

Mi alapján választunk a relációs
adatbázisok közül?

A relációs adatbázisok kötelezően elvárható
és opcionális képességei

Mi alapján választunk a relációs adatbázisok közül?

- A szoftver ára és a működési sebesség (performancia) messze nem az egyedüli kritériumok.
- És még ezek a kritériumok is igen összetettek - sok különböző aspektusuk és összetevőjük van.
- 1-1 ilyen választásnak jellemzően évtizedes igen széleskörű következményei vannak egy szervezet (pl. egy cég) működésére.
- Befolyásolja majd ez a személyzeti kérdéseket is, meg a szóba jöhető (kiválasztható) szoftver-rendszerek halmazát is.
- A kérdés azért is rendkívüli fontosságú, mert gyakran az informatikára elköltött pénzek legnagyobb részét is ez képezi.

1. Azok a képességek, amelyekkel mindegyik relációs adatbázezelőnek rendelkeznie kell

- Táblák, nézetek
- SQL nyelv: DDL, DML, Query
- Tranzakciókezelés
- ACID képességek:
 - Atomi tranzakciók (Atomicity)
 - Konzisztencia (Consistency)
 - Izoláció (Isolation)
 - Tartósság (Durability)
- Kényszerek
- Adatszótár
- Optimalizáló (automatikus végrehajtási terv generátor)
- ...

2. Az adatbáziskezelő rendszer költsége

- Licenzköltség: jellemzően a felhasználók számától vagy a hardver méretétől függ.
- A licenszben különböző szoftverkiszzerelések létezhetnek és különböző opcionális elemek.
- Szoftveres támogatás (support) éves díja
- Saját gépeken való használat licenszdíja, vagy felhőből bérelt szolgáltatás bérleti díja
- Esetleges adatvesztésből fakadó károk: kicsiny az esélye (de nem nulla), viszont hatalmas az üzleti kár
- Tervezett és tervezetlen állásidőből fakadó üzleti károk
- A szükséges hardver költsége
- Az üzemeltető személyzet költségei (bérek, képzés...)

3. Kiforottság, megbízhatóság

- Sajnos minden szoftver bug-os
- Kockázatos olyan adatbáziskezelőt választani, amelyben mi fogjuk először megtapasztalni a hibát.
- A jó szoftver az, amelyet már mások is alkalmaztak ugyanolyan célokra, ugyanolyan körülmények között.
- Ha bármi kételyünk van, az interneten találunk róla valami leírást.
- Olyan az adatbáziskezelő, mint a jó bor: minél öregebb, annál jobb.

4. Elterjedtség

- A jó adatbáziskezelőt rajtunk kívül sok száz másik munkahelyen használják (a szomszédban is ez van).
- Ha szükség van egy új, de tapasztalt munkatársra, könnyen találunk.
- Ha szükség van egy konzultánsra, megtaláljuk.
- Könnyű, szinte folyamatos a tapasztalatcsere.
- Ha szakmai segítségre, esetleg csak tanácsra van szükségünk, azt több helyről is megkaphatjuk.

5. Szabványok betartása

- A relációs adatbázisok de-facto szabványa az SQL.
- Ezen belül is előny, ha minél inkább betartja a gyártó az SQL szabvány részleteit is.
- A szabványtól eltérő (azon túlmutató) utasítások ugyan hasznosak lehetnek, de kockázatosak is:
 - Később nehezebb lesz adatbáziskezelőt váltani
 - Később megszűnhet a támogatásuk

6. A támogatott programozási nyelvek

- A jó adatbáziskezelő rendszernek minél több nyelvet kell támogatnia
- Java nyelvhez: JDBC
- C, C#, .NET támogatás szinte kötelező.
- PHP támogatás is előny.
- Cobol, Fortran, egyéb klasszikus programozási nyelvek támogatása: beágyazott SQL („embedded SQL”) technológiával.

