Beadandó feladat

Budapesti Corvinus Egyetem

Adatbázisok tárgy

2023.05.15.

G02 gyakorlat

Beniczky Tamás

Ladomerszky Torda

Veress Mária

**GitHub link:**

<https://github.com/maricaveress/adatbazis_beadando.git>

Tartalomjegyzék

[Választott feladat: 3 3](#_Toc166680024)

[A fejlesztői környezet és kapcsolat létrehozásának bemutatása 3](#_Toc166680025)

[Az adatbázis bemutatása 4](#_Toc166680026)

[Lekérdezések 4](#_Toc166680027)

[Első lekérdezés 4](#_Toc166680028)

[Második lekérdezés 5](#_Toc166680029)

[Harmadik lekérdezés 6](#_Toc166680030)

[Negyedik lekérdezés 6](#_Toc166680031)

[Ötödik lekérdezés 7](#_Toc166680032)

[CRUD műveletek 7](#_Toc166680033)

[READ (Olvasás) 8](#_Toc166680034)

[DELETE (Törlés) 8](#_Toc166680035)

[CREATE (Létrehozás) 8](#_Toc166680036)

[UPDATE 1. (Frissítés) 8](#_Toc166680037)

[UPDATE 2. (Frissítés) 9](#_Toc166680038)

[Ábrázolás, diagrammok 9](#_Toc166680039)

[Különböző férőhelyes szobák száma 9](#_Toc166680040)

[Klímás szobák aránya 10](#_Toc166680041)

# Választott feladat: 3

SQL vagy NoSQL adatbázis létrehozása (min. 4 tábla) importálással vagy parancsokkal, majd csatlakozás valamilyen lehetséges programnyelv (pl. Python) segítségével, és lekérdezések (legalább 5), valamint CRUD-műveletek (legalább 5) megvalósítása programkódból.

A mi csapatunk SQL adatbázist importált az egyetemi weboldalról. A szálláshely adatbázist választottuk, hiszen azt már megfelelően ismerjük. Visual Studio Code, illetve Jupyter Notebook fejlesztői környezetben Python programnyelv segítségével készítettünk lekérdezéseket, CRUD-műveleteket és a diagrammokat.

## A fejlesztői környezet és kapcsolat létrehozásának bemutatása

A python kódot Visual Studio Code és Jupyter Noptebook környezetben készítettük.

Az általunk írt kód a szálláshely adatbázissal történő kommunikációt valósít meg a pyodbc modul segítségével. A pyodbc modul importálása után létrehoztuk a kapcsolatot az adatbázissal a pyodbc.connect() függvény segítségével.

Az egyetemi SQL Server adatbázisára kapcsolódtunk a szükséges hitelesítési adatok megadásával. Ezután létrehoztunk egy cursor objektumot, amely segítségével végre tudjuk hajtani az SQL lekérdezéseket az adatbázison.

import pyodbc

# Kapcsolat létrehozása az adatbázissal

conn = pyodbc.connect(

'DRIVER={SQL Server};'

'SERVER=bit.uni-corvinus.hu;'

'DATABASE=szallashely;'

'UID=Hallgato;'

'PWD=Password123;'

)

# Cursor objektum létrehozása SQL lekérdezésekhez

cursor = conn.cursor()

Minden különálló kódrészlet végén bezártuk a kapcsolatot az adatbázissal.

# Kapcsolat bezárása

conn.close()

## Az adatbázis bemutatása

A projektfeladat elkészítése során használt adatforrásunk a szálláshely adatbázis, amelyet a bit.uni-corvinus.hu szerverről értünk el, az egyetem VPN-éhez csatlakozva importáltunk.

Azért erre a forrásra esett a választásunk, mivel a gyakorlati órák során már kiismertük magunkat az adatbázis felépítését illetően.

Az adatbázis négy táblából áll, melyek tartalmazzák a szálláshelyek, a szobák, a vendégek és a foglalások adatait. A szálláshely tábla idegen kulccsal van összekapcsolva a szoba táblához, ami szintén idegen kulccsal kapcsolódik a foglalás táblához, és végül a vendég tábla is össze van kapcsolva a foglalás táblához az usernev-en keresztül.

A képen szöveg, diagram, Párhuzamos, képernyőkép látható

Automatikusan generált leírás

Mivel importáltuk az adatbázist, nem volt szükség domain megszorítások, triggerek és kényszerek, és elsődleges kulcsok meghatározására.

