ilyen komponensek például:

▫ alkalmazások,

▫ szolgáltatások és API-k, amik adatokat tesznek elérhetővé,

▫ tűzfalak a hálózati forgalom védelmére,

▫ monitorozó eszközök naplózáshoz és észleléshez.

* Az adatbázis csupán egy elem egy nagyobb információs rendszerben. Ha bármely komponens sérülékeny, az adatbázis is veszélybe kerülhet.
* *Példa:* Ha egy webalkalmazás sebezhető, egy támadó elérheti az alatta

lévő adatbázist, még akkor is, ha az adatbázis jól van konfigurálva.

**Védelmi intézkedések**

* **Hozzáférés-kezelés:** Biztosítja, hogy csak jogosult felhasználók férhessenek hozzá vagy módosíthassák az adatokat. Magában foglalja a felhasználó hitelesítését (pl. felhasználónév/jelszó) és jogosultságokat.
* **Következtetés-védelem:** Megakadályozza, hogy támadók statisztikai lekérdezések vagy metaadatok segítségével következtessenek

információkra. Fontos pl. az összesített jelentések védelméhez.

* **Adatáramlás-szabályozás:** Megakadályozza az illetéktelen

adatáramlást felhasználók vagy rendszerek között. Biztosítja, hogy érzékeny adatok ne szivárogjanak alacsonyabb biztonságú környezetbe.

* **Titkosítás:** Az adatokat közvetlenül értelmezhetetlen formára alakítja algoritmusok és kulcsok segítségével. Védi az adatokat továbbítás közben és nyugalmi állapotban (az adatbázisban vagy a biztonsági mentésekben).

**Felhasználói fiókok kezelése**

* **Legkisebb jogosultság elve:** Csak a szükséges hozzáférést kapja meg a felhasználó.

▫ A szerepköröket és jogosultságokat rendszeresen felül kell vizsgálni,

különösen szerepkörváltás után.

* **Felhasználói fiókok kezelése:**

▫ A felhasználót egyedi felhasználónévvel és jelszóval kell hitelesíteni.

▫ Biztonságos jelszóházirendet (hossz, szimbólumok, lejárat/forgatás, zárolás) kell alkalmazni.

🞄 Nincs alapértelmezett jogosultság bejelentkezéskor.

▫ További információk: alapértelmezett DB, nyelv, elérhetőségek.

▫ A nem aktív vagy árva fiókok figyelése és mihamarabbi letiltása.

**Hitelesítési adatok kezelése**

* A **hitelesítési adatok kezelése** biztosítja, hogy a felhasználók valóban azok, akiknek mondják magukat, és hitelesítettek maradnak a hozzáférés során.
* Főbb gyakorlatok:

▫ *Többfaktoros hitelesítés (MFA):* Valami, amit tudunk (jelszó), amink van (mobilalkalmazás), vagy amik vagyunk (biometria) kombinációja.

▫ *Jelszóbiztonság:* Erős jelszavak, rendszeres cserék, összetettségi szabályok és zárolás sikertelen próbálkozások után.

▫ *Munkamenet-kezelés:* Inaktivitás után automatikusan zárja le a munkameneteket, és kötelező kijelentkezést ír elő.

▫ *Hitelesítési adatok frissítése:* Időszakos jelszócserék előírása, és az adatok visszavonása a felhasználó távozásakor vagy a szerepkörének változásakor.

**Jogosultságok kezelése**

* DCL utasítások (GRANT és REVOKE) szolgálnak az elérés szabályozására.
* Központosított jogosultságkezelés megvalósítása javasolt.
* A **GRANT … WITH GRANT OPTION** lehetővé teszi a delegálást, de bonyolítja a jogosultságok kezelését.
* **Adatbázis-jogosultságok típusai:**

▫ *Tábla jogosultságok:* SELECT, INSERT, UPDATE, DELETE utasítások

táblákon/nézeteken

▫ *Objektum jogosultságok:* táblák, indexek, triggerek létrehozása/törlése

▫ *Program jogosultságok:* tárolt eljárások biztonságos futtatása

▫ *Rendszerszintű jogosultságok:* DBMS-szintű műveletek végrehajtása

(pl. leállítás)

* **Jogosultságok kiosztása a PUBLIC felhasználónak**

▫ A PUBLIC egy speciális pszeudo-felhasználó, amely minden

bejelentkezett (hitelesített) felhasználót jelent.

▫ Hasznos lehet alapértelmezett jogosultságok kiosztására (pl. nyilvános nézetek vagy alprogramok elérése).

▫ Nem érzékeny objektumokra, csak olvasható műveletekre stb.

korlátozzuk.

* **Láncolt (kaszkádolt) visszavonás**

▫ Ha egy felhasználó a WITH GRANT OPTION-nel kap jogosultságot, azt továbbadhatja másoknak is.

▫ Ha az eredeti jogosultságot visszavonják, a felhasználó által kiosztott jogosultságok is visszavonásra kerülnek.

**Auditálás, mint biztonsági intézkedés**

* Az **auditálás** visszatartó és észlelő mechanizmusként is működik.
* Fontos események naplózása: DML, DDL, hitelesítési események, …
* Tevékenységnaplók: Ki, mikor, milyen adatot ért el, és milyen műveletet hajtott végre?
* Az audit naplók lehetővé teszik az alábbiak észlelését:

▫ illetéktelen hozzáférési kísérletek, jogosultságokkal való visszaélés,

gyanús vagy szokatlan viselkedési minták

* Kiemelten fontos a belső fenyegetések azonosításában és kezelésében.
* Kerüljük a túlzott auditálást – koncentráljunk az érzékeny műveletekre

és a nagy kockázatú felhasználókra.

* Meg kell találni a teljesítmény és az átláthatóság egyensúlyát.

**Szerepkörök és egyéb intézkedések**

* **Szerepkörök az adatbázisokban:**

▫ A szerepkörök jogosultságokat csoportosítanak az egyszerűbb

felhasználókezelés érdekében.

▫ Szerepköröket rendeljünk a felhasználókhoz a jogosultságok helyett.