7. Hardver és operációs rendszer támogatása

- Sok architektúrát támogat: Intel, SUN SPARC, HP PA-RISC, és különböző IBM hardverarchitektúrákon is működik.
- Támogatja a virtualizációt elméletben és árazásban is.
- Előny, ha működik Linux, Windows, Solaris, AIX, HP-UX, esetleg VMS operációs rendszereken is.
- A későbbi platformváltások legyenek könnyűek.
- „Embedded Database”: egybeolvad a felhasználói program és az adatbáziskezelő rendszer. Jellemzően nincs külön adatbázis, amelyet felügyelnünk kellene.
- „Mobile Database”: okostelefonokon, PDA-kon futó rendszerek.

8. Saját gépterem, vagy felhő

- Idővel egyre jelentősebb előnnyé válik majd az, hogy nemcsak a saját gépterünkben működtethetjük, hanem bérelhetünk a felhőben is ilyen szolgáltatást: „Platform as a Service”, vagyis PaaS.
- A PaaS nemcsak elméleti lehetőség kell hogy legyen, hanem a gyakorlatban is olajozottan kell működnie.

9. Adattípusok széleskörű támogatása

- Az adatbázisokban manapság már nemcsak szövegeket, számokat és dátumokat tárolunk.
- Képek, tetszőleges szöveg, térinformatikai adatok, stb.
- XMLType, JSON
- User Defined Data Type
- Új objektumtípusok és objektumok tárolása (attribútumokkal és metódusokkal).

10. Karakterkészlet

- A Unicode támogatása szinte kötelező.
- A Unicode legújabb szabványa 10.0. Ebben 136.000 jeltől áll a „repertoár” (régén ezt „Character Set”-nek nevezték volna, de ez most nem polkorrekt).
- A kódolást vagy UTF-8, vagy UTF-16 (ennek a régebbi, lebutított változata az UCS-2).
- A jó adatbáziskezelő rendszerben lehetséges a kódkonverzió az egyéb (régi) karakterkészletekről Unicode-ra.
- A jó adatbáziskezelőben ez a konverzió állásidő nélkül vagy rövid állásidővel elvégezhető.
- A jó adatbáziskezelő rendszerben nem nő számottevően a helyigény a Unicode miatt (tehát UTF-8 választható).

11. Elosztott tranzakciókezelés

- Nemcsak egy adatbázison belül kell megbízhatóan kezelnie a tranzakciókat, hanem két adatbáziskezelő rendszer között is.
- A két adatbázis közötti kommunikáció rendszerint két külön kategóriát is jelent:
 - Két azonos adatbáziskezelő rendszer közötti megbízható adatátvitel (pl. Oracle esetén az adatbázis link és a kétfázisú jóváhagyási mechanizmus teszi ezt lehetővé)
 - Két különböző gyártó rendszere között: XA tranzakciók támogatása
- Szükséges a megbízható tranzakciókezelés egy adatbáziskezelő rendszer és egy másik szoftver - mondjuk egy üzenetküldő rendszer („Messaging System”) - között is: rendszerint XA protokoll.

12. Procedurális lehetőségek

- Az SQL nyelv nagyszerű, de mégis akadnak esetek, amikor más kéne.
- Szükségünk lehet algoritmusokra is, és ezek adatbázison belüli tárolására.
- Így a feldolgozás az adatok „közelében” történhet.
- Így kiterjeszthető az adatbáziskezelő rendszer funkcionalitása:
 - Maga a gyártó is fejleszthet ilyen kiterjesztéseket
 - A felhasználó kifejlesztheti a saját kiterjesztéseit
 - A kényszerek halmaza is kibővíthető így
 - Az objektumorientált metódusok is ebben a nyelvben íródnak
- Pl. Oracle esetén két ilyen nyelv is rendelkezésre áll: PL/SQL és Java.

13. Adatbázis triggererek

- Igen hasznos egy olyan képesség, hogy bizonyos programok maguktól végrehajtódnak (elsülnek) egyes események bekövetkeztekor:
 - Adatok beszúrása előtt (Pre-INSERT triggererek). Ezeket jellemzően a beszúrás ellenőrzéseként használjuk.
 - Adatok beszúrása után (Post-INSERT triggererek). Ezeket gyakran a művelet naplózása érdekében alkalmazzuk.
 - INSERT mellett DELETE és UPDATE esetére is hasznosak a triggererek.
 - Nemcsak DML triggererek létezhetnek, hanem pl. olyanok, amelyek bejelentkezéskor, vagy pl. a teljes rendszer elindulása vagy leállása esetén futnak le.

