Frontend:
A frontend a programoknak, weboldalaknak az a része, amelyik a felhasználóval közvetlenül kapcsolatban van. Feldata az adatok megjelenése, befogadása a felhasználó, vagy ritkább esetben egy másik rendszer felöl.

Frontend fejlesztés:
A frontend fejlesztés a webfejlesztés egyik területe. Minden olyan webes elem és funkció ami a böngészőben jelenik meg és amikkel a felhasználók interakciókba kerülnek, frontend programozással valósul meg és működik. A front endfejlesztést más szóval kliensoldali fejlesztésnek is nevezhetjük.

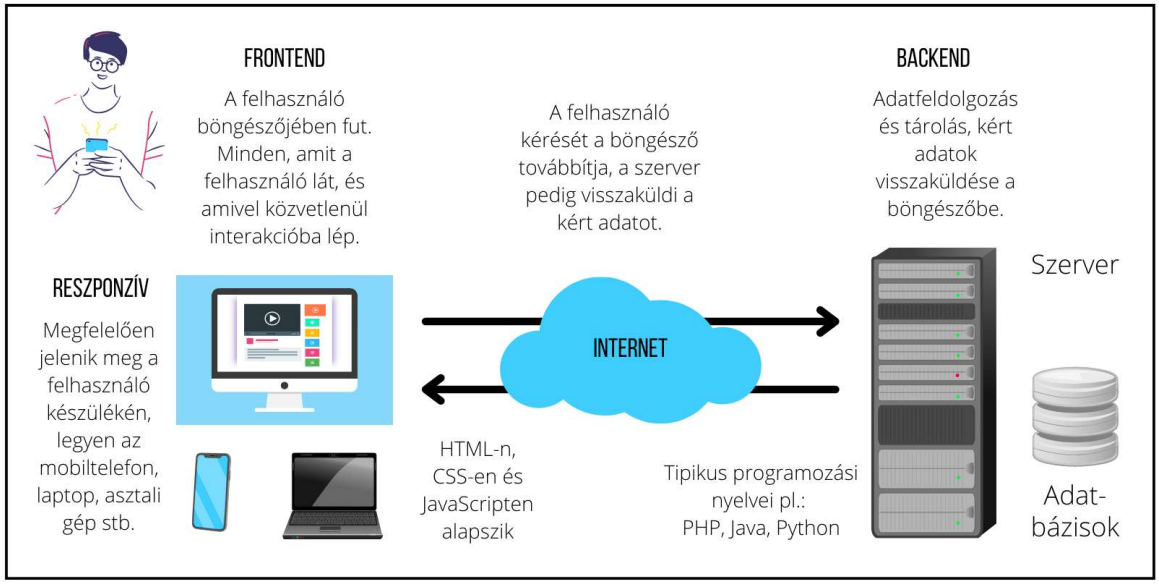
Amit egy-egy weboldalon látunk, szövegek, képek, gombok, animációk, navigáció stb. Ezeket mind frontend programozók valósították meg.

A kifejezés az angol front (elülső) és end (vég) szavakból tevődnek össze.

Kliensoldal és szerveroldal:
A webfejlesztés egy gyűjtőfogalom, ami a weboldalak és webes alkalmazások fejlesztésének különböző területeit foglalja magába. Két nagy ága van, a kliensoldali és a szerveroldali programozás. A kliensoldal fejlesztése az a folyamat, amely során a weboldal megjelenését, stílusát és tartalmát kódok formájában írják meg.

A szerveroldali programozás a szerveren futó háttérfolyamatok felépítéséről és fenntartásáról szól, tehát azoknak az adatoknak a feldolgozásáról, rendszerezéséről, tárolásáról, előhívásáról, amelyek a weboldal, valamint az ott elérhető szolgáltatás működéséhez szükségesek.

Böngésző és webszerver:
A böngésző az a program, aminek segítségével az interneten elérhető tartalmakhoz, egyszerű felhasználóként hozzá férünk. Ezek közül a legismertebbek: Google Chrome, Mozilla Firefox, Internet Explorer, Opera. A felhasználó eszközén (okostelefonján, számítógépén, tabletén stb.) futnak.
A szerverek a színfalak mögött találhatók, tárolják és kezelik az adatokat, legtöbbször hatalmas mennyiségben. A böngésző továbbítja az adatkéréseket (request) a felhasználó oldaláról, a szerver pedig visszaküldi a kért információt (response), amit újra a böngésző jelenít meg.