▫ Beépített szerepkörök, pl.: SYSADM, DBA, DBMAINT, OPER

* **Nézetek és tárolt eljárások használata a biztonság érdekében**

▫ A nézetek korlátozzák az adatok elérését oszlop (vertikális) vagy sor (horizontális) szinten.

▫ A tárolt eljárások kiválthatják a közvetlen táblaelérést.

▫ Fokozza az ellenőrizhetőséget és csökkenti az érzékeny objektumok

kitettségét.

**Érzékeny adatok típusai**

* **Természetüknél fogva érzékeny adatok:** olyan adatok, amelyek

önmagukban is bizalmasak, a kontextustól függetlenül.

▫ *Példa:* egészségügyi adatok, banki hitelesítési adatok, osztályzatok, …

* **Deklaráltan érzékeny adatok:** olyan adatok, amelyeket a szervezet

érzékenynek nyilvánított, még ha nem is természetüknél fogva azok.

▫ *Példa:* Egy vállalat belső irányelveket, teljesítményértékeléseket,

dokumentációkat nyilváníthat érzékenynek.

* **Érzékenység kontextusa:**

▫ Az adat önmagában nem érzékeny, de más adatokkal kombinálva azzá válhat (pl. név + születési dátum).

▫ Korábbi közzétételek vagy adatforrások befolyásolhatják a jelenlegi érzékenységet.

**A közzétételt befolyásoló tényezők**

* **Adatok elérhetősége**

▫ Csak akkor tegyünk elérhetővé adatokat, ha érvényesek és teljesek.

▫ Kerüljük a közzétételét frissítés, ellenőrzés vagy javítás közben.

* **Hozzáférés elfogadhatósága**

▫ Hozzáférés csak jogosult felhasználóknak adható, szerepkör és kontextus alapján.

▫ Korlátozzuk a hozzáférést engedélyezett időpontokra (pl. projekt

futamideje, munkaidő) és biztonságos/megbízható eszközökre.

* **Hitelesség biztosítása**

▫ Győződjünk meg az adatot kérő személy vagy rendszer kilétéről.

▫ Használjunk hitelesítési módszereket, jelszavak, biometrikus adatok vagy digitális tanúsítványok.

**Hozzáférés-vezérlés áttekintése**

* **Diszkrecionális hozzáférés-vezérlés**

▫ A felhasználók a tulajdonos belátása szerint kaphatnak jogosultságot.

▫ *Példa:* Aki létrehoz egy táblát, eldöntheti, ki férhet még hozzá.

* **Kötelező hozzáférés-vezérlés**

▫ A hozzáférés biztonsági címkék alapján (pl. bizalmas, titkos stb.) történik.

▫ A rendszer maga érvényesíti a szabályokat; nincs felhasználói felülírás.

* **Szerepkör-alapú hozzáférés-vezérlés**

▫ A jogosultságok szerepkörökhöz, nem egyénekhez kötöttek.

▫ Könnyebb kezelni a hozzáférést nagy szervezetek esetében.

▫ *Példa:* Az *Elemző* hozzáférhet nézetekhez, a *DBA* sémákat kezelhet.

* A gyakorlatban a rendszerek gyakran kombinálják a három megközelítést.

**Diszkrecionális hozzáférés-vezérlés**

* A **DAC (Discretionary Access Control)** lehetővé teszi, hogy az adatok tulajdonosai szabályozzák a hozzáférést az erőforrásaikhoz.
* Rugalmas és felhasználóbarát, de kockázatos lehet felügyelet nélkül.
* **Fiókszintű jogosultságok:** adatbázis-szintű műveletek szabályozása

▫ *Példák:* CREATE, DROP, ALTER jogosultságok

* **Relációszintű jogosultságok:** konkrét táblákhoz való hozzáférés

szabályozása

▫ *Példák:* SELECT, INSERT, UPDATE, DELETE jogosultságok

* **Nézetek használata:**

▫ A nézetek virtuális táblák, amelyek korlátozhatják a felhasználók

hozzáférését bizonyos oszlopokhoz vagy sorokhoz.

▫ *Példa:* Egy nézet elrejtheti a fizetés értékét az alkalmazottak adataiból.

**DAC – jogosultságkezelés**

* **Jogosultságok adása:**

▫ *Példa:* **GRANT SELECT, UPDATE ON Employees TO Analyst;**

▫ Az *Analyst* jogosultságot kap az *Employees* tábla olvasására és frissítésére.

* **Jogosultságok visszavonása:**

▫ *Példa:* **REVOKE UPDATE ON Employees FROM Analyst;**

▫ Eltávolítja az *Analyst* frissítési jogosultságát.

* **Horizontális terjedés:** Hány felhasználónak adható tovább egy

jogosultság.

* **Vertikális terjedés:** Hány szint mélyre terjedhet a jogosultság delegálása.
* Megjegyzés: Ha nem szabályozzák szigorúan, a jogosultságok túljuthatnak a

szándékolt felhasználókon.

▫ *Példa:* **GRANT SELECT ON Employees TO A WITH GRANT OPTION;**

**Kötelező hozzáférés-vezérlés**

* A **MAC (Mandatory Access Control)** biztonsági címkék alapján szabályozza a hozzáférést.
* A felhasználókhoz és adatokhoz biztonsági címkéket rendel.
* **Fő jellemzők:**

▫ Rendszer által érvényesített szabályok – az adminisztrátorok határozhatják meg a hozzáférési szabályokat

▫ A felhasználók nem módosíthatják a jogosultságokat, és nem oszthatják meg azokat

▫ Címkék: *szigorúan titkos, titkos, bizalmas, korlátozott, nem minősített/nyilvános*

* A MAC központosított, következetes és szigorú kontrollt biztosít az adatokhoz való hozzáférés felett.
* **Biztonsági szabályok:**

▫ *Egyszerű biztonsági tulajdonság ("nem olvashat fentről")*

🞄 A felhasználó csak a saját szintjén vagy alacsonyabb szinten lévő

adatokat olvashat.

