ADATBÁZISOK

Készítette: Szalontai István
Dátum: 2025-12-08
Utolsó módosítás: 2025-12-10

BEVEZETÉS AZ ADATBÁZISOK VILÁGÁBA

MI AZ ADATBÁZIS?

Definíció

- **Adatbázis**: Strukturált adatok szervezett gyűjteménye
- **Célja**: adatok hatékony tárolása, kezelése és lekérdezése
- **Mindennapi életünk része:** iskolai rendszerek, közösségi média, online vásárlás

Miért van rá szükségünk?

- 📊 Nagy mennyiségű adat tárolása
- 🔍 Gyors keresés és visszakeresés
- 🔒 Biztonságos adatkezelés
- 👥 Többfelhasználós hozzáférés

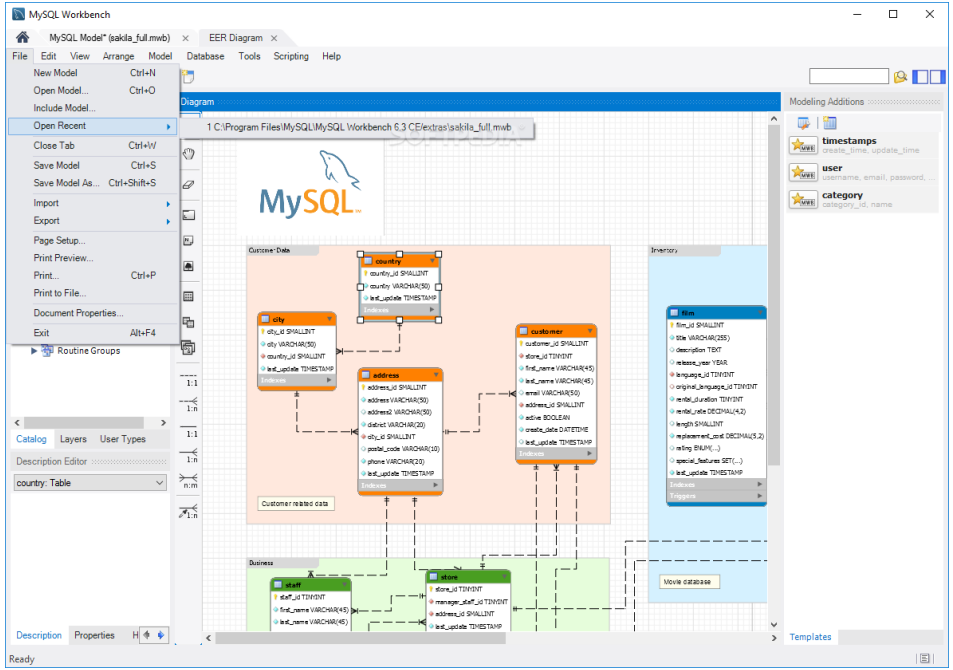
ADATBÁZIS VS. HAGYOMÁNYOS FÁJLOK

Fájlok problémái

- ❌ Nehéz adatfrissítés
- ❌ Adatismétlődés (redundancia)
- ❌ Nehéz keresés
- ❌ Biztonsági problémák

Adatbázis előnyei

- ✅ Központosított adatkezelés
- ✅ Adatintegritás
- ✅ Gyors lekérdezések
- ✅ Jogosultságkezelés



ADATBÁZIS-KEZELŐ RENDSZEREK (DBMS)

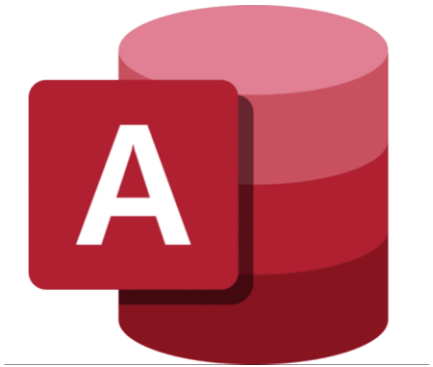
Mi az a DBMS?

Database Management System olyan szoftver, amely lehetővé teszi:

- Adatbázis létrehozását
- Adatok tárolását és módosítását
- Adatok lekérdezését
- Biztonság és hozzáférés kezelését

Népszerű DBMS-ek

- **Microsoft Access** - asztali alkalmazásokhoz, oktatáshoz
- **MySQL** - nyílt forráskódú, webalkalmazásokhoz
- **PostgreSQL** - fejlett, nyílt forráskódú
- **Microsoft SQL Server** - vállalati környezethez
- **Oracle** - nagy vállalatok választása
- **SQLite** - beágyazott rendszerekhez



MICROSOFT ACCESS

Mi az Access?

- **Microsoft Office** csomag része
- Asztali adatbázis-kezelő rendszer
- Vizuális, felhasználóbarát felület
- Ideális **tanuláshoz** és kis-közepes projektekhez

Főbb komponensek

- 📊 **Táblák** - adatok tárolása
- 🔍 **Lekérdezések** - adatok szűrése, rendezése
- 📝 **Űrlapok** - adatbevitel, szerkesztés
- 📄 **Jelentések** - nyomtatható kimutatások

ACCESS ELŐNYEI TANULÁSHOZ

Miért jó kezdőknek?

- ✅ **Vizuális tervezés** - táblák, kapcsolatok grafikusan
- ✅ **Egyszerű használat** - húzd-és-ejtsd felület
- ✅ **Integrált környezet** - minden egy helyen
- ✅ **SQL és vizuális mód** - mindkettőt lehet gyakorolni
- ✅ **Azonnali eredmény** - gyorsan látható, ami készül

Korlátok

- ❌ **Kapacitás** - max 2GB adatbázis méret
- ❌ **Egyidejű felhasználók** - körülbelül 10-20 fő
- ❌ **Platform** - csak Windows rendszeren
- ❌ **Web alkalmazások** - nehezebb integrálni

ACCESS VS. MÁS DBMS-EK

Access (Tanuláshoz, kis projektek)

- 🎓 Iskolai projektek, vizsgamunkák
- 📊 Kis vállalkozások nyilvántartásai
- 🏠 Házi adatbázisok (könyvtár, filmgyűjtemény)
- 👨‍🎓 Diákok első adatbázis-projekt

MySQL/PostgreSQL (Webfejlesztés)

- Weboldalak, webalkalmazások
- Mobilalkalmazások háttere
- Nagy forgalmú rendszerek

SQL Server/Oracle (Nagyvállalati)

- 🏢 Bankok, biztosítók
- 🏥 Kórházak, kormányzati rendszerek
- 💼 Nemzetközi vállalatok ERP rendszerei

ACCESS HASZNÁLATA ÓRÁN

Mit fogunk csinálni?

1. **Táblák létrehozása** - vizuális tervezővel
2. **Kapcsolatok definiálása** - kapcsolatok nézet
3. **Adatok bevitele** - űrlapok készítése
4. **Lekérdezések** - SQL és vizuális lekérdezéstervező
5. **Jelentések** - szép kimutatások készítése

Gyakorlati projektek

- 📚 Iskolai könyvtár nyilvántartás
- 🎬 Film/sorozat adatbázis
- 🎮 Játékgyűjtemény
- 👥 Diák/tanár adatbázis

RELÁCIÓS ADATBÁZISOK (RDB)

Alapfogalmak

- **Tábla (Table)**: adatok tárolására szolgáló struktúra
- **Sor (Record/Tuple)**: egy adatrekord
- **Oszlop (Column/Attribute)**: egy tulajdonság
- **Mező (Field)**: egy konkrét adat