Frontend programozási feladatok pl.:

- szövegek, képek, videók, animációk megjelenítése
- navigáció az aloldalak között
- interakció a felhasználó és a weboldal között (pl. adatok bekérése)

A frontend fő programozási nyelvei:
Minden weboldal három fő programozási nyelvre épül, a szerkezet vázát a **HTML** (Hypertext MarkUp Language) adja, erre épül rá a weboldal megjelenését, stílusát, színeit, elrendezését meghatározó **CSS** (Cascading Style Sheet), illetve az interaktivitásért, kattinthatóságért, navigációért felelős **JavaScript**.

HTML:
A *HTML* az angol HyperText Markup Language rövidítése, ami magyarul hiperszöveges jelölőnyelvet jelent. **Weboldalak készítéséhez kifejlesztett nyelv**, ami az oldal szövege mellett különböző leíró elemeket tartalmaz, amik blokkokba rendezik a tartalmat (például külön a menüpontokat, és a fő szöveget), illetve olyan plusz elemekkel egészítik ki, mint például képek, videók vagy űrlapok.

Amikor megnyitsz egy weboldalt, a böngésződ a webszerverről letölt egy HTML formátumban megírt szöveges fájlt (ez a weboldal tartalma), illetve általában egy CSS fájlt is, ami azt határozza meg, hogy ez a tartalom pontosan hogyan nézzen ki.
Egy tipikus HTML fájl részlete például így néz ki:

```
<h1>Köszönés</h1>
<p>Helló világ!</p>
```

Itt a kacsacsőrök között szereplő h1a tartalom címét jelenti (angolul heading), a p pedig azt, hogy a benne lévő szöveg egy bekezdést alkot (angolul paragraph).
A rövidítésben szereplő hiperszöveg szó arra utal, hogy a HTML oldalak általában nem egy könyvhöz hasonló, az elejétől a végéig hosszan olvasandó szöveges tartalmat, hanem hiperhivatkozásokkal (közismertebb nevükön linkekkel) teletűzdelt szöveget jelenítenek meg, amikben egy-egy szóra kattintva más weboldalakra juthat át az olvasó.

CSS:
Az angol Cascading Style Sheets rövidítése, melynek szószerinti fordítása: “Egymásba ágyazott stíluslapok”.

A CSS egy olyan **számítástechnikai nyelv, amivel a weboldalak kinézetét lehet megadni**, például hogy milyen színű legyen egy honlap háttere és mekkora betűk jelenjenek meg rajta.

Az angol ***C*ascading *S*tyle*S*heets rövidítése**, melynek szó szerinti fordítása "egymásba ágyazott stíluslapok". A CSS egy olyan **számítástechnikai nyelv, amivel a weboldalak kinézetét lehet megadni**, például hogy milyen színű legyen egy honlap háttere és mekkora betűk jelenjenek meg rajta.

A háttér például fehér, a fejléc azonban fekete színű, ami CSS-ben kifejezve így néz ki:

```
body { background: white; }
header { background: black; }
```

Az **egymásba ágyazhatóság** arra utal, hogy egy-egy meghatározott stílus (például a fehér háttér) a weboldal több elemére is érvényes, mert a stílusok "öröklődnek" a weboldal felépítése szerint. Ezt úgy kell elképzelni, mintha a weboldal halmazokból állna: ha a legnagyobb halmaznak megadsz egy stílust, akkor a benne lévő kisebb halmazok is "megöröklik" azt a külsőt. A fenti példában megadott fehér háttér például a legnagyobb halmazra, az egész weboldalra érvényes. Mivel a fejléc ezen a halmazon belül van, ezért innentől kezdve a fejlécnek is fehér a háttere. Ezért kellett az alatta lévő sorral felülírnunk a fehér hátteret a fekete szín megadásával.

A CSS első verziója 1996-ban jelent meg. Megalkotásának elsődleges célja az volt, hogy **a weboldalak tartalmát** (azaz azt, hogy mi jelenjen meg) **elkülönítsék a külalakjától** (azaz attól, hogy hogyan jelenjen meg). Az olvasott szövegek két helyről előhívva jelenik meg: a HTML tartalmazza magukat a mondatokat, és a CSS adja meg, hogy ezek a mondatok pl: fehér alapon fekete színű betűkkel jelenjenek meg, sőt azt is, hogy ez aláhúzott, *ez pedig dőlt* legyen. Ennek köszönhetően egyrészt sokkal kisebb a weboldalak mérete, így gyorsabban betöltődnek, másrészt lényegesen egyszerűbb megváltoztatni a weboldalak arculatát. Ha például kék színben szeretnénk megjeleníteni ezt a szöveget, csupán csak egy helyen, egyetlen sort kellene átírni a kódban ahelyett, hogy mindenhol kiadnánk a parancsot, hogy a betűk kékek legyenek.

