Az adatbázis tervezése: Lépések és alapelvek

A relációs adatbázis-tervezés egy sor egymást követő lépésből áll, amelyek a projekt céljainak megfelelően építkeznek fel:

1. Követelmények meghatározása

A projekt első fontos lépése az adatbázis céljainak és felhasználási területeinek meghatározása. A példában egy egyetemi hallgatói rendszer kialakítására kerül sor, melynek célja a hallgatók, kurzusok és az ezekhez tartozó tanulmányi eredmények nyilvántartása.

2. Adatmodellezés (ER-diagram)

A következő fázisban az entitások, attribútumok és azok kapcsolatai kerülnek meghatározásra. Az ER-diagram segítségével vizuálisan ábrázoljuk a diákok és kurzusok közötti több-a-többhöz kapcsolatot, amelyet a beiratkozás tábla foglal magában.

3. Logikai modell kialakítása

Az ER-diagram alapján elkészítjük a logikai modellt, amelynek során a táblák normalizálása következik. A normalizálás célja, hogy a táblák redundancia-mentesek és könnyen karbantarthatóak legyenek, miközben biztosítjuk az adatkonzisztenciát.

4. Fizikai modell megvalósítása

A fizikai modell a tényleges adatbázis megvalósítását jelenti. Itt választjuk ki a megfelelő adatbázis-kezelő rendszert (pl. MySQL, PostgreSQL) és definiáljuk a táblák struktúráját, kulcsokat, indexeket, valamint egyéb optimalizálási lehetőségeket.

A relációs adatbázis felépítése és adattípusok

A relációs adatbázisok táblákban tárolják az adatokat, ahol minden tábla sorokból és oszlopokból áll. Az adattípusok pontos meghatározása alapvető fontosságú a megfelelő adattárolás és a hatékony lekérdezés biztosítása érdekében.

Alapvető adattípusok

A leggyakrabban használt adattípusok a következők:

- **INTEGER**: Egész számok tárolására
- **VARCHAR**: Változó hosszúságú karakterláncok (szövegek) tárolása
- **DATE**: Dátumok tárolása
- **FLOAT/DOUBLE**: Lebegőpontos számok
- **BOOLEAN**: Logikai értékek (igaz/hamis)

Például, a diákok azonosítóit INTEGER típusú mezőben tárolhatjuk, míg nevüket VARCHAR típusú oszlopokban.

Az Egyetemi Hallgatói Rendszer adatbázisa

A dolgozatban egy egyszerű egyetemi hallgatói rendszer adatbázisát tervezzük, amely három alapvető táblát tartalmaz:

1. Diák tábla (Student)

- **student_id** (INTEGER, elsődleges kulcs)
- **first_name** (VARCHAR)
- **last_name** (VARCHAR)
- **birth_date** (DATE)
- **email** (VARCHAR)

2. Kurzus tábla (Course)

- **course_id** (INTEGER, elsődleges kulcs)
- **course_name** (VARCHAR)
- **credits** (INTEGER)

3. Beiratkozás tábla (Enrollment)

- **enrollment_id** (INTEGER, elsődleges kulcs)
- **student_id** (INTEGER, idegen kulcs, hivatkozik a Student táblára)
- **course_id** (INTEGER, idegen kulcs, hivatkozik a Course táblára)
- **enrollment_date** (DATE)

Ez a felépítés biztosítja, hogy a hallgatók és a kurzusok között egy több-a-többhöz kapcsolat alakuljon ki.

SQL Példák és Adatkezelési Műveletek

Az SQL segítségével az adatbázisban végzett műveletek egyszerűen végrehajthatók:

Táblák létrehozása

sql
Copy

```
CREATE TABLE Student (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    birth_date DATE,
    email VARCHAR(100)
);

CREATE TABLE Course (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(100),
    credits INT
```

```
);

CREATE TABLE Enrollment (
    enrollment_id INT PRIMARY KEY,
    student_id INT,
    course_id INT,
    enrollment_date DATE,
    FOREIGN KEY (student_id) REFERENCES Student(student_id),
    FOREIGN KEY (course_id) REFERENCES Course(course_id)
);
```

Adatbevitel (INSERT)

```
sql
Copy
INSERT INTO Student (student_id, first_name, last_name, birth_date, email)
VALUES (1, 'Kovács', 'Anna', '2000-05-15', 'anna.kovacs@example.com');
```

Adatlekérdezés (SELECT)

```
sql
Copy
SELECT S.first_name, S.last_name, C.course_name, E.enrollment_date
FROM Student S
JOIN Enrollment E ON S.student_id = E.student_id
JOIN Course C ON E.course_id = C.course_id;
```

Adat módosítása (UPDATE)

```
sql
Copy
UPDATE Student
SET email = 'uj.email@example.com'
WHERE student_id = 1;
```

Adat törlése (DELETE)

```
sql
Copy
DELETE FROM Enrollment
WHERE enrollment_id = 1001;
```

Adatkezelési Műveletek és Tranzakciókezelés

A CRUD (Create, Read, Update, Delete) műveletek az alapvető adatkezelési műveletek az adatbázisban. Tranzakciók segítségével biztosítható az adatbázis integritása és az adatkonzisztencia. A tranzakciók biztosítják, hogy vagy minden művelet sikeresen végbemehet, vagy egy hiba esetén az adatbázis visszaállítható az előző állapotba.

Tranzakció Példa:

```
sql
Copy
BEGIN TRANSACTION;

INSERT INTO Student (student_id, first_name, last_name, birth_date, email)
VALUES (2, 'Nagy', 'Béla', '1999-03-22', 'bela.nagy@example.com');
```

```
INSERT INTO Enrollment (enrollment_id, student_id, course_id, enrollment_date)
VALUES (1002, 2, 101, '2025-02-05');

COMMIT;
```

Összegzés

A dolgozat során részletesen bemutattuk, hogyan tervezhetünk és valósíthatunk meg egy relációs adatbázist, figyelembe véve az adatkezelés alapvető lépéseit. Az ER-diagram, adattípusok és SQL parancsok alkalmazásával olyan adatbázisokat hozhatunk létre, amelyek hatékonyan kezelhetik az adatokat. A tranzakciók és adatkezelési műveletek biztosítják a rendszer integritását és megbízhatóságát.