## Lekérdezések

Az általunk írt alábbi kódban az adatbázis kapcsolatának létrehozásától kezdve az adatok lekérdezéséig és kiírásáig minden lépés megtalálható.

Öt különböző összetett lekérdezést hajtottunk végre az adatbázison, mindegyik más-más kérdést tárgyalva. A lekérdezések végén egyesével kiírattuk az outputot a konzolra, hogy láthassuk a visszatérő adatokat.

### Első lekérdezés

Az első összetett lekérdezés arra ad választ, hogy melyik szálláshelyen található a legtöbb olyan foglalás, amely legalább 5 éjszakára szól, és a foglalás időszakában volt legalább 2 gyermek is. A gyakorlatban hasznos információt nyújthat arról, hogy melyik szálláshely a leginkább vonzó családok számára.

cursor.execute("""

SELECT TOP 1 s.szallas\_nev, COUNT(f.foglalas\_pk) AS foglalas\_szam, SUM(f.felnott\_szam + f.gyermek\_szam) AS osszes\_resztvevo

FROM szallashely s

JOIN szoba sz ON s.szallas\_id = sz.szallas\_fk

JOIN foglalas f ON sz.szoba\_id = f.szoba\_fk

WHERE DATEDIFF(DAY, f.mettol, f.meddig) >= 5 AND f.gyermek\_szam >= 2

GROUP BY s.szallas\_nev

ORDER BY foglalas\_szam DESC

""")

print("Lekérdezés 1.")

for row in cursor.fetchall():

print(row)

Ez a szálláshely jelen esetben a 'Hotel Három Hattyú'.

### Második lekérdezés

Azokat a szálláshelyeket listázza, ahol az összesített vendégszám eléri vagy meghaladja az átlagos foglalások vendégszámának kétszeresét. Ez fontos információ lehet például további üzleti döntések meghozatalához a szálláshelyek számára.

cursor.execute("""

SELECT TOP 1 s.szallas\_nev, COUNT(f.foglalas\_pk) AS foglalas\_szam

FROM szallashely s

JOIN szoba sz ON s.szallas\_id = sz.szallas\_fk

JOIN foglalas f ON sz.szoba\_id = f.szoba\_fk

GROUP BY s.szallas\_nev

HAVING SUM(f.felnott\_szam + f.gyermek\_szam) >= (SELECT 2 \* AVG(f2.felnott\_szam + f2.gyermek\_szam) FROM foglalas f2)

ORDER BY foglalas\_szam DESC

""")

print("Lekérdezés 2.")

for row in cursor.fetchall():

print(row)

A lekérdezés eredményeként szintén a 'Hotel Három Hattyú'-t kaptuk eredményül, amelyik 204 volt a foglalások száma.

### Harmadik lekérdezés

A hármas számú lekérdezésünk azt a szálláshelyet keresi meg, ahol a legtöbb olyan foglalás van, amit 1 gyermek és legalább 2 felnőtt részére tettek.

cursor.execute("""

SELECT TOP 1 s.szallas\_nev, COUNT(f.foglalas\_pk) AS foglalas\_szam

FROM szallashely s

JOIN szoba sz ON s.szallas\_id = sz.szallas\_fk

JOIN foglalas f ON sz.szoba\_id = f.szoba\_fk

WHERE f.felnott\_szam >= 2 AND f.gyermek\_szam >= 1

GROUP BY s.szallas\_nev

ORDER BY foglalas\_szam DESC

""")

print("Lekérdezés 3.")

for row in cursor.fetchall():

print(row)

Eredményül a 'Hotel Három Hattyú' szálláshelyet kaptuk, ahol 40 darab ilyen foglalás történt.

### Negyedik lekérdezés

Azokat a szálláshelyeket mutatja meg, ahol a legtöbb pótágyas szoba található. Ez a lekérdezés segíthet felmérni az egyes szálláshelyek kényelmi szolgáltatásait és akár a jövőbeli bővítési lehetőségeket.

cursor.execute("""

SELECT TOP 1 s.szallas\_nev, COUNT(\*) AS potalyas\_szoba\_szam

FROM szallashely s

JOIN szoba sz ON s.szallas\_id = sz.szallas\_fk

WHERE sz.potagy > 0

GROUP BY s.szallas\_nev

ORDER BY potalyas\_szoba\_szam DESC

""")

print("Lekérdezés 4.")

for row in cursor.fetchall():

print(row)

A 'Gold Hotel' az a szálláshely, amelyik jelenleg tartja a rekordot, pontosan 27 darab pótágyas szobával.