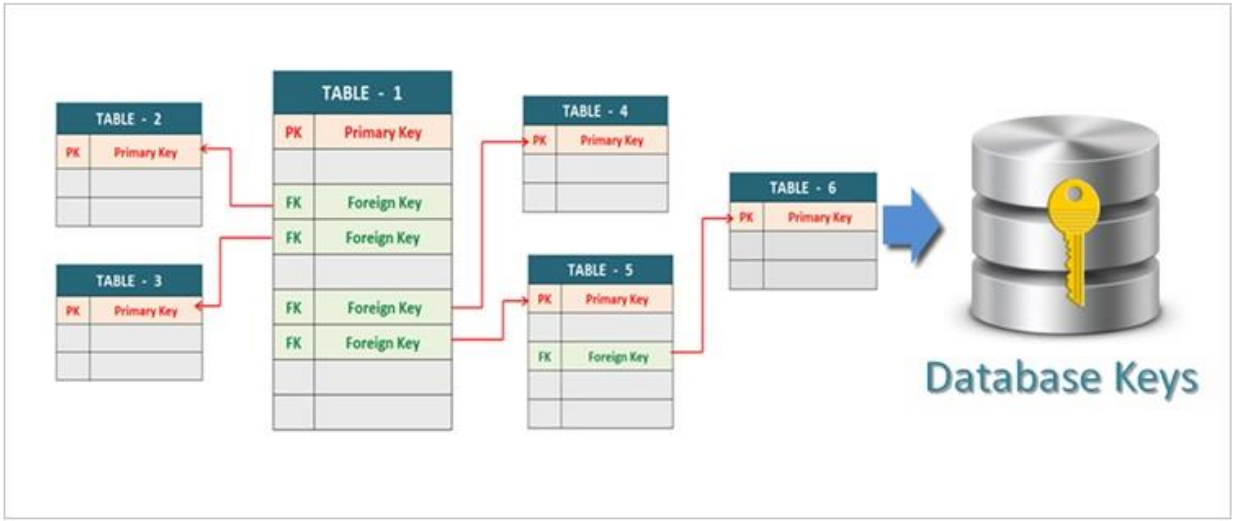
Példa: Diákok táblája

Diák_ID	Név	Osztály	Születési_év
1	Kiss Anna	10.A	2009
2	Nagy Péter	10.B	2010
3	Tóth Eszter	11.A	2008

KULCSOK AZ ADATBÁZISBAN

Elsődleges kulcs (Primary Key)

- Egyedileg azonosít minden rekordot
- Nem lehet NULL (üres)
- Nem ismétlődhet
- Példa: Diák_ID



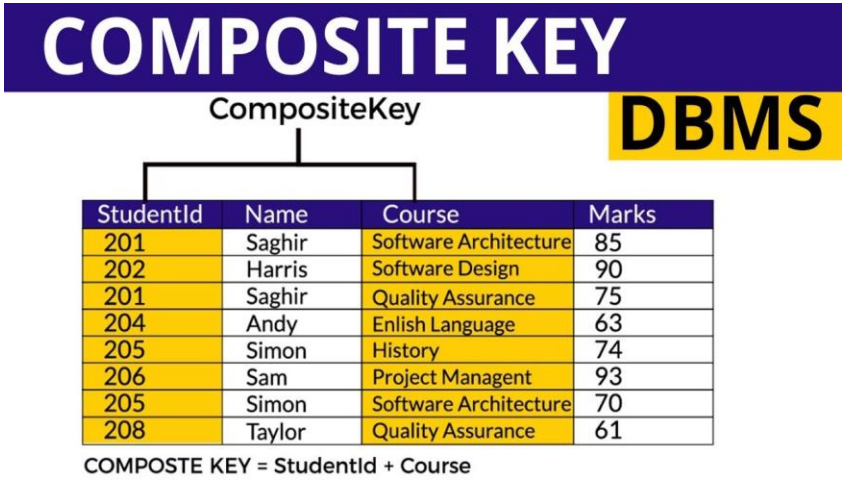
Idegen kulcs (Foreign Key)

- Kapcsolatot teremt két tábla között
- Egy tábla elsődleges kulcsára hivatkozik
- Biztosítja az adatok konzisztenciáját (következetes)

ÖSSZETETT KULCS (COMPOSITE KEY)

Mi az összetett kulcs?

- **Több oszlop együtt** alkotja az elsődleges kulcsot
- Egyenként ismétlődhetnek, de **együtt egyediek**
- Gyakran kapcsolótáblákban használatos



Példa: Diákok és Tantárgyak kapcsolótábla

Diáktantárgy tábla:

Diak_ID	Tantargy_ID	Jegy	Datum
1	101	5	2024-10-15
1	102	4	2024-10-20
2	101	5	2024-10-16

Elsődleges kulcs: (Diak_ID + Tantargy_ID)

- Egy diák többször szerepelhet (különböző tantárgyakkal)
- Egy tantárgy többször szerepelhet (különböző diákokkal)
- De egy diák + egy tantárgy kombináció **csak egyszer** szerepelhet

ÖSSZETETT KULCS PÉLDA SQL-BEN

Tábla létrehozása összetett kulccsal

```
-- MS Access / SQL általános szintaxis
CREATE TABLE Diaktantargy (
    Diak_ID INT,
    Tantargy_ID INT,
    Jegy INT,
    Datum DATE,
    PRIMARY KEY (Diak_ID, Tantargy_ID)
);
```

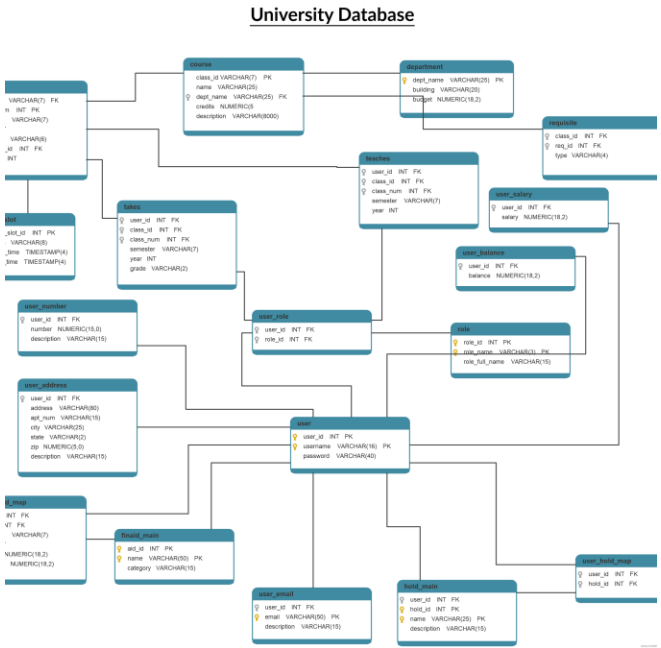
Mikor használjuk?

✅ **Több-több kapcsolatok** megvalósítása (diák-tantárgy, termék-rendelés)
✅ **Egyediség biztosítása** több oszlop alapján
✅ **Természetes kulcs** amikor nincs egyedi azonosító

Miért kerülik sokan?

❌ Bonyolultabb lekérdezések
❌ Több tárhely az idegen kulcs hivatkozásokban
❌ Nehezebb módosítani

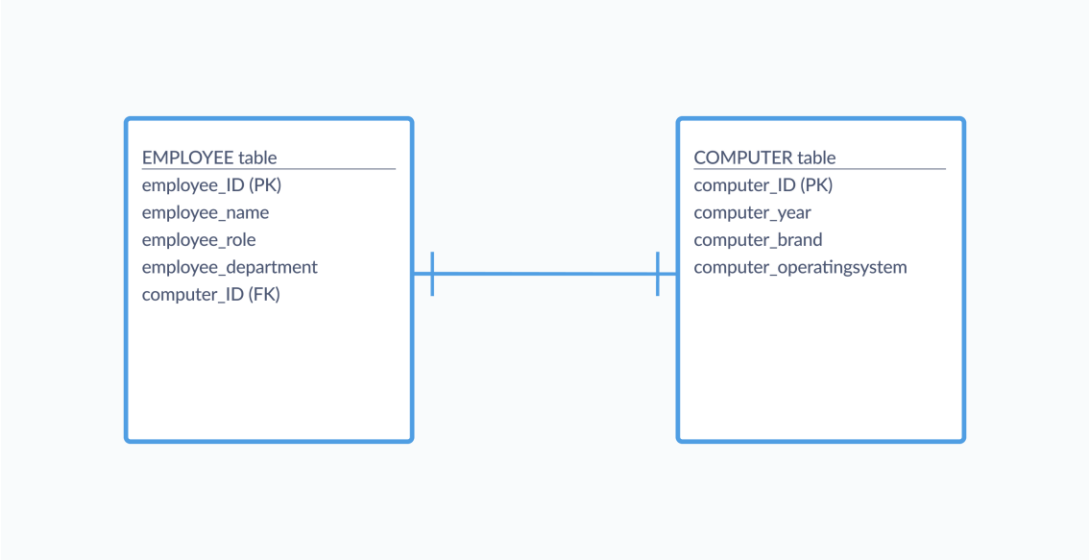
Alternatíva: Külön AutoNumber/ID mező + UNIQUE constraint a két oszlopra