JavaScript:
A JavaScript vagy röviden JS, egy olyan programozási nyelv, amivel legsűrűbben weboldalakon, a böngészőben futó programokat írnak.

A JavaScript segítségével **interaktív weboldalak**, és olyan webalkalmazások készíthetőek, mint például a Gmail. Amikor például egy weboldalon egy gombra kattintva elsötétül a képernyő, és az előtérben megjelenik egy kis ablak, akkor azt általában egy JavaScript nyelven írt program csinálja. A JavaScript programok forráskódját a böngésző teljes egészében letölti, majd értelmezi és futtatja. A hozzáértő személyek ezért a forrás megtekintésével könnyen meg tudják nézni, hogy egy-egy JavaScript program hogyan épül fel és pontosan mit csinál.
A legtöbb modern **weboldal működéséhez szükséges a JavaScript**. Gyakran használják reklámok megjelenítése céljából, illetve egy rosszindulatú oldal kártékony módon is használhatja, ezért néhányan alapból letiltják a futtatását, és csak utólag engedélyezik azt egy-egy megbízható oldal számára.

Backend:
A *back-end* a **programoknak, weboldalaknak a hátsó, a felhasználó elől rejtett, a tényleges számításokat végző része**. Feladata a front-end (a felhasználóval kapcsolatban lévő rész) felől érkező adatok feldolgozása, és az eredményeknek a front-end felé történő visszajuttatása.

Weboldal esetén a back-end fejlesztő foglalkozik **például** a PHP kódok és az adatbázis kezelésével, a front-endhez pedig például a HTML és a CSS kódok tartoznak.
A kifejezés az angol back (hátsó) és end (vég) szavak összetételéből áll.

Backend fejlesztés:

A **szerveroldali** programozás a **backend** fejlesztés.

Backend programozási feladat pl.:

- adatbázisokból lehívott információk
- adatbázisokban elhelyezni a felhasználó által beírt adatokat
- ajánlórendszereket felépíteni és működtetni, hogy a felhasználó számára releváns tartalom jelenjen meg.

PHP:
A PHP egy programozási nyelv, amit weboldalak készítésére használnak.

A PHP nyelven írt program a weboldalt üzemeltető távoli gépen fut le. Ilyenkor például elmenti egy látogató által kitöltött űrlap tartalmát, vagy kiolvassa egy adatbázisból, hogy a webáruház egyik kategóriájában milyen termékek vannak. A látogató ezután csak a program futtatásának az eredményét kapja meg, ami általában egy megjelenített weboldal tartalma.

Maga a rövidítés eredetileg Personal Home Page Tools kifejezésből ered, később azonban az önhivatkozást tartalmazó PHP: Hypertext Preprocesszor lett a hivatalos neve.

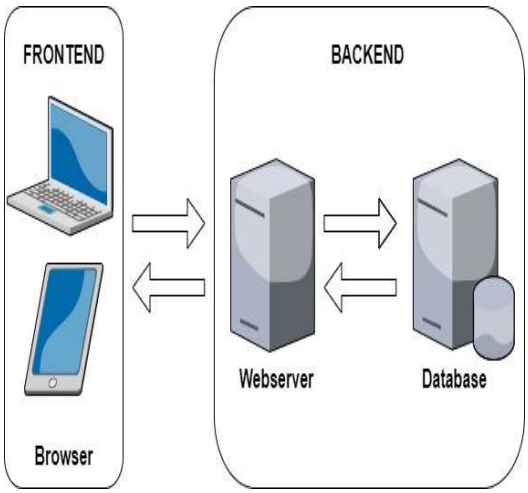
Mi a backend?
A backend a többrétegű architektúrának az a része, ami a frontend (prezentációs réteg) felől érkező

adatok és requestek feldolgozását végzi, illetve ami a szerver oldalon keletkező eredményeket

visszaadja a felhasználói felület felé.

Backend része az üzleti logika megvalósítása és adattárolási réteg elérése.

