OSZTALYNAPLO PROJEKT

176-os backend csoport Záróviszga előkpészítő projektje

– teljes anyag, tesztesetekkel -

–

Specifikáció a Konzolos Adatbevitelhez

1. Általános Cél

A konzolos alkalmazás célja, hogy diákok adatait (név, születési év, osztály) rögzítse egy fájlba, majd ezt a fájlt felhasználja a Spring-alapú rendszerben. A diákok adatait egy `diaknevsor.txt` nevű fájlban kell tárolni.

2. Funkciók és Feladatok

1. **Adatbevitel**:
    - A felhasználó konzolon keresztül viszi be a diákok adatait.
    - Az adatok a következők:
        - Név (String)
        - Születési év (int)
        - Osztály (String)
    - Az adatok rögzítéséhez a konzolos alkalmazás használja a `DiakNevsor` JavaBean-t.
2. **Fájlkezelés**:
    - Az alkalmazás rögzíti a diákok adatait egy `diaknevsor.txt` nevű fájlba.
    - Az adatok fájlba mentése során minden diák adatai egy sorban kerülnek tárolásra, vesszővel elválasztva.
3. **Adatok Listázása**:
    - A konzolos alkalmazás biztosít egy lehetőséget a felhasználó számára, hogy listázza a `diaknevsor.txt` fájlban tárolt adatokat.
    - Az adatok kiírása során minden diák adatai külön sorban jelennek meg.
4. **Átlagéletkor Számítása**:
    - A konzolos alkalmazás kiszámolja és megjeleníti a felvitt diákok átlagéletkorát.
5. **Medián Életkor Számítása**:
    - A konzolos alkalmazás kiszámolja és megjeleníti a felvitt diákok medián életkorát.

3. Felhasználói Interakciók

- **Diák felvitele**:

    - A felhasználó megadja a diák nevét, születési évét és osztályát.
    - A megadott adatokat az alkalmazás rögzíti a fájlban.
- **Diákok listázása**:
    - A felhasználó kérésére a konzolos alkalmazás kilistázza a `diaknevsor.txt` fájlban tárolt diákokat.
- **Átlagéletkor és Medián Életkor**:
    - A felhasználó kérésére az alkalmazás kiszámítja és megjeleníti az átlagéletkort és a medián életkort.

Specifikáció a Spring Alapú Rendszerhez

1. Általános cél

A Spring alapú rendszer célja, hogy az osztálynaplót kezelje, jegyeket rögzítsen, valamint számításokat végezzen az osztály- és iskolai átlagok alapján.

2. Fő Package struktúra

- A projekt fő package útvonala: `hu.<neved>.backend.osztalynaplo`.
- A konzolos alkalmazás, amely a `DiakNevsor.java`-t tartalmazza, a `diaknevsor` mappában található. Ez az alkalmazás nincs közvetlenül integrálva a Spring projekttel, de ugyanazon projekt részeként van elhelyezve.

3. Funkciók és Feladatok

1. **Diákok importálása**:
    - A `public void importDiakok(List<Diakfelvetel> diakfelvetelek)` metódus felelős a diákok adatainak importálásáért a `Diakfelvetel` objektumokból.
    - Az importálás során a diákokat a rendszer JPA entitásokként kezeli, és elmenti az adatbázisba.
2. **Jegyek rögzítése**:
    - A `public Jegy letrehozOsztalyzatot(JegyDto jegyDto)` metódus felelős egy új osztályzat létrehozásáért.
    - A metódus először ellenőrzi, hogy a diák létezik-e a rendszerben a `public Boolean isDiakLetezik(Long diakId)` metódus segítségével.
    - Ha a diák létezik, a jegy rögzítésre kerül; ha nem, hibaüzenetet ad vissza.
3. **Osztálynapló lekérdezése**:
    - A `public Map<String, List<DiakOsztalyzat>> getOsztalynaplo(String osztaly)` metódus visszaadja az adott osztályhoz tartozó naplót.
    - A naplóban minden tantárgyhoz (`K` - kulcs) tartozik egy lista (`V` - érték), amely a

diákok nevét és az adott tantárgyhoz tartozó osztályzataikat tartalmazza.
- Az eredmény JSON formátumban kerül visszaadásra.