TÁBLÁK KÖZÖTTI KAPCSOLATOK

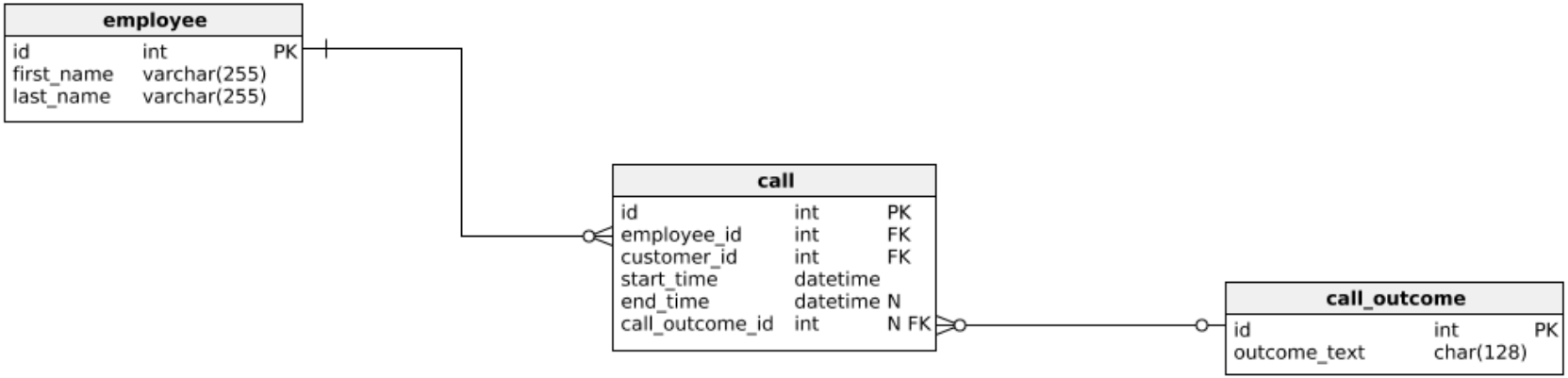
1. Egy-egy kapcsolat (1:1)

- Egy rekord pontosan egy másik rekordhoz kapcsolódik
- Példa: Egy diáknak egy tankártyája van



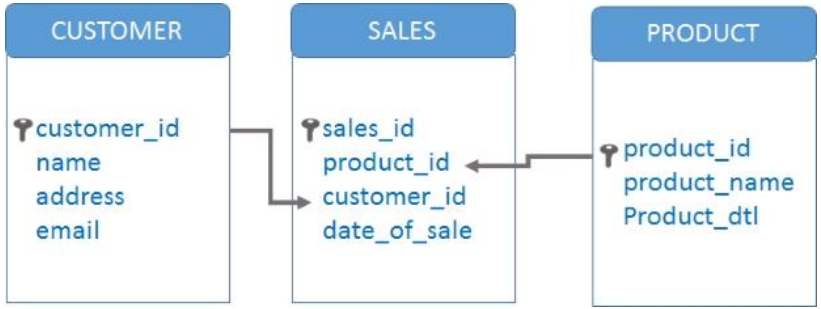
2. Egy-több kapcsolat (1:N)

- Egy rekord több másik rekordhoz kapcsolódik
- Példa: Egy tanár több diákot tanít
- Ez a leggyakoribb



3. Több-több kapcsolat (N:M)

- Több rekord több másik rekordhoz kapcsolódik
- Példa: Diákok és tantárgyak (egy diák több tantárgyat tanul, egy tantárgyat több diák tanul)



SQL - STRUCTURED QUERY LANGUAGE

Mi az SQL?

- Standard nyelv az adatbázisok kezeléséhez
- Könnyen tanulható, angol nyelvű parancsok
- Használható minden relációs adatbázisban

SQL főbb kategóriák

1. **DDL** (Data Definition Language) - szerkezet definiálása
2. **DML** (Data Manipulation Language) - adatok kezelése
3. **DQL** (Data Query Language) - adatok lekérdezése
4. **DCL** (Data Control Language) - jogosultságok kezelése

SQL DIALEKTUSOK - FONTOS KÜLÖNBSÉGEK

Mi az SQL dialektus?

- Az SQL **szabvány**, de minden DBMS kissé **másképp** implementálja
- Alapműveletek (SELECT, INSERT, UPDATE, DELETE) **mindenütt ugyanúgy** működnek
- **Speciális funkciók** eltérhetnek

Gyakori eltérések

Funkció	MS Access	MySQL/MariaDB	PostgreSQL
Első N rekord	`TOP N`	`LIMIT N`	`LIMIT N`
Automatikus ID	`AutoNumber`	`AUTO_INCREMENT`	`SERIAL`
Dátum formátum	`#2024-12-06#`	`'2024-12-06'`	`'2024-12-06'`
Helyettesítő karakter	`*` (szövegben)	`%`	`%`
String összefűzés	`&`	`CONCAT()`	`||`

TOP VS LIMIT - ELSŐ N REKORD

MS Access (TOP)

```
-- Első 5 diák
SELECT TOP 5 * FROM Diakok
ORDER BY Nev;

-- Első 3 legjobb átlag
SELECT TOP 3 Nev, AVG(Jegy) AS Atlag
FROM Diakok
GROUP BY Nev
ORDER BY Atlag DESC;
```

MySQL/PostgreSQL (LIMIT)

```sql
-- Első 5 diák
SELECT * FROM Diakok
ORDER BY Nev
LIMIT 5;

-- Első 3 legjobb átlag
SELECT Nev, AVG(Jegy) AS Atlag
FROM Diakok
GROUP BY Nev
ORDER BY Atlag DESC
LIMIT 3;
```

Mindkettő ugyanazt csinálja, csak más szintaxissal!

TOVÁBBI ACCESS-SPECIFIKUS JELLEMZŐK

Helyettesítő karakterek

```
-- Access: * és ?
SELECT * FROM Diakok WHERE Nev LIKE 'Kiss*';
-- bármi Kiss után
SELECT * FROM Diakok WHERE Nev LIKE 'An?a';
-- pontosan 1 karakter

-- MySQL/PostgreSQL: % és _
SELECT * FROM Diakok WHERE Nev LIKE 'Kiss%';
-- bármi Kiss után
SELECT * FROM Diakok WHERE Nev LIKE 'An_a';
-- pontosan 1 karakter
```

Dátumok kezelése

```
-- Access: # jelek között
SELECT * FROM Diakok WHERE Szuletesi_datum
= #2009-05-15#;

-- MySQL/PostgreSQL: idézőjelek között
SELECT * FROM Diakok WHERE Szuletesi_datum
= '2009-05-15';
```

String összefűzés

```
-- Access: & operátor
SELECT Vezeteknev & ' ' & Keresztnev AS
Teljes_nev FROM Diakok;

-- MySQL: CONCAT() függvény
SELECT CONCAT(Vezeteknev, ' ', Keresztnev)
AS Teljes_nev FROM Diakok;
```

CRUD MŰVELETEK

Mi a CRUD?

A CRUD az adatbázis-kezelés **4 alapműveletét** jelenti:

- **C**reate (Létrehozás) → `INSERT`
- **R**ead (Olvasás) → `SELECT`
- **U**pdate (Módosítás) → `UPDATE`
- **D**elete (Törlés) → `DELETE`

Miért fontos?