A backend réteg, tehát gyakorlatilag minden, ami nem frontend.



Legelterjettebb backend technológiák:
-Java
-PHP
-Python
-C#

Java:
Java az egyik legnépszerűbb programozási nyelv, amit már több mint 20 éve használnak a fejlesztők. Nagyon sokoldalú, platform független, általános célú objektumorientált programozási nyelv. Android alkalmazásokra és webes alkalmazásokra is használható, valamint képfeldolgozásra, adatelemzésre, játékfejlesztésre és szinte bármi másra is.

Népszerű Java backend keretrendszerek:
spring, struts, hibernate

Népszerű weboldalak, amik Java backenddel készültek:
Twitter, LinkedIn, eBay, de a NASA is előszeretettel használja.

PHP
Általános szerveroldali szkriptnyelv. Fő felhasználási területe: dinamikus weblapok készítése.

Emellett parancssorból is futtatható, továbbá kliens oldali alkalmazások is készíthetők segítségével.

Mára a weblapok igen jelentős része használ PHP-t. Ez az arány 60-70% körül mozog.

A PHP használható dinamikus oldalak létrehozására, űrlapok kezelésére, cookie-k kezelésére,

szerveroldali parancsfájlok készítésére és akár még asztali alkalmazások fejlesztésére is.

Népszerű PHP backend keretrendszerek:

laravel, codeigniter, symfony.

Az alábbi website-ok használnak például PHP backendet:

Facebook, Wikipedia, Tumblr, Slack, DocuSign, WordPress, Yahoo, stb.

Python:
Python a legerőteljesebben növekvő népszerűséggel bíró programozási nyelv. Sokoldalú programozási nyelv. Szintaxisa egyszerű és érthető. Elegáns, olvasható kód jellemzi. Nyílt forráskódú, nagyon magas szintű programozási nyelv.

A Python használható webfejlesztésre, gépi tanulásra, mesterséges intelligencia fejlesztésre, játékfejlesztésre, asztali alkalmazások fejlesztésére, még akár webcrawler fejlesztésre is.

Népszerű Python backend keretrendszerek:

Django, flask.

Python backend található az alábbi weboldalaknál:
Mozilla, Spotify, Pinterest, továbbá használja a Google, Facebook és Youtube is.

C#
A C# a .NET keretrendszer egyik leghasználtabb objektumorientált programozási nyelve. Korábban a .NET alkalmazások csak Windows alatt voltak elérhetőek, azonban 2016-ban kiadásra került a .NET Core.

A .NET Core megjelenésével a Microsoft kiterjesztette a .NET alkalmazhatóságát MacOS és Linux rendszerekre is.

C# backend található az alábbi weboldalaknál:

MSN, Bing, StackOverflow, GoDaddy.

Adatbázis:
Az adatbázis (**DB**: *Database*) számítógépen (általában háttértárakon) tárolt adatok összessége. Az adatbázist egy adatbázis kezelő rendszer (**DBMS** : *Database Management System*) segítségével használhatjuk. Nem minden (számítógépen tárolt) adathalmazt tekintünk adatbázisnak. A következőket szokás elvárni egy modern adatbázis kezelő rendszertől:

- az adatok valamilyen rendszer szerint vannak tárolva
- a rendszer képes (nagyon) sok adat tárolására
- az adatfelvitel, adatmódosítás és adatlekérdezés gyorsan elvégezhető
- az adatok párhuzamosan (osztottan) is elérhetők
- az adatok kezelése biztonságos
    - szabályozhatók a hozzáférési jogosultságok
    - biztonságosak a tranzakciók osztott hozzáférés esetén
    - **hatékonyan menthetők az adatok**

Relációs adatbázisok:
Az 1970-es években alkotta meg E. F. Codd a relációs adatmodell alapjait. Napjainkban ez az egyik legelterjedtebb modell, amiben alkalmazások milliói futnak relációs adatbázis kezelő rendszereken (**RDBMS**: *Relational Database Management System*).

Példák adatbázisok használatára:

Azért nehéz példákat adni adatbázisok használatára, mert szinte *minden* hálózati alkalmazása (akár az Interneten, akár a helyi hálózaton) használ valamilyen adatbázist szolgáltatásai megvalósításához.

Néhány jellegzetes alkalmazás:

- bankok, folyószámlák, tőzsde, elektronikus kereskedelem
- menetrendek
- digitális térképek
- filmek
- alkalmazás boltok
  közösségi hálózatok

Adatbázisok definíciói, típusai és példái:

Mik azok az adatbázisok?