4. **Átlagszámítás**:
    - Az `atlagolas()` metódus egy külön eljárás, amely az osztály és iskolai átlag számításáért felelős.
    - A metódus kétféleképpen kerül felhasználásra:
        - **Osztályátlag**: Az adott osztály összes diákjának összes jegyét veszi figyelembe.
        - **Iskolai átlag**: Az egész iskola összes diákjának összes jegyét veszi figyelembe.
5. **Kivételkezelés**:
    - Amikor osztályzatot adnak egy diákhoz, a rendszer ellenőrzi, hogy a diák létezik-e a `isDiakLetezik(Long diakId)` metódus segítségével.
    - Ha a diák nem található, hibaüzenetet ad vissza.

4. Kritériumok és korlátok

- A fejlesztés során **csak** az ebben a specifikációban szereplő követelményekkel kell foglalkozni.
- A `diaknevsor.txt` fájl és a `DiakNevsor.java` alkalmazás nem részei a Spring alapú rendszernek, csak segédeszközként szolgálnak ahhoz, pontosabban az adatok előállításához (az import végpontnál található funkcióval).
- Az osztályátlag és iskolai átlag számítása során csak a jelenlegi diákok és jegyeik kerülnek figyelembevételre.

5. Végpontok

- **Diákok importálása**: POST `/diakok/import`
- **Jegy létrehozása**: POST `/osztalyzat`
- **Osztálynapló lekérdezése**: GET `/diakok/osztalynaplo/{osztaly}`
- **Osztályátlag lekérdezése**: GET `/diakok/osztalyatlag/{osztaly}`
- **Iskolai átlag lekérdezése**: GET `/diakok/iskolaiatlag`

Megjegyzések

- Minden metódus és funkció összhangban áll a Spring keretrendszer követelményeivel és a JPA használatával.
- A specifikációban nem említett funkciók és metódusok megvalósítása **nem** szükséges, és a fejlesztés során azokra ne fordítsatok rá különösebb figyelmet.

Tesztesetek specifikációja:

1.1. `importDiakok(List<Diakfelvetel> diakfelvetelek)` Metódus Tesztje

- **Cél**: Ellenőrizni, hogy a `importDiakok` metódus helyesen importálja a diákokat a `Diakfelvetel` objektumokból, és helyesen menti el őket az adatbázisban.
- **Tesztelési esetek**:
    - Egyetlen diák importálása.
    - Több diák importálása.
    - Üres lista importálása (ellenőrizni kell, hogy nem történik-e adatbázis művelet).

1.2. `letrehozOsztalyzatot(JegyDto jegyDto)` Metódus Tesztje

- **Cél**: Ellenőrizni, hogy a `letrehozOsztalyzatot` metódus helyesen hoz létre egy új osztályzatot, és azt megfelelően menti el az adatbázisban.
- **Tesztelési esetek**:
    - Egy osztályzat létrehozása létező diák számára.
    - Egy osztályzat létrehozása nem létező diák számára (ellenőrizni kell, hogy hibaüzenetet ad-e vissza).
    - Különböző tantárgyakhoz tartozó osztályzatok létrehozása.

1.3. `isDiakLetezik(Long diakId)` Metódus Tesztje

- **Cél**: Ellenőrizni, hogy a `isDiakLetezik` metódus helyesen ellenőrzi, hogy egy adott diák létezik-e az adatbázisban.
- **Tesztelési esetek**:
    - Létező diák ellenőrzése (igaz érték visszaadása).
    - Nem létező diák ellenőrzése (hamis érték visszaadása).

1.4. `getOsztalynaplo(String osztaly)` Metódus Tesztje

- **Cél**: Ellenőrizni, hogy a `getOsztalynaplo` metódus helyesen adja vissza az adott osztályhoz tartozó osztálynaplót.
- **Tesztelési esetek**:
    - Egyetlen tantárgyhoz tartozó jegyek lekérdezése.
    - Több tantárgyhoz tartozó jegyek lekérdezése.
    - Üres osztály naplójának lekérdezése (üres eredmény visszaadása).

1.5. `getOsztalyAtlag(String osztaly)` Metódus Tesztje

- **Cél**: Ellenőrizni, hogy a `getOsztalyAtlag` metódus helyesen számolja ki az adott osztály átlagát.
- **Tesztelési esetek**:
    - Átlag számítása, ha minden diáknak van jegye.
    - Átlag számítása, ha egyes diákoknak nincs jegye.
    - Átlag számítása üres osztály esetén (0 visszaadása).