▫ *Csillag (\*) tulajdonság ("nem írhat lefelé")*

🞄 Csak a saját szintjére vagy magasabb szintre írhat adatokat. Ez

megakadályozza az adatszivárgást kevésbé biztonságos területekre.

* A MAC szigorú hozzáférés-vezérlési modell, amelyet magas biztonsági szintű környezetekben használnak.
* A Bell-LaPadula modell egy diszkrecionális biztonsági tulajdonságot is

definiál, amely egy hozzáférési mátrixot használ DAC megadásához.

**Szerepkör-alapú hozzáférés-vezérlés**

* Az **RBAC (Role-based Access Control)** egy olyan biztonsági modell, ahol a jogosultságokat szerepkörökhöz, nem pedig egyénekhez rendelik:

▫ A szerepkörök munkaköröket/funkciókat írnak le (pl. hallgató, oktató)

▫ A jogosultságokat ezekhez a szerepkörökhöz rendelik

▫ A felhasználókhoz szerepköröket rendelnek

▫ A felhasználók a szerepkörökön keresztül öröklik a jogosultságokat

* Széles körben használják szervezetekben, mivel egyszerűsíti a jogosultságkezelést, különösen sok hasonló feladatú felhasználó esetén.
* *Példa:*

▫ Hallgató szerepkör: jegyek megtekintése, beadandók beküldése

▫ Oktató szerepkör: anyagok feltöltése, beadandók értékelése

* **Az RBAC előnyei:**

▫ Egyszerűsíti a jogosultságkezelést

▫ Illeszkedik a szervezeti struktúrához

▫ Fokozza a biztonságot és a megfelelőséget

▫ Jól skálázható nagy rendszerekben

* **Az RBAC típusai:**

▫ *Egyszerű RBAC:* alap szerepkör-jogosultság összerendelés

▫ *Hierarchikus RBAC:* szerepkörök örökölhetnek másoktól

▫ *Korlátozott RBAC:* biztosítja a feladatok szétválasztását,

megakadályozza az összeférhetetlenséget (egy felhasználó ne tudja végrehajtani egy érzékeny folyamat összes lépését)

**„Finomszemcsés” hozzáférés-vezérlés**

* A „finomszemcsés” hozzáférés-vezérlés lehetővé teszi a hozzáférés szabályozását különböző szinteken, egészen az egyes rekordokig.
* **Címkealapú biztonság és sorszintű hozzáférés-vezérlés:**

▫ Biztonsági címkék hozzárendelése az adatbázis táblák egyes soraihoz

▫ Rekordszintű szabályozást nyújt (nem csak tábla- vagy oszlopszintű)

▫ *Példa:* egy vezető csak a saját csapatának fizetési adatait láthatja

* **XML hozzáférés-vezérlés:**

▫ Az XML-t széles körben használják strukturált adatcserére

▫ Hozzáférési szabályok alkalmazhatók egy XML dokumentum

meghatározott elemeire vagy szakaszaira

* **Tartalom-alapú hozzáférés-vezérlés (CBAC):**

▫ A hozzáférési döntések az adatok tényleges tartalma alapján születnek,

nem csak a felhasználó személyazonossága alapján.

▫ Ez a hozzáférés-vezérlési módszer más módszerekkel kombinálva is használható.

▫ Lehetővé teszi a finomszemcsés vezérlést a következők alapján:

🞄 Az adat érzékenysége – pl. egészségügyi vagy pénzügyi adatok

🞄 Szerepkör és kontextus – beosztás, helyszín, hozzáférés ideje

🞄 Üzleti szabályok – szervezet-specifikus irányelvek és folyamatok

▫ Pontosan szabályozható vele a hozzáférés; ideális olyan rendszerekhez,

ahol különböző felhasználók eltérő nézeteket igényelnek.

**A DBA szerepe a biztonságban**

* **Főbb biztonsággal kapcsolatos feladatai:**

▫ Felhasználói fiókok létrehozása és kezelése, szerepkörök és jogosultságok

hozzárendelése a szervezeti igények alapján.

▫ Az adatbázis objektumokhoz való hozzáférés szabályozása (pl. SELECT, UPDATE, DELETE) a legkisebb jogosultság elve alapján.

▫ Osztályozási szintek hozzárendelése adatokhoz és felhasználókhoz MAC-

alapú rendszerekben.

▫ Strukturált eljárások bevezetése a szerepkörök és jogosultságok kezelésére.

▫ Monitorozó és auditáló eszközök használata a tevékenységek nyomon

követésére, szabályzatok érvényesítésére és gyanús viselkedés észlelésére.

* A DBA biztonsági felelősként is működik – egyensúlyt tartva a hozzáférés,

elszámoltathatóság és megfelelés között.

**Biztonság a DBMS-en túl**

* A modern alkalmazások általában bárhonnan elérhetőek, nem csak biztonságos hálózatokon belül:

▫ A felhasználók különböző eszközökről, helyekről és szerepkörökből

férhetnek hozzá az alkalmazásokhoz.

▫ A biztonsági intézkedéseknek alkalmazkodniuk kell a kontextushoz és a kockázati szinthez.

* Az olyan rendszerek, mint az e-kereskedelmi vagy banki alkalmazások, érzékeny pénzügyi és személyes adatokat kezelnek.

▫ Az adatszivárgás jogi, pénzügyi és hírnévvesztési következményekkel is járhat.

* Olyan alkalmazásszintű szabályokra van szükség, amelyek figyelembe veszik a felhasználói viselkedést, az eszköz megbízhatóságát és a valós idejű fenyegetéselemzést.

**SQL befecskendezés**

* Az **SQL befecskendezés (SQL injection, SQLi)** egy gyakori és veszélyes támadási forma adatbázis-rendszerek ellen.
* Az SQLi támadások célja, hogy egy megfelelően előkészített bemenettel

módosítsa a támadó az alkalmazás által végrehajtott SQL lekérdezéseket:

▫ kijátssza a bejelentkezési/hitelesítési mechanizmusokat

▫ adatokat módosítson vagy töröljön

▫ jogosulatlan hozzáférést szerezzen érzékeny információkhoz

▫ akár az egész szerver felett átvegye az irányítást

* Ezek a támadások az olyan gyengén védett lekérdezéseket használják ki,

amelyek közvetlenül kombinálják a felhasználói bemenetet SQL kóddal.