- Minden adatbázis-alkalmazás ezekre épül
- Az összes adatkezelési feladat e 4 műveletből áll
- Web és mobilalkalmazások alapja

Create

SQL ALAPMŰVELETEK - CREATE

Tábla létrehozása

```
CREATE TABLE Diakok (
    Diak_ID INT PRIMARY KEY,
    Nev VARCHAR(50),
    Osztaly VARCHAR(10),
    Szuletesi_ev INT
);
```

Mit jelent ez?

- Létrehozunk egy “Diakok” nevű táblát
- 4 oszloppal: ID, Név, Osztály, Születési év
- Diak_ID lesz az elsődleges kulcs

Create

SQL ALAPMŰVELETEK - INSERT (CREATE)

Adatok beszúrása

```
INSERT INTO Diakok (Diak_ID, Nev, Osztaly,
Szuletesi_ev)
VALUES (1, 'Kiss Anna', '10.A', 2009);
```

Több rekord egyszerre

```
INSERT INTO Diakok VALUES
     (2, 'Nagy Péter', '10.B', 2010),
     (3, 'Tóth Eszter', '11.A', 2008),
     (4, 'Szabó Máté', '10.A', 2009);
```

Read

SQL ALAPMŰVELETEK - SELECT (READ)

Összes adat lekérdezése

```
SELECT * FROM Diakok;
```

Specifikus oszlopok

```
SELECT Nev, Osztaly FROM Diakok;
```

Szűrés feltétellel (WHERE)

```
SELECT * FROM Diakok
WHERE Osztaly = '10.A';
```

Rendezés

```
SELECT * FROM Diakok
ORDER BY Nev;
```

ORDER BY - RENDEZÉS

Mire jó?

- Eredmények **sorba rendezése** egy vagy több oszlop alapján
- **Növekvő** (ASC) vagy **csökkenő** (DESC) sorrend
- Áttekinthetőbb, rendezett eredmények

Alapszintaxis

```
SELECT * FROM tabla
ORDER BY oszlop [ASC|DESC];
```

ASC = Ascending (növekvő) - **alapértelmezett**

DESC = Descending (csökkenő)

ORDER BY PÉLDÁK

Nevek ABC sorrendben (növekvő)

```
SELECT * FROM Diakok
ORDER BY Nev ASC;
-- vagy egyszerűen:
SELECT * FROM Diakok
ORDER BY Nev;
```

Diák_ID	Név	Osztály	Születési_év
1	Kiss Anna	10.A	2009
2	Nagy Péter	10.B	2010
4	Szabó Máté	10.A	2009
3	Tóth Eszter	11.A	2008

ORDER BY TÖBB OSZLOP

Osztály szerint, azon belül név szerint

```
SELECT * FROM Diakok
ORDER BY Osztaly ASC, Nev ASC;
```

Legfiatalabb diákok elöl (csökkenő születési év)

```
SELECT Nev, Osztaly,
Szuletesi_ev
FROM Diakok
ORDER BY Szuletesi_ev
DESC;
```

Név	Osztály	Születési_év
Nagy Péter	10.B	2010
Kiss Anna	10.A	2009
Szabó Máté	10.A	2009
Tóth Eszter	11.A	2008

ORDER BY GYAKORLATI PÉLDÁK

Top 3 legjobb átlag (jegyek alapján)

```sql
-- MySQL/PostgreSQL szintaxis
SELECT Diak_ID, Nev, AVG(Jegy) AS Atlag
FROM Diakok
JOIN Jegyek ON Diakok.Diak_ID =
Jegyek.Diak_ID
GROUP BY Diak_ID, Nev
ORDER BY Atlag DESC
LIMIT 3;
```

Top 3 legjobb átlag (jegyek alapján)

```
-- MS Access szintaxis
SELECT TOP 3 Diak_ID, Nev, AVG(Jegy) AS
Atlag
FROM Diakok
INNER JOIN Jegyek ON Diakok.Diak_ID =
Jegyek.Diak_ID
GROUP BY Diak_ID, Nev
ORDER BY Atlag DESC;
```

Osztály és életkor szerinti rendezés

```
SELECT Nev, Osztaly, (2025 - Szuletesi_ev)
AS Kor
FROM Diakok
ORDER BY Osztaly ASC, Kor DESC;
```

Tipp: Kombinálhatod a WHERE, GROUP BY, ORDER BY záradékokat!

SQL ALAPMŰVELETEK - UPDATE (UPDATE)

Adatok módosítása

```
UPDATE Diakok
SET Osztaly = '11.A'
WHERE Diak_ID = 1;
```

Figyelem!

- Mindig használj WHERE feltételt!
- WHERE nélkül MINDEN rekord módosul!

```
-- VESZÉLYES! Minden diák osztályát
megváltoztatja
UPDATE Diakok SET Osztaly = '12.A';
```

SQL ALAPMŰVELETEK - DELETE (DELETE)

Rekord törlése

```
DELETE FROM Diakok
WHERE Diak_ID = 4;
```

VIGYÁZAT! WHERE nélkül az ÖSSZES rekord törlődik!

```
-- VESZÉLYES! Kiüríti a teljes táblát
DELETE FROM Diakok;
```

ÖSSZETETTEBB LEKÉRDEZÉSEK

Szűrés több feltétellel

```
SELECT * FROM Diakok
WHERE Osztaly = '10.A' AND Szuletesi_ev =
2009;
```

Keresés mintára (LIKE)

```
-- Név 'Kiss'-sel kezdődik

SELECT * FROM Diakok
WHERE Nev LIKE 'Kiss%';
```

Számolás (COUNT, SUM, AVG)

```
SELECT COUNT(*) FROM Diakok WHERE Osztaly
= '10.A';

SELECT AVG(Szuletesi_ev) FROM Diakok;
```

GROUP BY - CSOPORTOSÍTÁS

Mire jó?

- Adatok **csoportokba rendezése** egy vagy több oszlop alapján
- **Összesítő függvények** használata csoportonként
- Statisztikák készítése kategóriánként

Alapszintaxis

```
SELECT oszlop, AGGREGÁLT_FÜGGVÉNY(oszlop)
FROM tabla
GROUP BY oszlop;
```

GROUP BY PÉLDÁK

Diákok száma osztályonként

```
SELECT Osztaly, COUNT(*)
AS Letszam
FROM Diakok
GROUP BY Osztaly;
```

Osztály	Létszám
10.A	2
10.B	1
11.A	1

Átlagéletkor osztályonként

```
SELECT Osztaly, AVG(2025 - Szuletesi_ev)
AS Atlageletkor
FROM Diakok
GROUP BY Osztaly;
```

GROUP BY + HAVING

Mi a különbség WHERE és HAVING között?

- **WHERE**: szűrés csoportosítás **ELŐTT** (egyedi sorokra)
- **HAVING**: szűrés csoportosítás **UTÁN** (csoportokra)

Példa: Csak azok az osztályok, ahol több mint 1 diák van

```
SELECT Osztaly, COUNT(*) AS Letszam
FROM Diakok
GROUP BY Osztaly
HAVING COUNT(*) > 1;
```

Komplex példa jegyekkel

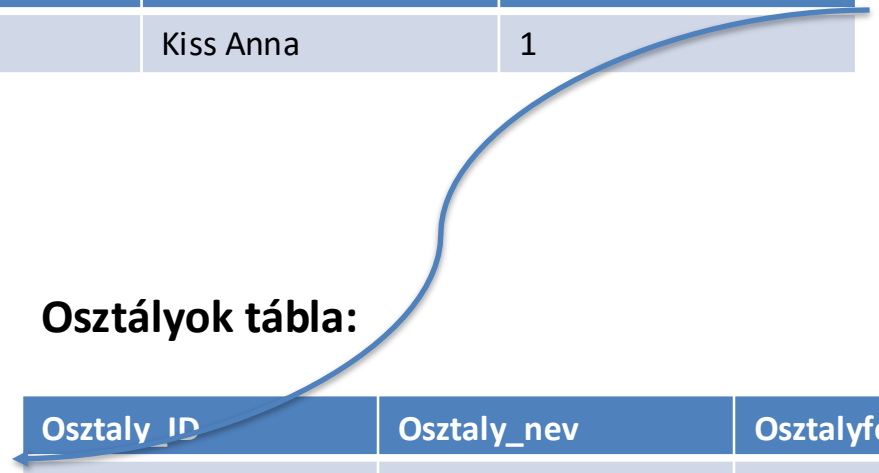
```
SELECT Diak_ID, AVG(Jegy) AS Atlag
FROM Jegyek
GROUP BY Diak_ID
HAVING AVG(Jegy) >= 4.0
ORDER BY Atlag DESC;
```