1.6. `getIskolaiAtlag()` Metódus Tesztje

- **Cél**: Ellenőrizni, hogy a `getIskolaiAtlag` metódus helyesen számolja ki az iskolai átlagot.
- **Tesztelési esetek**:
    - Átlag számítása, ha minden diáknak van jegye.
    - Átlag számítása, ha egyes diákoknak nincs jegye.
    - Átlag számítása üres adatbázis esetén (0 visszaadása).

SQL natív kérdések és lekérdezések:
A fenti projekt külső adatbáziskezelőben való használatának natív sql adatkezelési szükségletei:

2.1. Adatbázis Struktúra Létrehozása

```
CREATE DATABASE IF NOT EXISTS osztalynaplo;
USE osztalynaplo;

CREATE TABLE Diak (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    nev VARCHAR(255) NOT NULL,
    szuletesi_eve INT NOT NULL,
    osztaly VARCHAR(50) NOT NULL
);

CREATE TABLE Jegy (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    diak_id BIGINT NOT NULL,
    tantargy VARCHAR(255) NOT NULL,
    jegy INT NOT NULL,
    FOREIGN KEY (diak_id) REFERENCES Diak(id) ON DELETE CASCADE
);
```

2.2. Adatok Felvitele

```
-- Diakok felvitele
INSERT INTO Diak (nev, szuletesi_eve, osztaly) VALUES ('Kiss Anna', 2010,
'6.A');
INSERT INTO Diak (nev, szuletesi_eve, osztaly) VALUES ('Nagy Péter', 2009,
'7.B');
INSERT INTO Diak (nev, szuletesi_eve, osztaly) VALUES ('Kovács Bence', 2012,
'5.A');

-- Jegyek felvitele
INSERT INTO Jegy (diak_id, tantargy, jegy) VALUES (1, 'Matematika', 5);
INSERT INTO Jegy (diak_id, tantargy, jegy) VALUES (1, 'Történelem', 4);
INSERT INTO Jegy (diak_id, tantargy, jegy) VALUES (2, 'Matematika', 3);
INSERT INTO Jegy (diak_id, tantargy, jegy) VALUES (3, 'Matematika', 4);
INSERT INTO Jegy (diak_id, tantargy, jegy) VALUES (3, 'Történelem', 5);
```

2.3. Alapvető Lekérdezések

```
-- Lekérdezés az összes diák adatainak megjelenítésére
SELECT * FROM Diak;

-- Lekérdezés az összes jegy megjelenítésére, tantárgy szerint csoportosítva
SELECT tantargy, nev, jegy
FROM Jegy
JOIN Diak ON Jegy.diak_id = Diak.id
ORDER BY tantargy;

-- Az adott osztály összes diákjának és jegyének megjelenítése
SELECT osztaly, nev, tantargy, jegy
FROM Jegy
JOIN Diak ON Jegy.diak_id = Diak.id
WHERE osztaly = '6.A';
```

2.4. Szelektív Frissítés (UPDATE)

```
-- Egy diák osztályzatának frissítése egy adott tantárgyból
UPDATE Jegy
SET jegy = 5
WHERE diak_id = 2
AND tantargy = 'Matematika';
```

2.5. Szelektív Törlés (DELETE)

```
-- Egy diák összes jegyének törlése
DELETE FROM Jegy
WHERE diak_id = 3;

-- Egy konkrét tantárgyból kapott jegy törlése
DELETE FROM Jegy
WHERE diak_id = 1
AND tantargy = 'Történelem';
```

2.6. Ösztönzés a SQL szkript készítésére

Másoljátok ki a fenti SQL parancsokat a MySQL Workbench-ből, és mentsétek el egy `szkript.sql` fájlba.

Ez segít a későbbi újrafelhasználásban, valamint az adatbázis könnyebb létrehozásában és kezelésében.

Összefoglalás

Ezek az SQL parancsok lefedik az adatbázis létrehozását, alapvető adatok felvitelét, különböző lekérdezéseket, valamint szelektív frissítést és törlést. Javaslom, hogy ezeket az SQL parancsokat egy szkriptfájlba mentsétek el, hogy könnyebben tudjátok kezelni az adatbázisokat a MySQL Workbench-ben.