**SQL befecskendezés – támadási módok**

* **SQL manipuláció**

▫ Kártékony bemenet beillesztése a lekérdezésekbe

▫ A meglévő lekérdezéseket módosítja például OR, UNION vagy – (megjegyzés) hozzáfűzésével

▫ *Példa:* **SELECT \* FROM users WHERE name='' OR '1'='1’;**

* **Függvényhívás befecskendezés**

▫ Nem biztonságos SQL függvényeket vagy tárolt eljárásokat hív meg

▫ *Példa:* **EXEC xp\_cmdshell 'dir'; -- MS SQL Server**

* **Hitelesítés megkerülése**

▫ A bejelentkezési űrlapok gyakori célpontok – a támadók manipulált bemenetet használnak, hogy azonosító adatok nélkül lépjenek be

**SQL befecskendezés - következmények**

A támadók

* érzékeny vagy bizalmas adatokat szerezhetnek meg, ami jogi és

pénzügyi következményekkel járhat;

* Módosíthatnak vagy törölhetnek kritikus adatokat, ezzel veszélyeztetve az adatintegritást;
* sérülékenységek kihasználásával magasabb szintű hozzáférést

szerezhetnek, akár teljes kontrollt is az adatbázis felett.

* egyes környezetekben a támadók operációs rendszer szintű parancsokat is végrehajthatnak.
* manipulált lekérdezésekkel kimeríthetik a rendszer erőforrásait,

teljesítménycsökkenést vagy elérhetetlenséget okozva

(szolgáltatásmegtagadási (Denial of Service, DoS) támadás).

Az adatok titkosításának alapjai

A **titkosítás** az adatokat olyan

formába (kriptoszöveg) alakítja, amely csak egy megfelelő kulccsal fejthető vissza.

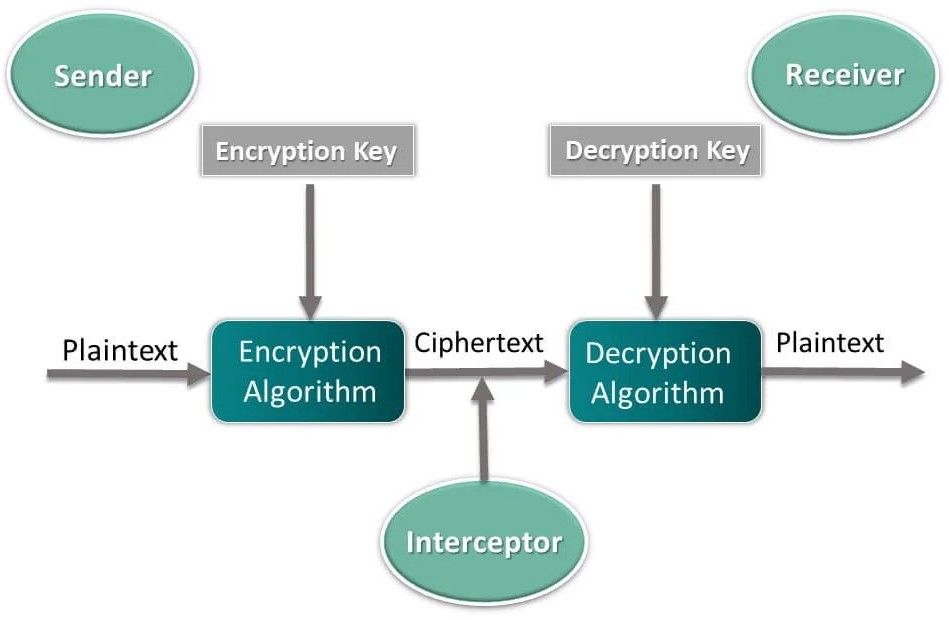
* Az adatok matematikai

algoritmusok és kulcsok

használatával titkosíthatók.

* A visszafejtés során egy kulcs segítségével nyerhető vissza az eredeti adat.
* A titkosítás lehet szimmetrikus

vagy aszimmetrikus.



* **Advanced Encryption Standard (AES):**

▫ Jelenlegi ipari szabvány a *szimmetrikus titkosításhoz*

▫ Gyors és kellően erős biztonságot nyújt

▫ Felhasználás: HTTPS, VPN, Wi-Fi (WPA2/3), adatbázis-titkosítás stb.

▫ Probléma: a kulcsot biztonságosan kell megosztani a felek között

* **RSA (Rivest–Shamir–Adleman):**

▫ Széles körben alkalmazott *aszimmetrikus titkosítási* algoritmus

▫ Nyilvános kulcs/titkos kulcs párt használ a biztonságos kulcscseréhez

▫ Felhasználás: TLS/SSL, e-mail titkosítás, digitális aláírás stb.

▫ Lassabb, mint az AES, de jól illeszkedik a nem megbízható hálózatokon keresztül történő kommunikáció megvalósításához

**Nyilvános kulcsú infrastruktúra (PKI)**

A **PKI (Public Key Infrastructure)** egy bizalmi keretrendszert biztosít a biztonságos kommunikációhoz, lehetővé téve a titkosítás, hitelesítés és a digitális aláírás megvalósítását nyílt hálózatokon.

1. **Kulcsgenerálás:**

▫ A felhasználók és rendszerek egy nyilvános/titkos kulcspárt generálnak.

▫ A titkos kulcs biztonságosan tárolandó és nem osztható meg.

▫ A nyilvános kulcs mások számára elérhetővé válik.

1. **Kulcsok terjesztése PKI-n keresztül:**

▫ A nyilvános kulcsok jellemzően tanúsítványokkal kerülnek megosztásra, amelyeket egy megbízható tanúsítványkiadó (CA) ad ki.

▫ A PKI biztosítja a kulcstulajdonos hitelességét, és véd a megszemélyesítéses támadások ellen.

**Nyilvános kulcsú infrastruktúra (PKI)**

1. **Titkosítás és üzenetküldés:**

▫ A küldő a címzett nyilvános kulcsával titkosítja az üzenetet.