ÖSSZEKAPCSOLÁSOK (JOIN)

Példa adatstruktúra

Diakok tábla:

Diak_ID	Nev	Osztaly_ID
1	Kiss Anna	1



Osztályok tábla:

Osztaly_ID	Osztaly_nev	Osztalyfonok
1	10.A	Nagy Béla

Összekapcsolás WHERE-rel (régi módszer)

```
SELECT Diakok.Nev, Osztalyok.Osztaly_nev,
Osztalyok.Osztalyfonok
FROM Diakok, Osztalyok
WHERE Diakok.Osztaly_ID =
Osztalyok.Osztaly_ID;
```

INNER JOIN (modern módszer)

```
SELECT Diakok.Nev, Osztalyok.Osztaly_nev,
Osztalyok.Osztalyfonok
FROM Diakok
INNER JOIN Osztalyok ON Diakok.Osztaly_ID
= Osztalyok.Osztaly_ID;
```

Mindkettő ugyanazt az eredményt adja! Az INNER JOIN olvashatóbb és kifejezőbb.

ADATINTEGRITÁS

Mit jelent?

- Az adatok pontossága és konzisztenciája
- Hibás vagy ellentmondásos adatok megelőzése

Hogyan biztosítjuk?

1. **NOT NULL** - a mező nem lehet üres
2. **UNIQUE** - az érték nem ismétlődhet
3. **PRIMARY KEY** - egyedi azonosító
4. **FOREIGN KEY** - kapcsolatok érvényessége
5. **CHECK** - egyéni szabályok (pl. életkor > 0)

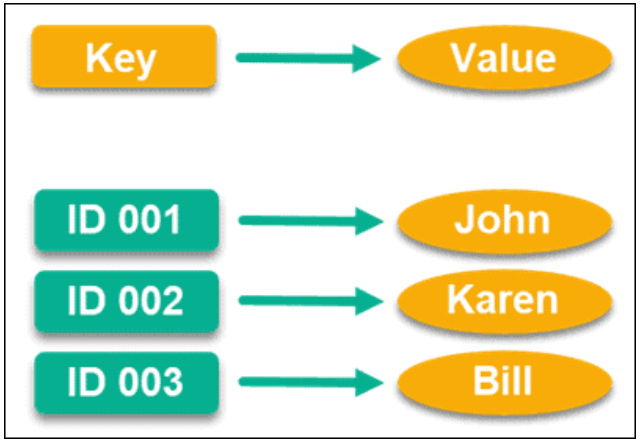
NOSQL ADATBÁZISOK

Mi a különbség?

- **Relációs**: strukturált táblák, SQL nyelv
- **NoSQL**: rugalmas séma, különböző adatmodellek

NoSQL típusok

1. **Dokumentum-alapú**: MongoDB, CouchDB
2. **Kulcs-érték**: Redis, DynamoDB
3. **Oszlop-alapú**: Cassandra, HBase
4. **Gráf**: Neo4j, Amazon Neptune



Mikor használjuk?

- Nagy mennyiségű, változatos adat
- Gyors skálázhatóság
- Rugalmas adatstruktúra

ADATBÁZIS-TERVEZÉS

Lépések

1. **Követelmények elemzése** - Mit akarunk tárolni?
2. **Konceptuális terv** - Entitások és kapcsolatok
3. **Logikai terv** - Táblák és mezők meghatározása
4. **Fizikai terv** - Implementáció

Normalizálás

- Adatismétlődés csökkentése
- Konzisztencia biztosítása
- Hatékonyabb tárolás
- 1NF, 2NF, 3NF normálformák

ADATBÁZIS-BIZTONSÁG

Fő területek

Hitelesítés - Ki vagy? (felhasználónév, jelszó)

Jogosultságkezelés - Mit csinálhatsz?

Titkosítás - Adatok védelme

Biztonsági mentés - Adatvesztés megelőzése

Naplózás - Ki mit csinált?

Alapelvek


Minimum jogosultság elve
Erős jelszavak
Rendszeres mentések
Frissítések telepítése

TRANZAKCIÓK

Mi az a tranzakció?

- Több művelet egyetlen egységként
- Vagy minden végrehajtódik, vagy semmi

ACID tulajdonságok

- **A**tomicity (Atomicitás) - Oszthatatlan egység
- **C**onsistency (Konzisztencia) - Érvényes állapotok
- **I**solation (Izoláltság) - Függetlenek egymástól
- **D**urability (Tartósság) - Megmarad a változás

Példa

Pénzátutalás: le kell vonni ÉS hozzá kell adni, nem csak az egyik!

WEB ALKALMAZÁSOK ARCHITEKTÚRÁJA

Hogyan működik egy webes adatbázis-alkalmazás?

A modern weboldalak **3 rétegű architektúrát** használnak:

1. **Kliens réteg** (Frontend) - böngésző, mobilalkalmazás
2. **Alkalmazás réteg** (Backend) - PHP, Python, Node.js
3. **Adatbázis réteg** - MySQL/MariaDB, PostgreSQL

Példa: webshop, közösségi oldal, online bank

HAGYOMÁNYOS WEB ARCHITEKTÚRA

Komponensek

- [Felhasználó Böngészője]
  ↕ **HTTP Request/Response**
  [Webszerver: Apache/Nginx]
  ↕
  [PHP Alkalmazás]
  ↕ **SQL lekérdezések**
  [MariaDB/MySQL Adatbázis]

Folyamat lépései

1.  Felhasználó megnyit egy weboldalt (pl. `webshop.hu/termekek`)
2.  Böngésző HTTP kérést küld a szervernek
3.  Webszerver (Apache/Nginx) fogadja a kérést
4.  PHP feldolgozza az oldalt, SQL lekérdezést küld
5.  MariaDB visszaadja az adatokat
6.  PHP generál HTML-t az adatokból
7.  Webszerver visszaküldi a HTML-t a böngészőnek
8.  Böngésző megjeleníti az oldalt

PÉLDA: TERMÉK LISTA LEKÉRDEZÉSE

1. Felhasználó kérése

```
GET http://webshop.hu/termekek.php
```

2. PHP kód (termekek.php)

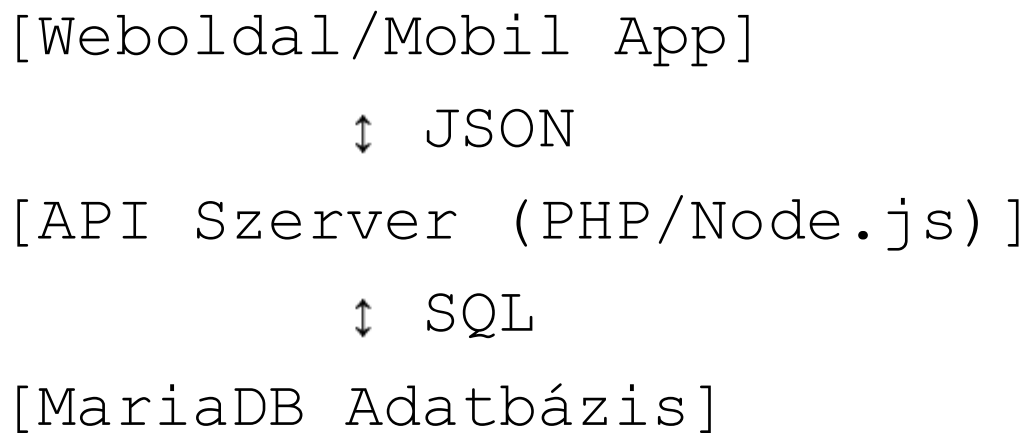
```php
<?php
// Adatbázis kapcsolat
$conn = new mysqli("localhost", "user", "pass", "webshop_db");

// SQL lekérdezés
$sql = "SELECT nev, ar, keszlet FROM termekek WHERE keszlet > 0";
$result = $conn->query($sql);

// HTML generálás
echo "<h1>Termékek</h1><ul>";
while($row = $result->fetch_assoc()) {
    echo "<li>{$row['nev']} - {$row['ar']} Ft</li>";
}
echo "</ul>";
?>
```

3. MariaDB válasz

```
Laptop - 250000 Ft
Egér - 5000 Ft
...
```

MODERN WEB API ARCHITEKTÚRA

RESTful API struktúra



Miért jobb?