Minimális definícióként azt mondhatjuk, hogy az adatbázis egymáshoz kapcsolódó információk bármilyen gyűjteménye. Ha egy bevásárlólistát írunk fel egy papírlapra, máris egy kis analóg adatbázist hozunk létre. De mit nevezünk adatbázisnak a számítástudományban? Itt az adatbázist úgy definiáljuk, mint információk olyan gyűjteményét, amelyet adatokként tárolunk egy számítógépes rendszeren. Ilyen lehet többek között a környékbeli szupermarket készletleltára is.

Mire használhatók az adatbázisok?

Az adatbázisokat arra használjuk, hogy az adatokat úgy tárolják és szervezzék, hogy azokat egyszerűen lehessen kezelni és elérni. Ahogy az adatok gyűjteménye növekszik és bonyolultabbá válik, úgy lesz egyre nehezebb az is, hogy az adatokat megfelelően rendszerezzük, hogy elérjük őket és hogy gondoskodjunk a biztonságukról is. Ebben a feladatban segít az adatbázis-kezelő rendszer (angolul „database management system", betűszóval DBMS), amely felügyeleti eszközöket tartalmaz.

Mit jelent az adat?

Adatnak nevezünk bármely olyan információt, amelyet rögzítünk és tárolunk például egy személyről, egy tárgyról, egy objektumról (ezeket entitásoknak hívjuk), de adatnak nevezzük az entitások jellemzőit (azaz attribútumait) is.

Ha például a helyi éttermekről rögzítünk és tárolunk információkat, akkor minden egyes étterem egy entitás lesz, és az entitás attribútumai lesznek az étterem neve, a címe és a nyitva tartási ideje. Minden információ adat, amit a kedvenc éttermünkről összegyűjtünk és tárolunk.

Az adatbázisok típusai

Az adatbázisokat nagy vonalakban relációs és nem relációs típusokra oszthatjuk. A relációs adatbázisok nagy mértékben strukturáltak, és értelmezni tudják a Structured Query Language (magyarul strukturált lekérdezőnyelv), azaz az SQL nevű programnyelvet. A nem relációs adatbázisok rendkívül változatosak, és számos különböző adatstruktúrát támogatnak. Számos nem relációs adatbázis nem használ SQL-t, ezért ezeket gyakran NoSQL-adatbázisoknak is nevezik.

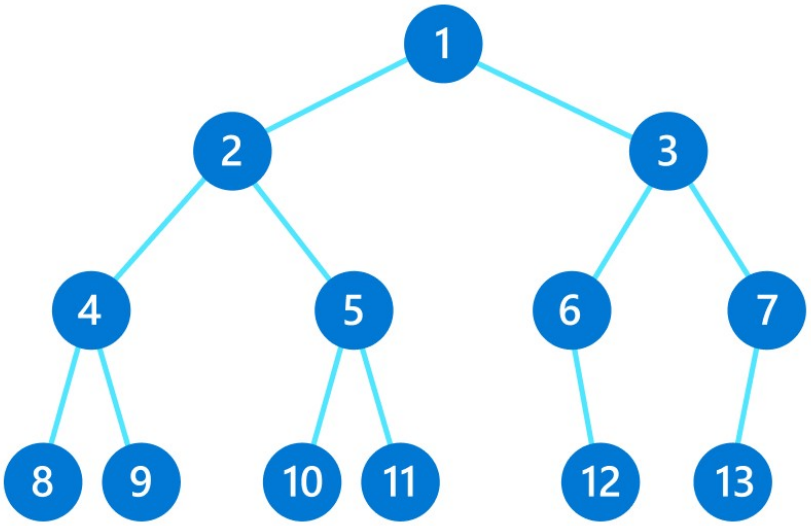
Az adatstruktúrák típusai

A **táblastruktúrák** relációs adatbázisstruktúrák, amelyek sorokba és oszlopokba rendezik az adatokat. A sorok az entitásokat tartalmazzák, az oszlopok pedig az entitások attribútumait. A **széles táblák** vagy **széles oszlopos tárak** üres attribútumokkal rendelkező ritka oszlopokat használnak, amivel jelentősen megnövelik a táblában elhelyezhető oszlopok teljes számát. Mivel ezeknél bizonyos helyek üresen maradnak, a széles táblákat nem relációs adatbázis-struktúráknak is tekinthetjük.