▫ Csak a címzett, aki birtokában van a megfelelő titkos kulcsnak, tudja

visszafejteni az üzenetet.

1. **Üzenet fogadása és visszafejtés:**

▫ A címzett a saját titkos kulcsával visszafejti az üzenetet és visszanyeri

az eredeti tartalmat.

* + Megjegyzés: A tanúsítványok tartalmazzák a tulajdonos azonosítóját, nyilvános kulcsát, az érvényesség idejét és a tanúsítványkiadó (CA)

digitális aláírását.

**A védelem egyensúlya**

* **Az adatbázis-biztonság céljai:**

▫ Az adatok pontosságának és részletességének maximalizálása, hogy a

tárolt adatok megfelelőek maradjanak elemzéshez és döntéshozatalhoz.

▫ A bizalmas adatok jogosulatlan kiszivárgásának megelőzésére.

* **Az egyensúly elérésének stratégiái:**

▫ *Ellenőrzött adatkibocsátás:* Az érzékeny adatokhoz való hozzáférés korlátozása. Az adatok anonimizált vagy összesített formában való biztosítása a kockázat minimalizálása érdekében.

▫ *A szabályzatok felülvizsgálata:* A védelmét célzó intézkedések rendszeres értékelése és frissítése. A szabályzatok igazítása az új kockázatokhoz és technológiai változásokhoz.

▫ A felhasználhatóság és védelem közötti kompromisszum meghatározása.

**Biztonsági mentés és helyreállítás**

* Egy DBMS eleinte jellemzően hibátlanul működik, de idővel a környezete megváltozhat. Például:

▫ nő a felhasználók száma és a kezelendő adatmennyiség,

▫ új alkalmazások kerülnek bevezetésre,

▫ frissül a hardver- vagy a hálózati infrastruktúra, ...

* Ezek a változások olyan hibákat eredményezhetnek, amelyek veszélyeztetik

a *rendelkezésre állást*, az *integritást* és a *használhatóságot*.

* A DBA-nak gyorsan és hatékonyan kell reagálnia az ilyen problémákra.
* A cél egy átfogó biztonsági mentési és helyreállítási terv kialakítása és karbantartása.

▫ Figyelembe kell venni az összes lehetséges hibaforrást.

**Biztonsági mentés (backup):**

* Az adatbázisfájlok teljes vagy részleges másolatainak létrehozása az

adatvesztés elleni védekezés és a gyors helyreállítás érdekében.

**Visszaállítás (restoration):**

* A mentett fájlok visszamásolása a mentési tárolóról az DBMS háttértájára.
* Ebben a fázisban még nem történik a tranzakciós változások alkalmazása.

**Helyreállítás (recovery):**

* Archivált redo naplók, valamint – ha alkalmazva van – inkrementális

mentések alkalmazása a visszaállított fájlokra annak érdekében, hogy az adatbázis egy konzisztens, egy időponthoz kötött állapotba kerüljön.

* Ez magában foglalja a redo bejegyzések visszajátszását és a végrehajtott tranzakciók előregörgetését.

**Példányhiba**

* A **példányhiba (instance failure)** olyan szolgáltatáskiesés, amely nem jár adatvesztéssel.
* Jellemző okok:

▫ A DBMS belső kivétele (pl. nem kezelt hiba az adatbázismotorban)

▫ Operációs rendszer összeomlása vagy kernelpánik

▫ Hardverhibák, amelyek nem az adattárolást érintik (CPU, memória

stb.)

* Helyreállítás:

▫ A DBMS példány újraindítása.

▫ Az automatikus összeomlás utáni helyreállítás tranzakciónaplók

alapján, a konzisztens állapot visszaállításához.

▫ A normál működés perceken belül folytatódhat.

**Alkalmazási hibák**

* Az **alkalmazáshibák** az adatok hibás műveletekből eredő sérülését vagy inkonzisztenciáját okozzák.

▫ Programok, szkriptek vagy tranzakciók rossz időben, érvénytelen

bemenetekkel vagy helytelen sorrendben történő futtatása által okozott hibák.

* Következmény: Hibás adatok bekerülése az adatbázisba, ami sérti az integritást.
* Nem DBMS hiba: az alkalmazás logikájából vagy emberi hibából ered
* Megelőzés: a bemeneti validálása, a monitorozás és a riasztások minimalizálják a károkat
* Elhárítás:

▫ Érvényes adatok visszaállítása biztonsági mentésekből.

▫ Hibás rekordok kézi javítása vagy eltávolítása.

* A gyors azonosítás és beavatkozás csökkenti a hibás adatok mennyiségét.

**Médiahibák**

* A **médiahibák** fizikai vagy logikai tárolási sérülések, amelyek az adatok

újjáépítését igénylik

* Adatbázisfájlok elvesztése vagy sérülése adathordozó problémái miatt, pl.,

▫ lemezhibák, fájlrendszer sérülései vagy törölt fájlok, a szalag degradációja vagy sérülése, memóriahibák (bit-hibák)

* Megelőzés:

▫ Redundáns tárolási technológiák (RAID, lemeztükrözés)

▫ Proaktív hardverállapot-ellenőrzés és integritásmonitorozás

* Helyreállítás:

▫ A meghibásodott adathordozó cserélje vagy javítása.

▫ Az érintett fájlok visszaállítása a legfrissebb biztonsági mentésekből.

▫ Teljes helyreállítás végrehajtása a DB konzisztens, működőképes állapotának visszaállításához.

**Főbb biztonsági mentés kategóriák**

**Fizikai (képmásolat) biztonsági mentések**

* Teljes mentés: Minden adatblokkot egyetlen konzisztens képben rögzít.
* Inkrementális mentés: Csak az utolsó biztonsági mentés óta megváltozott

adatblokkokat rögzíti. (Differenciális: az utolsó *teljes biztonsági mentés* óta.)

* Összevont mentés: Egy teljes és az összes inkrementális mentés egyesítése.