- ✅ **Szeparáció** - Frontend és Backend külön fejleszthető
- ✅ **Többplatformos** - Azonos API web, mobil, desktop appoknak
- ✅ **JSON formátum** - Modern, strukturált adatcsere
- ✅ **Skálázhatóság** - Könnyebb terheléselosztás

API PÉLDA: TERMÉKEK LEKÉRDEZÉSE

1. API kérés (Frontend)

```
// JavaScript (böngészőben fut)
fetch('http://webshop.hu/api/termekek')
    .then(response => response.json())
    .then(data => {
        data.forEach(termek => {
            console.log(termek.nev, termek.ar);
        });
    });
```

2. API válasz (JSON formátum)


```
[
    {"id": 1, "nev": "Laptop", "ar": 250000,
"keszlet": 5},
    {"id": 2, "nev": "Egér", "ar": 5000,
"keszlet": 20},
    {"id": 3, "nev": "Billentyűzet", "ar": 15000,
"keszlet": 10}
]
```

API BACKEND - PHP PÉLDA

API endpoint (api/termekek.php)

```php
<?php
header('Content-Type: application/json');

// Adatbázis kapcsolat
$conn = new mysqli("localhost", "user", "pass", "webshop_db");

// SQL lekérdezés
$sql = "SELECT id, nev, ar, keszlet FROM termekek WHERE keszlet > 0";
$result = $conn->query($sql);

// Adatok JSON formátumba
$termekek = array();
while($row = $result->fetch_assoc()) {
    $termekek[] = $row;
}

// JSON válasz küldése
echo json_encode($termekek);
?>
```

Mit csinál?

1. JSON formátumú választ küld
2. Adatbázisból lekérdezi a termékeket
3. PHP tömbként tárolja
4. JSON-né alakítja (`json_encode`)
5. Visszaküldi a kliensnek

REQUEST-RESPONSE FOLYAMAT RÉSZLETESEN

Példa: Bejelentkezés 1/2

1. Kliens kérés (HTTP POST)

```
POST /api/login HTTP/1.1
Host: webshop.hu
Content-Type: application/json

{
    "email": "kiss.anna@email.hu",
    "password": "titkos123"
}
```

Példa: Bejelentkezés 2/2

2. PHP feldolgozás

```php
$email = $_POST['email'];
$password = $_POST['password'];

// Biztonságos jelszó hash
$sql = "SELECT id, nev FROM felhasznalok
        WHERE email = ? AND password_hash = SHA2(?, 256)";
```