14. A funkcionalitás kiterjesztése SQL-en túlra

- A többmillió adatbázis-felhasználó cégnek (esetleg embernek) gyakran van olyan közös igénye, amely túlmutat az SQL-en, de mégis praktikus. Ilyen pl:
 - Ütemezett feladatok végrehajtása
 - Email-ek küldése
 - Fájlok olvasása/írása
 - Üzenetek küldése és fogadása
 - ...
- Amennyiben létezik procedurális lehetőség az adatbázison belül, akkor rendszerint a gyártó biztosít ilyen funkcionális kiterjesztéseket.
- Nyílt forráskódú adatbáziskezelő rendszer esetén ez lehet közösségi fejlesztés is.

15. Magas rendelkezésre állás

- A „High Availability” egyre fontosabbá válik.
- Egyre gyakrabban van szükségünk 7*24 órás rendelkezésre állásra. Persze valamiféle állásidőnek manapság még ilyenkor is mindenképpen lennie kell.
- Az állásidőnek két fajtája van:
 - Betervezett (előre bejelentett)
 - Nem betervezett (valamilyen hiba miatt)
- Pl. az „5 kilences” rendelkezésre állás (vagyis a 99.999%-os rendelkezésre állás) azt jelenti, hogy évente kb. 5-6 percet állunk csupán. Ez nagyon nehezen teljesíthető, de nem lehetetlen.
- Oracle esetén pl. a RAC opció a legfőbb magas rendelkezésre állási képesség.
- Problémás a szoftverek frissítése: oprendszer, adatbáziskezelő rendszer, az adatbázis adatszótára...



16. Helyreállíthatóság

- Előbb utóbb nemcsak összeomlik egy szoftver, hanem adatvesztés is történhet (pl. lemezhiba miatt).
- Az adatbáziskezelő rendszernek olyan mentési mechanizmus kell, amely biztosítja nemcsak a régi mentésre való visszaállást („Restore”), hanem a legfrissebb állapotba való helyreállást is („Recovery”).
- A helyreállítás ideje is kritikus:
 - a jó eset manapság néhány perc
 - a nem túl jó eset több óra
 - Az elfogadhatatlanul lassú helyreállítási idő manapság a több nap
- Fontos, hogy a helyreállítás könnyű és megbízhatóan végrehajtható feladat legyen.
- Azért mindenre kiterjedő abszolút biztonság nem létezik.

17. Katasztrófatűrés

- Nemcsak a ténylegesen várható veszélyekre illik felkészülni, hanem olyan katasztrófa-helyzetekre is, amelyek szinte kizártnak tűnnek:
 - Tűzvész, árvíz, földrengés
 - Terrortámadás
- Aki ilyen ellen is védekezni akar (egyre többen), azok távoli adatbázis-másolatokat szeretnének működtetni.
- Ezt megoldhatja valamilyen hardveres vagy szoftveres távoli tükrözés is, de lehet ez akár az adatbáziskezelő rendszer extra képessége.
- Oracle esetén a DataGuard a katasztrófatűrő megoldás.

18. Hatékonyság (gyors működés)

- Szándékosan nem az elsők között került említésre. Sajnos a korai fázisban sokan ez alapján választanak. Egy erősebb hardver ellensúlyozhatja a szoftver hatékonyságát.
- Leginkább a funkcionalitás rovására válik 1-1 adatbáziskezelő rendszer gyorsá.
- Máskor azért gyors mert memóriában dolgozik. Ez persze sérülékenyebbé teszi.
- Ha a sebesség kiemelkedően fontos, akkor is csak a releváns terheléseket vegyük figyelembe.
- Szinte mindegyik adatbáziskezelő rendszer gyártója kozmetikázza a hatékonysági mutatók értékeit.
- Minden újabb verzióról azt mondják, hogy gyorsabb mint a megelőző verzió, közben rendszerint lassabb.