**Logikai mentések:** Logikai adatok (pl. táblák, sémák) kinyerése olyan

eszközökkel, mint az EXPORT, DUMP vagy UNLOAD, az adatok egy másik adatbázisba történő átviteléhez.

**Biztonsági mentési módok**

* Hideg (offline): Az adatbázis le van állítva; a fájlok másolása közben nem történik tevékenység.
* Forró (online): A fájlok másolása az adatbázis működése közben történik; a

konzisztencia biztosítása érdekében a redo/naplófájlok mentése szükséges.

**Logikai és OS szintű biztonsági mentések**

* **Logikai mentések (EXPORT/DUMP/UNLOAD):**

▫ Felhasználási esetek: tábla/sor szintű visszaállítás, platformok közötti

migráció, verziófrissítések

▫ Előnyök: olvasható SQL vagy CSV; hordozható; nincs verzióhoz kötve

▫ Hátrányok: Hosszabb exportálás/importálás; az integritást fenn kell tartani

* **OS-szintű mentések:**

▫ Adatfájlok (és opcionálisan csatolt kötetek) másolása a DBMS-en kívülre

▫ Előfeltétel: Az objektumok nyugalomba helyezése vagy zárolása a részleges írások elkerülése érdekében.

▫ Előnyök: DB-független, kihasználhatja a meglévő SAN/NAS snapshot-okat.

▫ Hátrányok: A konzisztencia biztosítása odafigyelést igényel; szükség lehet extra redo log rögzítésre

**Biztonsági mentési módok**

**Hideg (offline) biztonsági mentés:** Az adatállományok, vezérlőfájlok és archivált naplók másolása előtt az adatbázis normál módon leáll.

▫ Garantált konzisztencia: Nincs aktív tranzakció; nincs szükség a naplók

visszajátszására.

▫ Szabványos fájlmásoláson vagy a tároló pillanatfelvételein alapul.

▫ Az adatbázis offline állapotú, így nem áll fenn a részleges írás veszélye.

* Előnyök:

▫ A legkönnyebben megvalósítható és validálható mentési forma.

▫ Azonnal visszaállítja a konzisztens állapotot redo alkalmazása nélkül.

* Hátrányok:

▫ Teljes leállítás szükséges: A felhasználók nem férhetnek hozzá az

adatbázishoz a biztonsági mentés alatt.

▫ Nem alkalmas 24×7-es működést igénylő, nagy rendelkezésre állású környezetekhez.

**Biztonsági mentési módok**

**Forró (online) biztonsági mentés:** Az adatfájlok, vezérlőfájlok és archivált naplók biztonsági mentése, miközben az adatbázis fut és

tranzakciókat szolgál ki.

▫ A felhasználók olvashatnak/írhatnak a biztonsági mentés alatt.

▫ Naplózást igényel: Az online biztonsági mentés visszaállítása szükségessé teszi a redo naplózást a konzisztencia biztosításához.

* Előnyök:

▫ Nincs állásidő; támogatja a 24×7-es működést.

▫ A biztonsági mentéseket munkaidőben is lehet ütemezni.

* Hátrányok:

▫ Komplex helyreállítás: A redo naplók használatával kell előre haladva a

konzisztens állapot elérni.

▫ A mentési műveletek némileg lassítják a működést.

**Táblatér biztonsági mentése**

A **táblatér biztonsági mentése** során a táblatér egy speciális, biztonsági mentési módba kerülnek.

* Ez lehetővé teszi az adott táblatér fizikai, operációs rendszer szintű másolatát.

▫ Az Oracle például lehetővé teszi, hogy a *biztonsági mentési mód*

beállítás egyszerre az összes táblatérre vonatkozzon.

* A biztonsági mentési mód azonban teljesítményproblémákat okoz

▫ Az adott táblatér összes módosítását a redo naplókba kell rögzíteni.

🞄 Ha egy redo naplócsoportot újra kell hasznosítani, akkor a naplókat archiválni is kell.

▫ Minél tovább marad egy táblatér biztonsági mentési módban, annál jelentősebb a teljesítménycsökkenés, annál hosszabb lesz a helyreállítás időtartama, és annál több redo log adat keletkezik.

**Biztonsági mentési szintek és lefedettség**

* **Biztonsági mentési szintek**

▫ Level 0: az adatbázis teljes biztonsági mentése (az adatbázishoz tartozó

összes fizikai fájl mentése)

▫ Level 1: a legutóbbi level 0 biztonsági mentés óta megváltozott összes adat mentése

▫ Level 2: az összes olyan adat biztonsági mentése, amely a legutóbbi, bármelyik szintű (0, 1 vagy 2) biztonsági mentés óta megváltozott

* **Lefedettség**

▫ Naplóarchiválás - az időponti helyreállításhoz szükséges

▫ Rendszerkatalógus - biztonsági mentési és helyreállítási információkat

tartalmaz

▫ Adatbázis objektum definíciók - jó, ha idővel nyomon követhető, hogy mi

változott, és szükség esetén újra létre lehet hozni az objektumokat

**Biztonsági mentési irányelvek**

* **Több biztonsági másolat fenntartása**

▫ Legalább két független másolattal (pl. lemez → szalag, vagy szalag → off- site tároló) elkerülhető, hogy az egyik adathordozó elvesztése vagy sérülése esetén hibapont (single point of failure) álljon elő.

* **A katasztrófa helyreállítási (disaster recovery, DR) tervekkel való összehangolás**

▫ A biztonsági mentési ütemtervnek és eljárásoknak integrálódniuk kell a szélesebb körű DR-stratégiába, pl. hogy egy egész telephelyre kiterjedő kiesés esetén lehetővé váljon a másodlagos helyre történő átállás.

* **A lemezes archiválás szalagos archiválás előtt**

▫ Először egy gyorsabb, lemezes tárolóra történő mentés gyorsítja a biztonsági mentési folyamatot és lehetővé teszi a gyorsabb teszt visszaállításokat is; ha validálták, ezeket a mentéseket hosszú távú megőrzés céljából szalagra (vagy felhőbe) lehet mozgatni.

