Dunaújvárosi Egyetem Bánki Donát Technikum
Projekt feladat dokumentáció

Projekt tervező: Radnai Szabolcs
Projekt címe: Adatbáziskezelés
Osztály: 12.C/ IpInf
Dátum: 2023.11.21

1. Bevezetés

A relációs adatbázis-kezelő rendszerek (RDBMS) a modern informatikai megoldások alapját képezik, mivel strukturált adatokat tárolnak, kezelnek és lekérdeznek. Jelen dolgozatban egy egyszerű példán keresztül mutatjuk be az adatbázis-tervezést, az adattípusok meghatározását, valamint az alapvető adatkezelési műveletek (CRUD – Create, Read, Update, Delete) alkalmazását.

2. Az adatbázis tervezésének lépései

Az adatbázis kialakítása több lépésből áll, melyek szorosan összefonódnak:

2.1. Követelmények felmérése

Minden projekt első lépése az adatbázis céljának és felhasználási területeinek meghatározása. Példánkban egy egyetemi hallgatói rendszert valósítunk meg, amely a diákok, kurzusok és tanulmányi eredmények nyilvántartására szolgál.

2.2. Adatmodellezés (ER-diagram)

A következő lépés az entitások, attribútumok és azok kapcsolatai azonosítása. Példánkban három fő entitást azonosítunk:

- Diák (Student)
- Kurzus (Course)
- Beiratkozás (Enrollment)

Az ER-diagram vizuálisan ábrázolja, hogy egy diák több kurzusra beiratkozhat, míg egy kurzus több diákot is befogadhat.

2.3. Logikai modell kialakítása

Az ER-diagram alapján a logikai modellt normalizáljuk, azaz redundancia-mentes táblákat hozunk létre. E lépések során biztosítjuk, hogy az adatok konzisztensen legyenek tárolva.

2.4. Fizikai modell létrehozása

A fizikai modell az adatbázis tényleges megvalósítását jelenti. E szakaszban kiválasztjuk az adatbázis-kezelő rendszert (pl. MySQL, PostgreSQL vagy SQLite), és definiáljuk a táblák struktúráját, indexeket, kulcsokat és egyéb optimalizálási elemeket.

3. Relációs adatbázis és adattípusok

A relációs adatbázis adatok táblákban való tárolását jelenti, ahol minden tábla sorokból (rekordokból) és oszlopokból (attribútumokból) áll.

3.1. Alapvető adattípusok

Az adatbázisokban gyakori adattípusok a következők:

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

Például egy diák azonosítójának tárolására az INTEGER, míg a nevét VARCHAR(100) típusú oszlopban definiálhatjuk.

4. Az adatbázis felépítése – Példa: Egyetemi Hallgatói Rendszer

4.1. Táblák és kapcsolatok

A dolgozatunkban három fő táblát hozunk létre:

4.1.1. Diák (Student) tábla

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

4.1.2. Kurzus (Course) tábla

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

4.1.3. Beiratkozás (Enrollment) tábla

- **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 struktúra biztosítja a diákok és kurzusok közötti több-a-többhöz (many-to-many) kapcsolatot a beiratkozás táblán keresztül.

4.2. SQL kód példák

Az alábbi SQL parancsok szemléltetik a táblák létrehozását és az alapvető adatkezelési műveleteket.

4.2.1. 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)
);
```

4.2.2. Adatbevitel (INSERT)

```
sql
Copy
INSERT INTO Student (student_id, first_name, last_name, birth_date, email)
VALUES (1, 'Szabo', 'Bela', '1978-10-4', 'belaszabo@example.com');

INSERT INTO Course (course_id, course_name, credits)
VALUES (101, 'Adatbázisok alapjai', 3);

INSERT INTO Enrollment (enrollment_id, student_id, course_id,
enrollment_date)
```

```
VALUES (1001, 1, 101, '2025-02-01');
```

4.2.3. 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;
```

4.2.4. Adat módosítás (UPDATE)

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

4.2.5. Adat törlés (DELETE)

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

5. Adatkezelési műveletek és tranzakciókezelés

Az adatkezelési műveletek négy alapvető funkciót foglalnak magukban:

- **Létrehozás (CREATE)**: Új adatok beillesztése.
- **Lekérdezés (READ)**: Adatok kivonása és megjelenítése.
- **Módosítás (UPDATE)**: Meglévő adatok frissítése.
- **Törlés (DELETE)**: Adatok eltávolítása.

5.1. Tranzakciók

A tranzakciók segítenek megőrizni az adatbázis integritását. Egy tranzakció során több művelet hajtható végre, és az „ACID” (Atomicity, Consistency, Isolation, Durability) elvek garantálják, hogy vagy minden művelet sikeres, vagy az adatbázis visszatér az eredeti állapotába.

sql
Copy
```
BEGIN TRANSACTION;

INSERT INTO Student (student_id, first_name, last_name, birth_date, email)
VALUES (2, 'Toth', 'Gabor', '1999-03-22', gabor.toth@example.com');

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

COMMIT;
```

Ha a tranzakció egy része sikertelen, a teljes tranzakció visszagörgethető:

sql
Copy
```
BEGIN TRANSACTION;
-- Hiba esetén:
ROLLBACK;
```

6. Összegzés

A dolgozat során részletesen bemutattuk a relációs adatbázis kialakításának lépéseit a követelmények felmérésétől a fizikai modell létrehozásáig. Megismertük az alapvető adattípusokat és az SQL segítségével végrehajtható adatkezelési műveleteket. A példán keresztül láthattuk, hogyan tervezzük és valósítjuk meg egy egyetemi hallgatói rendszer adatbázisát, demonstrálva az elméleti tudás gyakorlati alkalmazását.

Ez a minta dolgozat hasznos alapot nyújthat egy iskolai feladat elkészítéséhez, ahol az adatbázis-tervezés és -kezelés alapelveit kell részletesen bemutatni.

7. Mellékletek (opcionális)

7.1. Diagramok

Az ER-diagram és a relációs modell vizuális ábrázolása segíti az összefüggések jobb megértését, például, hogy:

- Egy diák több kurzusra beiratkozhat.
- Egy kurzus több diákot fogadhat.
- A beiratkozás tábla biztosítja a diák és kurzus közötti kapcsolatot.

7.2. További adatbázis-kezelési technikák

A dolgozat kiterjeszthető a következő témák ismertetésével:

- Indexelés és teljesítményoptimalizálás.
- Biztonsági mentések (backup) és visszaállítási (recovery) eljárások.
- Jogosultságkezelés és felhasználói hozzáférések szabályozása.