Régi anyag:
https://docs.google.com/presentation/d/1_nmA1F4ag_O-qlJAE4TfVuyfLgSruZLW/edit?usp=sharing&ouid=110539736176923279178&rtpof=true&sd=true

Adatbázisok története és fejlődése

Relációs adatbázisoktól a NoSQL-en át a vektor adatbázisokig

1. Bevezetés

Az adatbázisok az informatikai rendszerek alapvető elemei, amelyek lehetővé teszik az adatok hatékony tárolását, visszakeresését, rendezését és módosítását. A modern világban hatalmas mennyiségű adat keletkezik másodpercenként, és ezek strukturált tárolása elengedhetetlen a döntéstámogatáshoz, automatizáláshoz és gépi tanuláshoz.

Miért van szükség adatbázisokra?

- Az adatok szervezett tárolása lehetővé teszi azok gyors visszakeresését és elemzését.
- Több felhasználó egyidejűleg dolgozhat ugyanazon adatokkal (konkurens hozzáférés).
- Biztosítani lehet az **adatintegritást** és a **hozzáférés-ellenőrzést**.
- Az adatok rendszeres **mentése és visszaállítása** egyszerűbbé válik.
- Automatizált folyamatokhoz (pl. webalkalmazások, gépi tanulás) szükséges egy megbízható háttértároló.

Példa: Egy webshopban több ezer termék, rendelés és felhasználói adat van. Ezek kereshető tárolása adatbázis nélkül gyakorlatilag lehetetlen lenne.

Adat vs. információ

- **Adat**: nyers tények, mért vagy rögzített értékek, amelyek még nem feltétlenül bírnak jelentéssel.
  - Példa: `42`, `2025-05-07`, `„Piros”`
- **Információ**: értelmezett adat, amely kontextusba helyezve új tudást jelent.
  - Példa: „42 éves a felhasználó”, vagy „A rendelés dátuma: 2025-05-07”

Az adat tehát az alap, amiből – megfelelő feldolgozással – információt nyerhetünk. Az adatbázis célja az adatok strukturált tárolása annak érdekében, hogy információvá alakíthassuk őket.

Strukturált, félstrukturált és strukturálatlan adatok

- **Strukturált adat**:
  - Táblázatos formában rendezett, előre meghatározott sémával rendelkezik.
  - Példa: SQL adatbázisok, Excel-táblák, CRM rendszerek.
  - Jellemző: gyors kereshetőség, jól lekérdezhető.

- **Félstrukturált adat**:
- Nincs fix sémája, de tartalmaz címkéket vagy metaadatokat.
- Példa: XML, JSON, YAML fájlok, logok.
- Rugalmasabb, de nehezebb indexelni és lekérdezni.

- **Strukturálatlan adat**:
    - Nincs formázása, nem egyértelmű a tartalma.
    - Példa: szövegfájlok, képek, videók, e-mailek, hangfelvételek.
    - Ezeket általában keresőmotorokkal vagy mesterséges intelligenciával lehet feldolgozni (pl. szöveg- vagy képelemzés).

Az adatbázisok fejlődése szorosan összefügg azzal, hogyan és milyen típusú adatokat akarunk hatékonyan kezelni.

2. Relációs adatbázisok (RDBMS)

A relációs adatbázisok az egyik legelterjedtebb adatbázis-típusok közé tartoznak. Az adatok táblákban (relációkban) tárolódnak, ahol minden tábla sorokból (rekordokból) és oszlopokból (mezőkből) áll. A relációs modell megbízható és jól definiált struktúrát biztosít az adatok kezeléséhez.

Történet

- A relációs modellt **Edgar F. Codd** matematikus javasolta 1970-ben.
- A modell a halmazelméletre és a predikátumlogikára épül.
- A ’70-es, ’80-as években megjelentek az első RDBMS rendszerek: IBM System R, Ingres, Oracle.
- A **SQL (Structured Query Language)** nyelv szabványosítása nagyban hozzájárult az elterjedéséhez.