3. MariaDB válasz

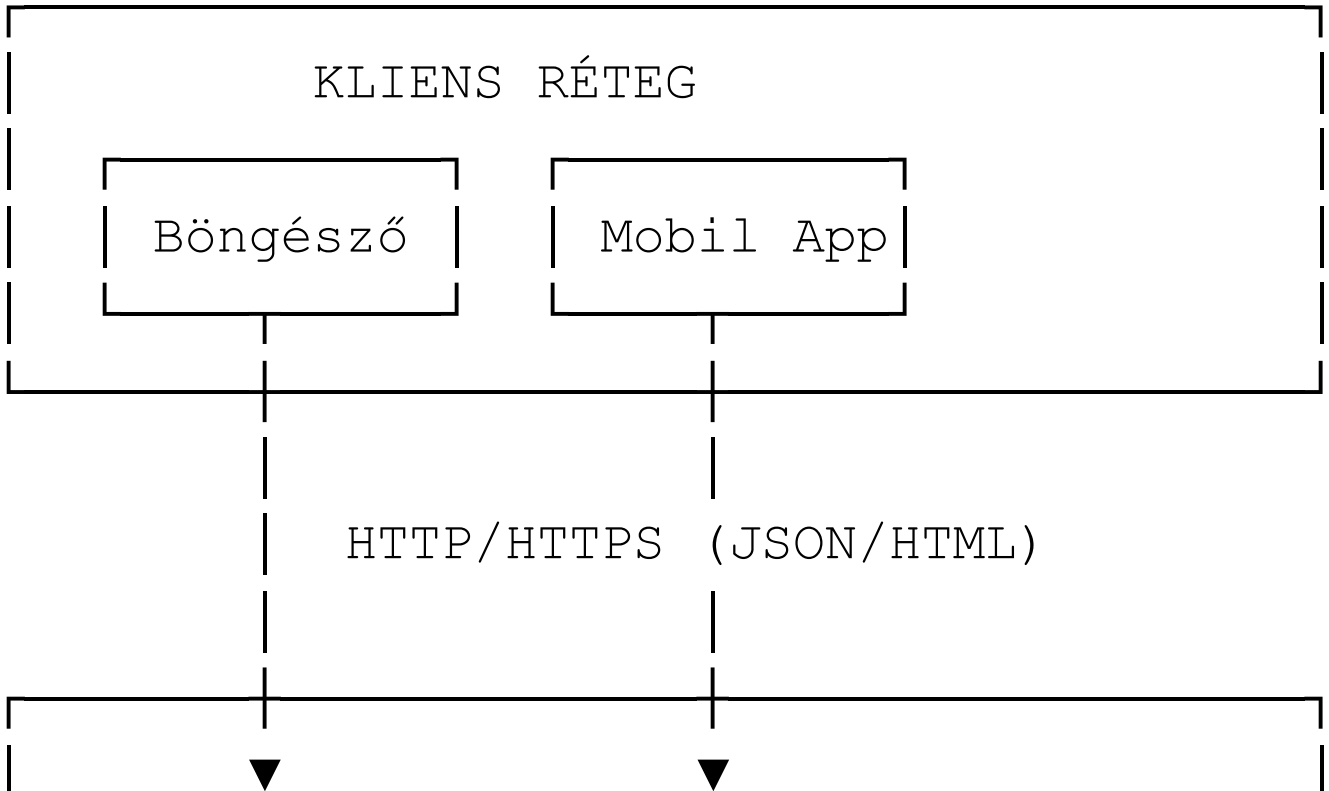
```
id: 42, nev: Kiss Anna
```

4. Szerver válasz (JSON)

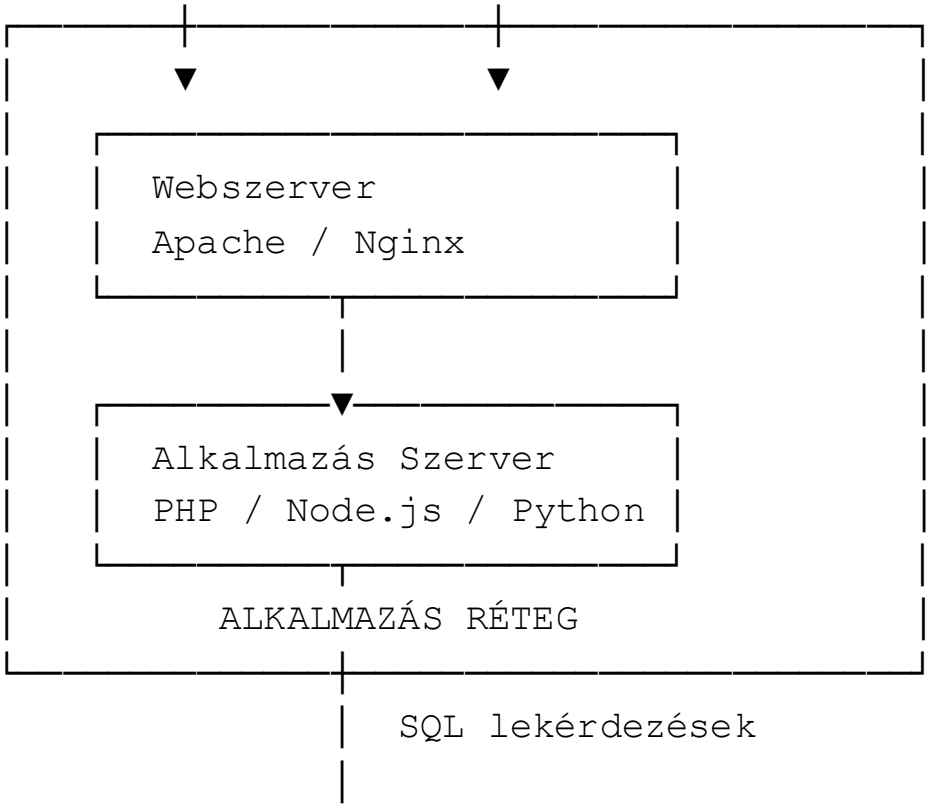
```json
{
    "success": true,
    "user": {"id": 42, "name": "Kiss Anna"},
    "token": "eyJhbGciOiJIUzI1..."
}
```

TELJES ARCHITEKTÚRA DIAGRAM

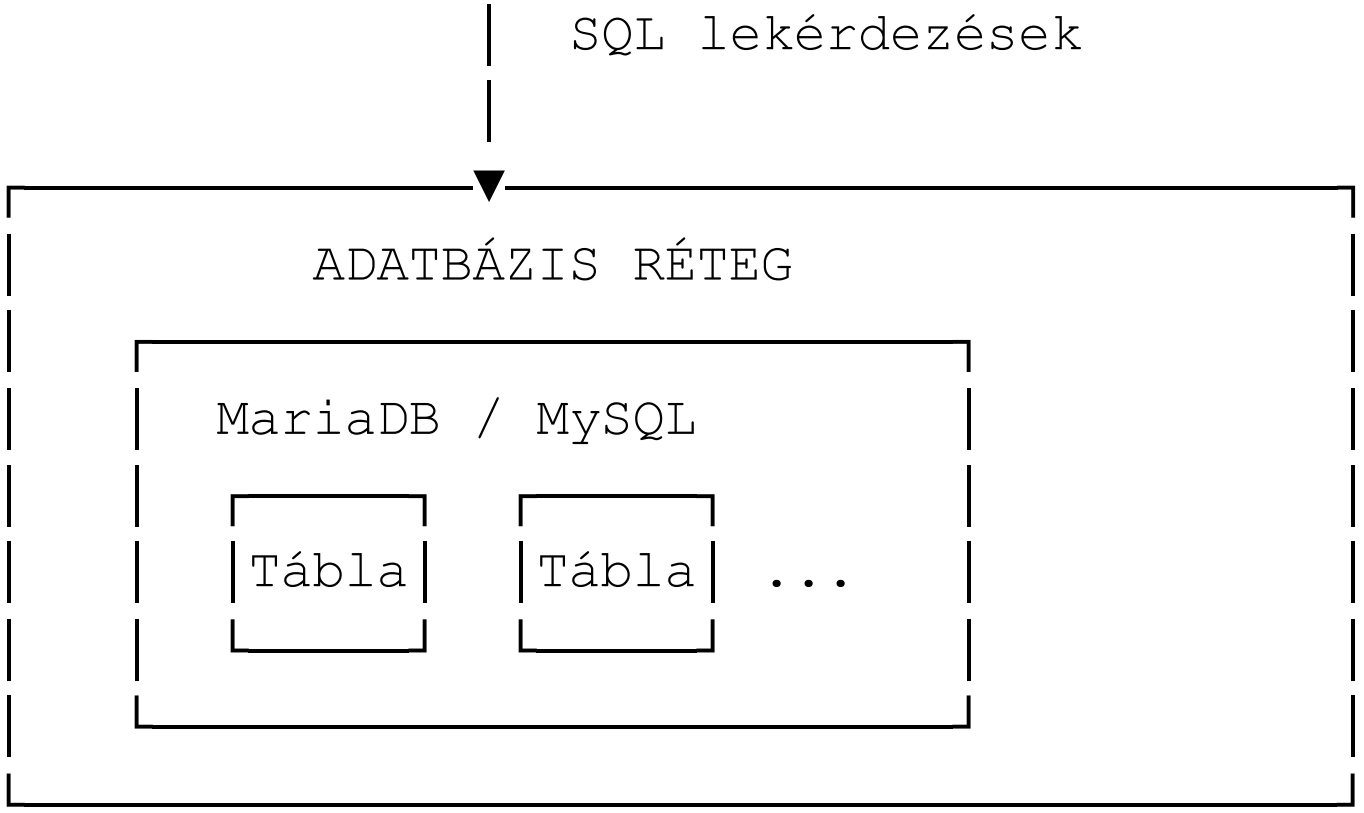
Komponensek és kapcsolataik 1/3



Komponensek és kapcsolataik 2/3



Komponensek és kapcsolataik 3/3



KONKRÉT PÉLDA: WEBSHOP

Architektúra komponensei

- **Frontend (Kliens)** - HTML, CSS, JavaScript - React / Vue.js (opcionális) - Böngészőben fut
- **Backend (Szerver)** - **Webszerver**: Nginx vagy Apache - **Alkalmazás**: PHP 8.x - **Port**: 80 (HTTP) vagy 443 (HTTPS)
- **Adatbázis** - **DBMS**: MariaDB 10.x - **Port**: 3306 - **Táblák**: termekek, rendelesek, felhasznalok

WEBSHOP PÉLDA - REQUEST FOLYAMAT

1. Termék keresés

Felhasználó: "laptop" → Keresés gomb

↓

Böngésző: GET /api/kereses?q=laptop

↓

Nginx: Kérés továbbítása PHP-nak

↓

PHP: SELECT * FROM termekek WHERE nev LIKE '%laptop%'

↓

MariaDB: [Laptop 1, Laptop 2, Laptop 3]

↓

PHP: JSON generálás

↓

Nginx: JSON küldése

↓

Böngésző: Megjelenítés

2. Kosárba helyezés

Felhasználó: "Kosárba" gomb kattintás

↓

JavaScript: POST /api/kosar {"termek_id": 5, "mennyiseg": 1}

↓

PHP: INSERT INTO kosar (felhasznalo_id, termek_id, mennyiseg) VALUES (...)

↓

MariaDB: OK, rekord beszúrva

↓

PHP: {"success": true, "message": "Kosárba helyezve"}

↓

BIZTONSÁG WEBES KÖRNYEZETBEN

SQL Injection védelem

❌ **Veszélyes kód:**

```
$sql = "SELECT * FROM users WHERE email =
'" . $_POST['email'] . "'";
// Támadás: ' OR '1'='1
```

✅ **Biztonságos kód:**

```
$stmt = $conn->prepare("SELECT * FROM
users WHERE email = ?");
$stmt->bind_param("s", $_POST['email']);
$stmt->execute();
// Prepared statements használata!
```

További biztonsági lépések

- 🔒 HTTPS használata (SSL/TLS)
- 🔑 Jelszavak hash-elése (bcrypt, SHA-256)
- 🚫 XSS védelem (input szűrés)
- 🛡️ CSRF tokenek
- ⏱️ Rate limiting (túl sok kérés blokkolása)

TELJESÍTMÉNY OPTIMALIZÁLÁS

Adatbázis szinten

```sql
-- Index létrehozása gyakori keresésekhez
CREATE INDEX idx_termek_nev ON
termekek(nev);

-- Lekérdezés optimalizálás
SELECT id, nev, ar FROM termekek  -- csak
szükséges oszlopok
WHERE kategoria = 'Laptop'
LIMIT 20;  -- lapozás
```