19. Skálázhatóság

- A sebesség mellett legalább olyan fontos kérdés, hogy ha növekszik a terhelés, tudjuk-e ehhez igazodva növelni a teljesítményt.
- Ideális a lineáris skálázhatóság lenne: kétszer annyi hardverrel kétszer annyi munka elvégzése.
- A lineáris skálázhatóság szinte csak álom
- Lehet egyre több processzor egy gépen belül, vagy egyre több számítógép egy „cluster”-ben.
- Oracle esetén a RAC lenne a válasz a skálázhatóságra is.

20. Adatok titkosítása

- A merevlemez-alapú adatbázisoknál az adatok a merevlemezen fájlokban tárolódnak
- Komoly kockázat, hogy ezeket a fájlokat esetleg ellopja valaki és így jut hozzá az adatokhoz
- Ezt a komoly adatbáziskezelők úgy védik ki, hogy a fájlban már titkosított módon tárolhatnak adatot
- Ilyenkor az INSERT utasítás „titkosítva ír”, és a SELECT utasítás fejti azt vissza.
- Aki tehát SQL művelettel fér hozzá az adatokhoz, azt ez a titkosítás „nem érinti”.
- Ugyanilyen titkosítás létezhet a kliens-szervet kommunikáció során is.
- Oracle esetén ezt „Advanced Encryption Option”-nek nevezik. És sajnos feláras lehetőség.

21. A használat auditálása

- Az adatbáziskezelő rendszerek zsargonjában auditálásnak nevezik az adatbázis-használat figyelését biztonsági célokból.
- Azt kell tudnunk, hogy melyik felhasználó mikor milyen műveleteket hajtott végre.
- Az auditálás konfigurálható kell hogy legyen:
 - Opcionálisan eldönthető, hogy legyen-e audit
 - Ha van audit, akkor konfigurálható kell hogy legyen annak a részletessége
- Az auditálás működtetése nem lassíthatja számottevően az adatbáziskezelő rendszert.
- Hatékony elemzési lehetőségek kellenek az „Audit Record”-ok felett.
- Figyelem: ez nemcsak van/nincs kérdés. Lehet auditálni úgy is, hogy az hasznavehetetlen.

22. A mentések sokszínűsége

- A jó adatbáziskezelő rendszernek saját mentési program/módszer kell, de legyen alternatíva is.
- Ezzel nemcsak lementhető, hanem egyúttal logikailag ellenőrizhető is az adatok tartalma.
- Előny az, ha ez a program az adatbáziskezelő rendszer belsejébe van beleépítve.
- Hatékony (valószínűleg párhuzamosítható) mentési módszer kell.
- Teljes és inkrementális mentésre is legyen lehetőség.
- A mentés elvégezhető legyen online.
- A mentési katalógus is nagyon hasznos.
- Lemezre és szalagos mentőegységre is lehessen menteni.
- Harmadik fél eszközeivel (módszereivel) is lehessen menteni.

23. Milyen segédprogramok léteznek

- A gyakorlatban felmerül az igény különböző nem-SQL feladatok egyszerű és hatékony elvégzésére.
- Ilyenek pl.:
 - Adatok betöltése az adatbázistáblákba:
 - szöveges fix formátumú állományokból
 - CSV („Comma Separated Values”) fájlokból
 - Excell táblákból
 - XML vagy JSON állományokból
 - Adatok igény szerinti áthordozása adatbázistáblákból bináris állományokba és később ezek visszatöltése ugyanabba, vagy másik adatbázisba
- Pl. Oracle esetén ezek az eszközök az SQL*Loader és az Oracle DataPump Export és DataPump Import.



24. Szakember általi monitorozás, hangolás lehetősége

- Az adatbázis sebessége ugyan nem a legfontosabb kiválasztási kritérium, de annak mégis hatalmas jelentősége van, ha egy rendszer nagyon precízen monitorozható, és kideríthető róla, hogy hol van a szűk-keresztmetszete
- Ha ezután még át is paraméterezhető a működés, úgy, hogy a szűk keresztmetszet eltűnjön, vagy csupán enyhüljön, az már főnyeremény.
- Nagyon fontos, hogy ez a monitorozhatóság SELECT utasításokkal történjék. Ezáltal ugyanis harmadik fél is gyártani tud monitoring eszközt (nyílt rendszerek).