Jellemzők

- **Adatszerkezet**: táblázatos (sorok és oszlopok).
- **Szigorú séma**: minden táblának előre meghatározott szerkezete van (mezőtípusok, kulcsok stb.).
- **Kapcsolatok**: táblák között idegen kulcsokon keresztül hozunk létre kapcsolatokat.
- **ACID tulajdonságok**:
    - *Atomicity* – a tranzakciók oszthatatlanok.
    - *Consistency* – az adatbázis érvényességi szabályai nem sérülnek.
    - *Isolation* – párhuzamos tranzakciók nem befolyásolják egymást.
    - *Durability* – a végrehajtott tranzakciók tartósan megmaradnak.

Előnyök

- **Adatintegritás**: kulcsok, megszorítások és szabályok segítségével biztosítható.
- **Lekérdezhetőség**: a SQL nyelv rendkívül erős és kifejező eszköztárat ad.
- **Stabilitás és érettség**: hosszú távon kipróbált, optimalizált rendszerek.
- **Többfelhasználós működés**: jogosultságkezelés, tranzakciókezelés.

Hátrányok

- **Skálázhatóság**: nagy adatmennyiség és sok párhuzamos felhasználó esetén nehezen osztható szét több gépre (horizontális skálázás).
- **Rugalmatlanság**: séma módosítása bonyolult lehet, különösen ha az adatstruktúra gyakran változik.
- **Strukturálatlan adatok kezelése**: szöveg, kép, videó típusú adatok kezelésére nem ideális.

Népszerű relációs adatbázis-kezelő rendszerek (RDBMS-ek)

- **MySQL** – nyílt forráskódú, népszerű webalkalmazások körében (pl. WordPress).
- **PostgreSQL** – fejlett funkciók, szabványos SQL támogatás, objektum-orientált kiegészítések.
- **Oracle Database** – robusztus vállalati megoldás, fejlett biztonság és replikációs lehetőségek.
- **Microsoft SQL Server** – integráció Microsoft-technológiákkal, könnyű használat.

Példa: Egyszerű SQL tábla és lekérdezés

Tábla: `diakok`

id	nev	szuletesi_ev
1	Kovács Anna	2003
2	Nagy Péter	2002

SQL lekérdezés:

```
SELECT nev FROM diakok WHERE szuletesi_ev < 2003;
```

Ez a lekérdezés visszaadja azoknak a diákoknak a nevét, akik 2003 előtt születtek.

3. NoSQL adatbázisok

A NoSQL adatbázisok a relációs modellek alternatívájaként jelentek meg, amikor az internetes alkalmazások, a Big Data és a valós idejű feldolgozások új követelményeket támasztottak az adatkezeléssel szemben. A „NoSQL” nem feltétlenül jelenti azt, hogy nem használható benne SQL, hanem inkább azt, hogy nem relációs modell alapján működik.

Miért jelentek meg?

- Az adatok mennyisége gyorsan növekedett (pl. közösségi média, szenzoradatok).
- Az adatszerkezet egyre rugalmasabb lett – gyakran változó mezők, új attribútumok.
- Szükség volt **horizontális skálázásra** – sok szerver párhuzamos futtatása.
- Alkalmazásoknak nagy sebességre és alacsony késleltetésre volt szüksége.

Jellemzők

- **Sémanélküli modell**: nincs előre rögzített mezőszerkezet.
- **BASE modell**:
  - *Basically Available* – mindig elérhető adatbázis
  - *Soft state* – az adat idővel változhat
  - *Eventually consistent* – idővel elérhető a konzisztencia (nem garantált azonnal)
- **Horizontálisan skálázható**: könnyen bővíthető több gépre.
- **Nagy teljesítmény**: ideális valós idejű alkalmazásokhoz.