A lineáris struktúrák egy szekvenciába rendezik az elemeket.

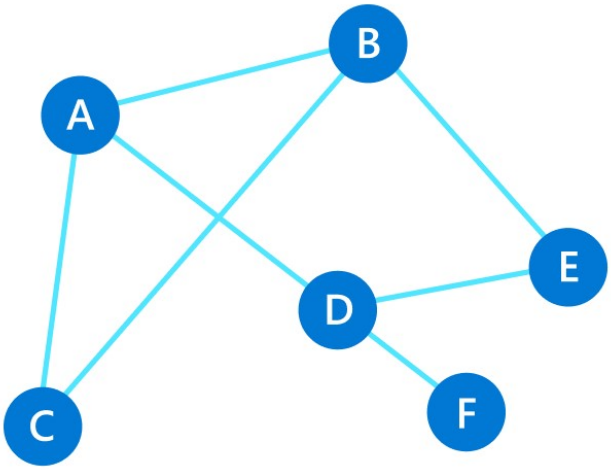
Bináris fa:

A **fastruktúrák** az elemeket a csomópontok hierarchikus adatbázisába szervezik olyan szülő-gyermek kapcsolatokba, amelyek egyetlen gyökércsomópontból származnak.



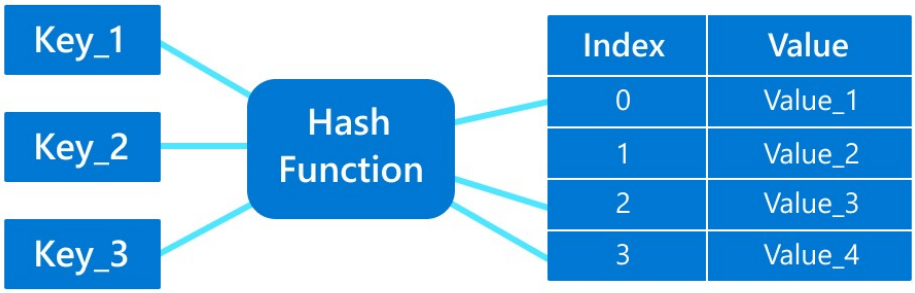
Gráf:

A **gráfstruktúrák** az elemeket egy nem hierarchikus csomóponthálózatba szervezik, amelyben az elemek egymással összetett kapcsolatokkal rendelkeznek.



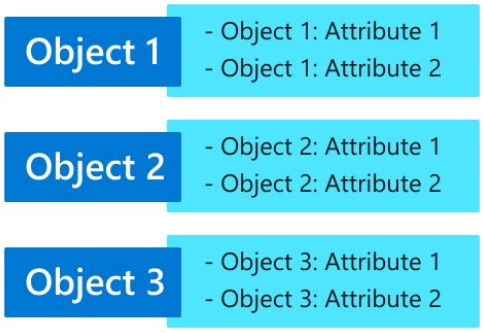
Kivonattábla:

A **kivonatalapú** struktúrák kulcsokat képeznek le értékekre olyan kivonatfüggvények használatával, amelyek a kapcsolódó adatokat úgy társítják egymáshoz, hogy indexeket rendelnek a kivonattáblákhoz.



Dokumentumorientált adatbázisok:

A **dokumentumorientált adatbázisok** az egy entitással kapcsolatos információmennyiséget egyetlen objektumba szervezik (a dokumentumba), amely elkülönül magától az objektumtól. Az objektumokat nem kell egymáshoz viszonyítva leképezni, és egyetlen objektum szerkeszthető anélkül, hogy ez hatással lenne a többi objektumra.



Relációs adatbázisok:

A relációs adatbázis a leggyakoribb típus, amelyben az adatok olyan táblákba vannak szervezve, amelyek minden egyes entitásról információkat tárolnak, és előre meghatározott kategóriákat tartalmaznak sorokban és oszlopokban. Ezek a strukturált adatok egyszerre hatékonyak és rugalmasan elérhetők.

Relációs adatbázis például az SQL Server, az Azure SQL, a MySQL, a PostgreSQL és a MariaDB.

Nem relációs adatbázisok:

A nem relációs adatbázisok strukturálatlan és félig strukturált adatokat tárolnak. A relációs adatbázisokkal ellentétben nem használnak oszlopokat és sorokat tartalmazó táblákat. Ehelyett olyan tárolási modellt használnak, amely a tárolt adatok típusának speciális követelményeire van optimalizálva. A nem relációs adatbázisok lehetővé teszik, hogy gyorsan lehessen elférni, frissíteni és elemezni elosztott adatok nagyobb csoportját.