25. Automatikus monitorozás és hangolás

- Egyre több rendszert működtetünk. Egyes cégeknél több száz adatbázis van napi használatban. Nem érkeznek a rendszergazdák mindegyikre folyamatosan odafigyelni.
- Nő a „monitoring”, mint tevékenységi kör jelentősége.
- A jó rendszereket nem elsősorban kívülről figyelik („polling”), hanem azok saját magukat figyelik belülről, és riasztanak minket, ha baj van. Erre még nincs egységes szóhasználat, de gyakran „Server Generated Alert System” a neve.
- Az igazán jó rendszerek nemcsak riasztanak, hanem megoldást is javasolnak, sőt esetleg a javasolt változtatást meg is teszik automatikusan.

26. A szoftver fejlődésének a képessége és üteme

- Az igények folyamatosan bővülnek és változnak.
- Azt a szoftvert el kell kerülni, amelynek a gyártója nem fejleszti a terméket nagy tempóban.
- Egyes szoftverek nagyon gyenge architektúrális alapokra épültek, és ezért nem tudnak továbbfejlődni.
- Másik probléma lehet, ha a kód már ősrégi, és most már csak nehezen karbantartható.
- Fontos kérdés, hogy milyen programozási nyelvet használtak az adatbáziskezelő rendszer gyártói.
- E téren pl. az Oracle C-ben íródott, ami ma már inkább rossz tulajdonság, mint jó, és a kód jelentős része igen „dohos”. Ezt próbálják ellensúlyozni azzal, hogy rengetegen fejlesztik.

27. A szoftvertámogatás minősége

- „Bugs are facts of life” - úgy tűnik, hogy sajnos elkerülhetetlenek.
- Ha már vannak hibák, kritikus kérdés, hogy van-e aki javítsa őket?
- Hányan és hány problémát oldanak meg? A mi problémánkat jellemzően megoldják-e, és ha igen, mennyi idő alatt?
- A megoldási módszer is lényeges, hiszen érzékeny adatokat tárolunk. Ki férhet hozzá a hiba felderítése során?
- A „support” általában borsos áron történik.
- Jellemzően nemcsak a hibajavításokat, hanem az új verziókat is fedli a support-díj.
- A támogatás nemcsak a téves kódra, hanem a téves használatra is kiterjed-e vajon?

28. Feltörési lehetőségek, adatlopások

- Minden adatlopás egy tragédia. Sok esetben a következmény a cég megszűnése.
- Nem létezik 100%-os biztonság.
- Mégis érdemes felmérni a választás előtt, hogy az adott szoftvert milyen gyakran törik fel? Hány incidens került nyilvánosságra az elmúlt években?
- Hogyan reagál a szoftvergyártó, ha kiderül 1-1 sebezhetőség?
- Megtesz-e a gyártó minden tőle telhetőt ahhoz, hogy megelőzze az újabb sebezhetőségek kialakulását.
- Csupán „Denial-Of-Service” típusú sérülékenységekről beszélünk, vagy komoly adatlopások, adatmódosítások is történhetnek (pl. „SQL Injection”)?

29. Mennyire védettek az adatok a DBA-tól

- Egy komoly és eldöntendő kérdés, hogy veszélyforrásként tekint-e a cég a rendszergazdákra, vagy nem.
- Ha félünk a DBA-k és az operációs rendszerek rendszergazdáinak a jogosultságaitól, akkor vajon létezik-e olyan szoftververzió, ahol a DBA és a rendszergazda sem férhetnek hozzá az adatokhoz?
- Pl. az Oracle adatbázis esetén ezt a megoldást Oracle Database Vaultnak nevezzük.

30. És még sok egyéb szempont is felmerülhet

- Az eddigiekben felsorolt kritériumok nem egy végleges listát alkotnak.
- Az évek során és a technológia fejlődésével állandóan új szempontok merülnek fel.
- Az látszik, hogy a „melyik a jobb adatbátiskezelő rendszer” kérdés nagyon sokrétű, komplex.