* **Rendszer/katalógus objektumok mentése**

▫ A rendszer tábláinak, katalógusainak és konfigurációs fájljainak biztonsági mentése kritikus fontosságú a teljes helyreállításhoz, és minden biztonsági mentésnek részét kell képeznie.

* **Rendszeresen ellenőrizni kell a mentések integritását**

▫ Teszt-visszaállítás, ellenőrzőösszegek vagy más érvényesítési rutinok

használata a biztonsági mentések ellenőrzéséhez.

* **Az adatok és a metaadatok mentések szinkronizálása**

▫ Az alkalmazás metaadatainak (konfiguráció, séma definíciók stb.) az adatfájlokkal azonos időpontban történő rögzítése biztosítja a

visszaállítási műveletek esetén a megfelelő verziók elérhetőségét.

**Biztonsági mentés ütemezése**

* **Helyreállítási idő cél (Recovery Time Objective, RTO):**

▫ Milyen gyorsan kell helyreállítani a szolgáltatást?

* **Helyreállítási pont célkitűzés (Recovery Point Objective, RPO):**

▫ Mennyire tolerálható az adatvesztés?

* **Főbb tényezők:**

▫ *Üzleti tevékenység időablaka:* A biztonsági mentéseket akkora kell ütemezni, amikor a terhelés alacsony.

▫ *Az adatok volatilitása:* A gyorsan változó adatokhoz gyakoribb biztonsági mentések szükségesek.

▫ *A rendszer kapacitása:* A biztonsági mentés CPU igénye és I/O

műveletei ne akadályozzák a DBMS (és így az IR) működését.

* **Stratégiák:**

▫ *Teljes vagy inkrementális mentés:* Teljes mentések pl. hetente vagy éjszakánként; inkrementális mentések (level 1) pl. óránként történő végrehajtása az RPO csökkentése érdekében.

▫ *Megőrzési politika:* A sérült biztonsági mentések elleni védelem érdekében legalább 2 generációt kell megőrizni.

▫ *Csúcsidőn kívüli, több lépcsős kitárolás:* A biztonsági mentések először egy gyors lemezre, majd onnan hosszú távú tárolóra történő migrációja csökkenti a rendszer terhelésének idejét.

**Sémák és rendszerkatalógusok mentése**

* Az adatok DDL (táblák, indexek, eljárások stb.) nélkül nem használhatók.
* A katalógusok és a konfigurációs fájlok pl. a felhasználói fiókok, jogosultságok

és a biztonsági mentés metaadatait tárolják.

* **Mit kell tartalmaznia egy mentésnek az adatfájlokon kívül?**

▫ DDL szkriptek: CREATE utasítások generálása minden objektumhoz

▫ Rendszerkatalógusok: rendszertáblák vagy metaadattáblák exportálása

▫ DBMS konfigurációs fájlok: paraméterfájlok, figyelők konfigurációja stb.

* **Jó gyakorlatok:**

▫ A DDL exportálásának automatizálása a séma módosítása után.

▫ Katalógus exportálása minden egyes teljes biztonsági mentéssel együtt.

▫ A szükséges szkriptek tárolása a mentési képek mellett a helyreállításhoz.

**Adatbázis-helyreállítás**

**Az adatbázis-helyreállítás lépései**

1. **Hiba azonosítása:** Határozzuk meg a kiesés típusát például

példányhiba, médiahiba, alkalmazáshiba) és kiterjedését.

1. **Helyzet elemzése:** Vizsgáljuk meg a hibanaplókat, riasztási naplókat és rendszerüzeneteket a kiváltó ok és az érintett

komponensek azonosítása érdekében.

1. **A helyreállítás terjedelmének meghatározása:** Határozzuk meg, hogy mely adatfájlokat, táblaterületeket vagy objektumokat szükséges visszaállítani az üzleti követelmények teljesítéséhez.
2. **Megfelelő biztonsági mentés megkeresése**

▫ Válasszuk ki a helyreállítani kívánt időpontnak megfelelő képmásolatot vagy mentéskészletet.

1. **Biztonsági mentés fájljainak visszaállítása**

▫ Másoljuk a kiválasztott mentés fájljait az eredeti vagy új helyükre.

1. **Előre görgetés (naplók alkalmazása)**

▫ Alkalmazzuk az archivált redo naplókat (és az esetlegesen használt

inkrementális mentéseket) a visszaállított fájlokra annak érdekében,

hogy az adatbázis konzisztens állapotba kerüljön.

**Visszatérés a jelenlegi állapothoz**

* **A jelenlegi állapot helyreállítása** általában a médiakhibák esetében szükséges. A visszaállítás teljes, kumulatív vagy inkrementális biztonsági mentésekből végezhető.

**System Change Number (SCN)**

Az **SCN** egy folyamatosan növekvő sorszám, amelyet az Oracle minden

COMMIT műveletnél (és egyes belső műveleteknél) generál.

* **Egyetlen, globális „óra”:** Biztosítja, hogy minden adatfájl és vezérlőfájl ugyanarra a logikai pontra hivatkozzon; így elkerülhető az úgynevezett “fuzzy” állapot.
* **Granuláris visszaállítás:** Pontosan megadható, meddig gördítsük

előre a naplókat.

* **Konzisztens olvasás:** A lekérdezések olyan pillanatképet kapnak, amelynek SCN-je nem haladja meg a lekérdezés indításakor rögzített “query SCN” értéket.
* Az Oracle SCN-hez hasonló az MS SQL Server és a PostgreSQL esetében

az *LSN* (Log Sequence Number).

**Point-in-time helyreállítás**

* **Point-in-time (PIT) helyreállítás** logikai vagy alkalmazáshibák utáni helyreállításra használják jellemzően.
* Az adatbázis visszagörgetése a hiba előtti pillanatra:
  1. Teljes adatbázis-mentés visszaállítása: A legfrissebb, konzisztens képmásolat vagy teljes mentés megkeresése és visszaállítása.
  2. Naplók előre görgetése: Az archivált redo naplók (és az

inkrementális mentések) a mentési ponttól a hibás módosítást

megelőző időpontig történő alkalmazása.