Alkalmazás szinten

- **Caching**: Redis, Memcached
- **Connection pooling**: Adatbázis kapcsolatok újrahasznosítása
- **Lazy loading**: Adatok betöltése csak szükség esetén
  *(párja az **Eager loading**: minden adatot betölt kliens oldalra)*

Webszerver szinten

- **Nginx**: Static fájlok kiszolgálása (képek, CSS, JS)
- **Gzip tömörítés**: Kisebb adatforgalom
- **CDN**: Statikus tartalmak gyorsítása

LOAD BALANCING - NAGY FORGALOM KEZELÉSE

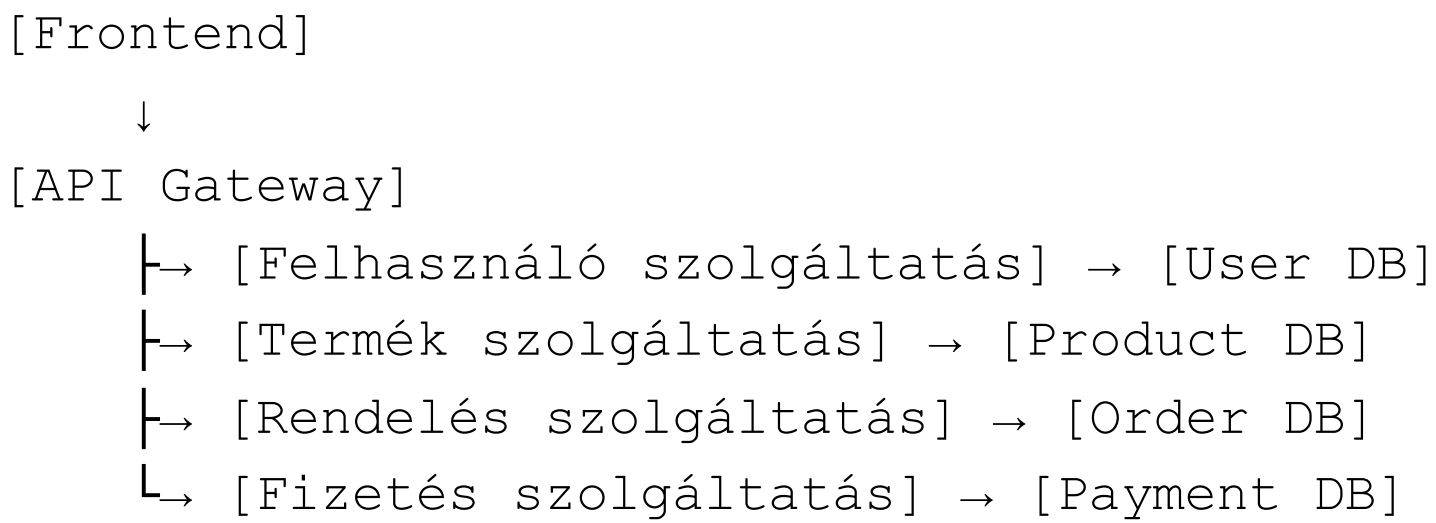
Horizontális skálázás

Miért fontos?

- 🚀 Több ezer egyidejű felhasználó kiszolgálása
- 💪 Ha egy szerver leáll, mások tovább működnek
- 📈 Forgalom növekedéssel könnyű bővíteni

MIKROSZERVIZ ARCHITEKTÚRA (MODERN MEGKÖZELÍTÉS)

Szolgáltatások szétbontása



Előnyök

- ✅ Független fejlesztés és deployment
- ✅ Különböző technológiák használata
- ✅ Jobb skálázhatóság
- ✅ Hibák izolálása

Példák

- Netflix, Amazon, Uber - mind mikroszerviz alapú

ADATBÁZISOK A VALÓ VILÁGBAN

Közösségi média

- **Facebook**: milliárd felhasználó adatai
- **Instagram**: képek, kommentek, lájkok
- **TikTok**: videók, követések, interakciók

E-kereskedelem

- **Amazon, eMag**: termékek, rendelések, készlet
- Felhasználói profilok, kívánságlisták
- Értékelések és vélemények

Egészségügy

- Betegadatok, kezelések története
- Orvosi leletek, receptek
- Sürgősségi információk

ADATBÁZISOK A MINDENNAPOKBAN

Iskolai rendszerek

- Elektronikus napló (Kréta, e-Napló)
- Diákok, tanárok, jegyek, hiányzások
- Órarend, tantárgyak

Közlekedés

- Jegyrendszerek (BKK, MÁV)
- Navigációs alkalmazások
- Járatinformációk

Szórakozás

- Netflix, Spotify: ajánlórendszerek
- Játékok: játékos profilok, eredmények
- YouTube: videók, feliratkozások

ADATVÉDELEM ÉS GDPR

Miért fontos?

- Személyes adatok védelme
- Magánélet tiszteletben tartása
- Jogi megfelelés

GDPR alapelvek

- **Adattakarékosság** - csak a szükséges adatok
- **Célhoz kötöttség** - konkrét célra
- **Transzparencia** - tudd, mit tárolnak
- **Törléshez való jog** - kérheted az adataid törlését
- **Hozzáférés joga** - megnézheted az adataid

KARRIERLEHETŐSÉGEK

Adatbázis-adminisztrátor (DBA)

- Adatbázisok telepítése, karbantartása
- Teljesítmény optimalizálás
- Biztonsági mentések kezelése

Adatbázis-fejlesztő

- Adatbázis tervezése
- SQL lekérdezések írása
- Alkalmazások adatbázis-kapcsolatának fejlesztése

Adatelemző / Data Scientist

- Nagy adathalmazok elemzése
- Minták felismerése
- Üzleti döntések támogatása

HASZNOS ESZKÖZÖK TANULÁSHOZ

Online gyakorlóprogramok

- **W3Schools SQL Tutorial** - ingyenes gyakorlatok
- **SQLZoo** - interaktív feladatok

Ingyenes adatbázisok

- **MySQL** - letölthető, könnyen telepíthető
- **MariaDB** -
- **PostgreSQL** - professzionális funkciók
- **SQLite** - programozás nélkül is használható

Vizuális eszközök

- **phpMyAdmin** - webes felület MySQL-hez
- **DBeaver** - univerzális adatbázis-kezelő
- **MySQL Workbench** - tervezés és adminisztráció

GYAKORLATI TIPPEK

Hogyan kezdj neki?

1. 📚 Tanuld meg az SQL alapokat
2. 💻 Telepíts egy ingyenes DBMS-t
3. 🎯 Hozz létre saját projekteket (pl. könyvtár, filmgyűjtemény)
4. 🔍 Gyakorold a lekérdezéseket
5. 🌐 Csatlakozz online közösségekhez

Jó gyakorlatok

- ✍️ Használj beszédes neveket
- 📝 Kommenteld a bonyolult lekérdezéseket
- 💾 Rendszeresen mentsd az adatokat
- 🔒 Soha ne tárolj jelszavakat titkosítás nélkül

ÖSSZEFOGLALÁS

Mit tanultunk?

- ✅ Mi az adatbázis és miért fontos
- ✅ Relációs adatbázisok felépítése
- ✅ SQL alapműveletek (CREATE, INSERT, SELECT, UPDATE, DELETE)
- ✅ Táblák közötti kapcsolatok
- ✅ Adatbiztonság és integritás
- ✅ Valós alkalmazások
- ✅ Karrierlehetőségek

Következő lépések

- Gyakorold az SQL-t online platformokon
- Tervezz saját adatbázist egy kedvenc témádhoz
- Fedezd fel a fejlettebb adatbázis-koncepciókat

KÖSZÖNÖM A FIGYELMET!

Kérdések?

Hasznos linkek:

- w3schools.com/sql
- sqlzoo.net
- mysql.com
- postgresql.org

Jó tanulást és sok sikert az adatbázisok világában! 🚀