Nem relációs adatbázisok például a MongoDB, az Azure Cosmos DB, a DocumentDB, a Cassandra, a Couchbase, a HBase, a Redis és a Neo4j.

Bizonyos nem relációs adatbázisokat NoSQL-adatbázisoknak nevezünk. A NoSQL itt arra utal, hogy a lekérdezésekhez nem használunk SQL-t, vagy nem csak SQL-t használunk hozzájuk. A NoSQL-adatbázisok ehelyett más programnyelveket és szerkezeteket használnak az adatok lekérdezéséhez. Ezzel együtt azonban számos NoSQL-adatbázis támogatja az SQL-kompatibilis lekérdezéseket is, ezeket a lekérdezéseket azonban nem úgy hajtják végre, mint ahogy a hagyományos relációs adatbázisok hajtják végre az SQL-lekérdezéseket.

A nem relációs adatbázisok egyik típusa – **az objektum-adatbázis** – objektumorientált programozást használ. Az objektumok olyan állapottal (tényleges adatokkal) vannak kódolva, amely egy mezőben vagy változóban vannak tárolva, és egy olyan viselkedéssel, amely egy metódussal vagy egy függvénnyel megjeleníthető. Az objektumokat lehetséges egy perzisztens tárhelyen örökre tárolni, és ezek közvetlenül beolvashatók és leképezhetők API vagy eszköz használata nélkül, ami miatt az adatok gyorsabban érhetők el, és jobb lesz a teljesítmény is. Az objektum-adatbázisok azonban kevésbé népszerűek, mint más adatbázistípusok, és a támogatásuk is nehézkes lehet.

Memóriabeli adatbázisok és gyorsítótárak:

A memóriabeli adatbázisban található összes adat a számítógép RAM-jában van tárolva. Az ilyen típusú adatbázis lekérdezésénél vagy frissítésénél közvetlenül a főmemóriát érjük el. Nincsenek lemezműveletek. Az adatok gyorsan betöltődnek, mert a főmemória elérése (amely az alaplapon a processzor közelében található) sokkal gyorsabb, mint a lemezek elérése.

A memóriabeli adatbázisokat leginkább a gyakran használt információk, például díjszabás vagy leltáradatok tárolására használják. Ezt gyorsítótárazásnak hívjuk. Amikor gyorsítótárazzuk az adatokat, akkor az adat egy példányát egy ideiglenes helyen tároljuk, így az a következő lekéréskor gyorsabban betöltődik.

Adatbázispéldák:

Az adatbázisok olykor láthatatlan rejtélyeknek tűnhetnek, a legtöbben azonban nap mint nap használjuk őket. Az alábbiakban néhány példát mutatunk be relációs adatbázisokra, NoSQL-adatbázisokra és memóriabeli adatbázisokra:

- Pénzügyi tranzakciók
- E-kereskedelmi katalógusok
- Közösségi hálózatok
- Személyreszabott eredmények
- Üzleti elemzés

Adatbázis-kezelő rendszerek:

Az adatbázisgazdák adatbázis-kezelő rendszerekkel (DBMS) vezérlik az adatokat – különösen akkor, ha **big data** típusú adatokkal dolgoznak. A big data olyan nagy mennyiségű strukturált és strukturálatlan adatot jelent, amelyet a rendszer gyakran valós időben vagy csaknem valós időben

fogad. A DBMS abban is segít, hogy olyan adatokat kezeljünk, amelyeket több alkalmazás is használ, vagy amelyek több helyen vannak tárolva.

A különböző felügyeleti rendszerek különböző szervezhetőségi, skálázhatósági és alkalmazásszinteket kínálnak. A rendszerezni kívánt adatok típusa és elérésének módja mellett a használandó DBMS attól is függ, hogy hol találhatók az adatok, hogy milyen típusú architektúrát használ az adatbázis, és hogy Ön hogyan tervezi a skálázást.

Források: https://lexiq.hu
https://codeberryschool.com
https://bluebird.hu/
http://info.berzsenyi.hu/adatbazisok/az-adatbazis-fogalma
https://azure.microsoft.com/hu-hu/resources/cloud-computing-dictionary/what-are-databases/