* 1. Az adatbázis megnyitása a cél SCN-nél/időbélyegnél
* A helyreállítási pont előtt véglegesített összes helyes tranzakció eredménye megmarad; az azt követő módosítások pedig visszavonásra kerülnek.

**Tranzakcióhelyreállítás**

* A **tranzakcióhelyreállítás** egy tranzakció (alkalmazás-logikai hiba

vagy felhasználói tévedés) okozta problémákat kezel.

▫ **PIT helyreállítás:** Az egész adatbázist egy választott időpontra

állítja vissza, így az addigi valamennyi – helyes és hibás – tranzakciót visszagörgeti.

▫ **UNDO alapú helyreállítás:** Csak a hibás tranzakció hatásait vonja vissza, miközben minden más, már véglegesített módosítás megmarad.

▫ **REDO alapú helyreállítás:** Először PIT-helyreállítással eltávolít minden változást, majd az archivált redo naplók segítségével csak az érvényes tranzakciókat játssza vissza, hogy az adatbázis naprakész

állapotba kerüljön.

**Magas rendelkezésre állású alternatívák**

* **Készenléti adatbázis/Adatbázis-tükrözés:**

▫ Az elsődleges adatbázis folyamatosan frissített, azonos másolata.

Meghibásodás esetén az alkalmazások minimális kieséssel állhatnak

át a tartalék adatbázisra.

* **Adatreplikáció/Tranzakcióalapú replikáció:**

▫ Egyes táblák vagy adatok valós idejű másolása egy külön adatbázisba. Ezáltal redundáns, földrajzilag elosztott adattárolók hozhatók létre az olvasási teljesítmény növeléshez és a gyors átálláshoz szükség esetén.

* **Hardveres redundancia lemezreplikációval:**

▫ A tárolókötetek szinkron vagy aszinkron tükrözése, akár lokálisan, akár egy távoli helyszínen. Eszközhibák esetén is védelmet biztosít egy teljes lemezkép duplikátumának fenntartásával.

**Katasztrófa utáni helyreállítás tervezése**

* **A katasztrófa-helyreállítási tervezés (vészhelyzeti tervezés)** a

kritikus üzleti alkalmazások nem tervezett, hosszabb kieséseire készül fel.

▫ Számítási erőforrások kiesése miatti, pl. 48 órát meghaladó kiesések.

▫ Alacsony valószínűségű, nagy bizonytalansággal járó és potenciálisan katasztrofális hatású események.

🞄 *Példák:* természeti katasztrófák (pl. tüzek, árvizek, hurrikánok) és

ember okozta események (pl. szabotázs, súlyos áramkimaradások).

* A működés helyreállítására összpontosít a nagyobb zavarok után.
* Előre meghatározott irányelveket, eljárásokat és erőforrás-elosztást igényel.
* Eredmény: Egy dokumentált, tesztelt terv, amely lehetővé teszi a szervezet

számára a kritikus funkciók gyors helyreállítását és a pénzügyi és reputációs

veszteség minimalizálását.

**Irányelvek**

* **A leállási időt és az adatvesztés minimalizálása**

▫ Olyan stratégiákat dolgozzunk ki, amelyek egy incidens után gyorsan

visszaállítják a szolgáltatásokat és az adatokat.

* **Az üzleti kockázatok értékelése, nem a technikai kérdéseké**

▫ Azonosítsuk a kritikus üzleti folyamatokat és azok fennakadással

szembeni tűrőképességét.

* **Az alkalmazások priorizálása kritikusságuk szerint**

▫ A rendszereket kategóriák szerint rangsoroljuk (például 1. szint: létfontosságú, 2. szint: fontos, 3. szint: nem kritikus).

* **Összehangolás a vállalati szintű működéssel**

▫ Biztosítsuk, hogy a helyreállítási tervek lefedjenek minden fontos funkciót, valamint a szervezeten belüli kölcsönös függőségeket.

**Irányelvek**

* **Külső helyszíni helyreállítási központ létrehozása**

▫ Biztosítsunk földrajzilag elkülönített helyszínt az adatok mentésére és a

vészhelyzeti átállásra.

* **Terv dokumentálása és megosztása**

▫ Készítsünk katasztrófahelyreállítási kézikönyvet, és juttassuk el minden

érintett munkatárshoz.

* **Rendszeres tesztelés és validálás**

▫ Ütemezett gyakorlatok és szimulációk révén ellenőrizzük a terv hatékonyságát, és szükség esetén aktualizáljuk azt.

**Oracle helyreállítási stratégia**

* **Helyreállítás médiahiba után: helyreállítás a jelenlegi állapotba**

▫ A biztonsági másolatok visszaállítása, és az összes szükséges archivált

redo napló alkalmazása, hogy az adatbázis a legutóbbi konzisztens

állapotba kerüljön.

▫ A felhasználó által kezelt helyreállítás esetében a nem rendszerfájlokkal kapcsolatos hibák gyakran online is kezelhetők, de egy maradandó sérülés teljes helyreállítást és adathordozó helyreállítást igényel.

* **Helyreállítás alkalmazáshiba után: PIT és Flashback**

▫ Egy biztonsági mentést visszaállítása, az adatbázis előregörgetésé a hiba előtti időpontra, ezzel visszavonva az azt követő összes módosítást (PIT).

▫ Az Oracle Flashback teljes visszaállítás nélkül, kizárólag a logikai

változtatások visszagörgetésével állítja helyre az adatokat.

* **Helyreállítás példányhiba után: automatikus összeomlás utáni helyreállítás**

▫ Indításkor az Oracle automatikusan elvégzi a példány helyreállítását.

▫ Az online redo naplókból a véglegesített tranzakciókat előre gördíti, a nem véglegesített tranzakciókat pedig vissza.

▫ A DBA egyszerűen újraindítja az adatbázist; nincs szükség manuális visszaállításra.

▫ Gyakori okok:

🞄 Áramkimaradás vagy hardverhiba

🞄 Az operációs rendszer összeomlása

🞄 Egy Oracle háttérfolyamat hibája

🞄 SHUTDOWN ABORT parancs kiadása

Mentési kategóriák teszt kérdés