NoSQL típusok és példák

- **Dokumentum-orientált adatbázisok**
  - Minden rekord egy önálló dokumentum (pl. JSON formátumban).
  - Rugalmas szerkezet, ideális REST API-khoz.
  - Példa: **MongoDB**, CouchDB
- **Kulcs-érték tárolók**
  - Egyszerű szerkezet: kulcs → érték (mint egy szótár).
  - Nagyon gyors, kiváló gyorsítótárazásra.
  - Példa: **Redis**, Amazon DynamoDB
- **Oszlop-orientált adatbázisok**
  - Az adatok nem sorokban, hanem oszlopokban tárolódnak.
  - Hatékony analitikai feldolgozás nagy adattáblákon.
  - Példa: **Apache Cassandra**, HBase
- **Gráf adatbázisok**
  - Adatok és azok kapcsolatai gráfként ábrázolva (csomópontok és élek).
  - Nagyon hasznos hálózati elemzésekhez, pl. közösségi hálók, útvonalkeresés.
  - Példa: **Neo4j**, ArangoDB

Előnyök

- Nagy teljesítmény, alacsony válaszidő.
- Könnyen skálázható horizontálisan.
- Rugalmas adatszerkezet – ideális gyorsan változó alkalmazásokhoz.
- Néhány esetben beépített replikáció, sharding, elosztott feldolgozás.

Hátrányok

- Nincs szabványos nyelv (mint az SQL).
- Az adatintegritás és konzisztencia nem mindig garantált.
- Alkalmazásfejlesztéskor a logika egy része az adatbázis helyett a kódban jelenik meg.
- Komplex lekérdezések megvalósítása nehézkes lehet.

Példa: MongoDB dokumentum

Egy MongoDB kollekcióban tárolt dokumentum példája:

```
{
  "_id": "u123",
  "nev": "Kiss János",
  "eletkor": 29,
  "cim": {
    "varos": "Budapest",
    "iranyitoszam": "1111"
  }
}
```

Egy lekérdezés: azokat a felhasználókat kérdezzük le, akik életkora nagyobb mint 25:

```
db.felhasznalok.find({ "eletkor": { "$gt": 25 } })
```

4. NewSQL adatbázisok

- **Cél**: Relációs modell + NoSQL skálázhatóság
- **Jellemzők**:
  - ACID + elosztott tranzakciók
- **Példák**:
  - Google Spanner
  - CockroachDB
  - VoltDB

5. Vektor adatbázisok

- **Motiváció**:
  - Gépi tanulás, jelentés szerinti keresés
  - Embeddingek (szövegek, képek, hangok reprezentációja)
- **Működés**:
  - Objektum → Vektor (embedding)
  - Közelségi keresés: k-nn (legközelebbi szomszédok)
- **Példák**:
  - FAISS (Facebook)
  - Milvus
  - Weaviate
  - Pinecone
  - Chroma (LLM-es RAG rendszerekhez)
- **Alkalmazások**:
  - Dokumentumkeresés (RAG)
  - Képkeresés
  - Ajánlórendszerek

6. Tendenciák és jövőkép

- Multimodális adatbázisok (szöveg + kép + struktúrált adatok)
- Hybrid keresés: SQL + vektoros keresés együtt
- AI-native rendszerek: LLM + adatbázis integráció (pl. LangChain)

7. Összefoglaló táblázat

Típus	Példa	Előnyök	Hátrányok
Relációs	PostgreSQL	Stabil, jól ismert	Nehezen skálázható
NoSQL	MongoDB	Rugalmas, horizontálisan skálázható	Nincs szigorú séma, konzisztencia lehet gyenge
NewSQL	CockroachDB	ACID + skálázhatóság	Új technológia, kisebb ökoszisztéma
Vektor	FAISS	Jelentésalapú keresés, ML támogatás	Nem SQL-alapú lekérdezések, új gondolkodásmód

8. Demó

- SQL lekérdezés vs. MongoDB lekérdezés
- Szöveg → embedding → hasonló dokumentumok keresése vektor alapján

From:
https://edu.iit.uni-miskolc.hu/ - **Institute of Information Science - University of Miskolc**

Permanent link:
https://edu.iit.uni-miskolc.hu/tanszek:oktatas:muszaki_informatika:adattarolas_adatbazisok?rev=1746600644

Last update: **2025/05/07 06:50**