### Ötödik lekérdezés

Az utolsó összetett lekérdezésünk listázza azokat a szálláshelyeket, ahol legalább egy olyan szoba van, amelyben légkondicionáló van és legalább 2 férőhely áll rendelkezésre a foglalásra.

cursor.execute("""

SELECT s.szallas\_nev

FROM szallashely s

JOIN szoba sz ON s.szallas\_id = sz.szallas\_fk

JOIN foglalas f ON sz.szoba\_id = f.szoba\_fk

WHERE sz.klimas = 'i' AND f.felnott\_szam + f.gyermek\_szam >= 2

GROUP BY s.szallas\_nev;

""")

print("Lekérdezés 5.")

for row in cursor.fetchall():

print(row)

Kiíratáskor láthatjuk, hogy pontosan 11 darab ilyen szálláshely található az adatbázisban.

## CRUD műveletek

Az alábbi kódok különböző CRUD (Create, Read, Update, Delete) műveleteket hajtanak végre egy SQL adatbázison. A kódok sorrendben kezelik a létrehozást, olvasást, frissítést és törlést, és mindegyikük más módon manipulálja az adatokat az ideiglenes táblán keresztül.

### READ (Olvasás)

Az első kódrészlet létrehoz egy ideiglenes táblát (#temp\_szoba), és az adatokat feltölti a szoba táblából. Ezután kiírja az összes rekordot az ideiglenes táblából a konzolra. Ez a művelet lehetővé teszi az adatok megtekintését a kiválasztott táblából anélkül, hogy közvetlenül az adatbázist érintené.

# Cursor objektum létrehozása SQL lekérdezésekhez

cursor = conn.cursor()

# READ - Temporary table létrehozása és adatokkal feltöltése

cursor.execute("SELECT \* INTO #temp\_szoba FROM szoba")

# Adatok kiírása

cursor.execute("SELECT \* FROM #temp\_szoba ")

# Output kiírása

print("Adatok:")

for row in cursor.fetchall():

print(row)

### DELETE (Törlés)

A második kódrészlet ugyanazt a folyamatot végzi el, mint az olvasásnál, létrehoz egy ideiglenes táblát és feltölti adatokkal. Ezután törli azokat a rekordokat az ideiglenes táblából, amelyeknek a szoba\_id mezője 1-es értéket vesz fel.

# Temporary table-ből egy sor törlése

cursor.execute("DELETE FROM #temp\_szoba WHERE szoba\_id = 1")

### CREATE (Létrehozás)

A harmadik kódrészlet ismét létrehoz egy ideiglenes táblát és feltölti adatokkal. Ezután beszúr egy új sort az ideiglenes táblába az adatok megadásával.

# Temporary table-ba egy sor beszúrása

cursor.execute("INSERT INTO #temp\_szoba (szoba\_id, szoba\_szama, ferohely, potagy, klimas, szallas\_fk) VALUES (1000, 1000, 10, 1, 'n', 1000)")

### UPDATE 1. (Frissítés)

Ez a kódrészlet frissíti a kiválasztott rekordot az ideiglenes táblában a megadott feltételek szerint, vagyis módosítja a 2-es szoba\_id férőhely mezőjét.

# Update - Egy sor frissítése a temporary table-ben.

cursor.execute("UPDATE #temp\_szoba SET ferohely = 100 WHERE szoba\_id = 2")

### UPDATE 2. (Frissítés)

Az utolsó kódrészlet a 3-as szoba azonosítóval rendelkező sor értékeit változtatja.

# Update - Egy sor frissítése a temporary table-ben.

cursor.execute("UPDATE #temp\_szoba SET ferohely = 20, potagy = 20, klimas = 'n' WHERE szoba\_id = 3")

## Ábrázolás, diagrammok

Az adatok megjelenítésére és különböző lekérdezések vizualizációjára készítettünk egy oszlop, valamint egy kördiagrammot.

### Különböző férőhelyes szobák száma

Az alábbi oszlopdiagram megmutatja, hogy milyen szálláshely adatbázisban lévő különböző férőhelyű szobák eloszlása.

Először lekértük az adatokat az adatbázisból, beolvastuk egy pandas DataFrame-be, majd ezeket egy oszlopdiagramon jelenítettük meg matplotlib segítségével.

import pyodbc

import pandas as pd

import matplotlib.pyplot as plt

import sqlite3

Ezután a query változóban egy SQL lekérdezést definiáltunk, amely az adatbázisból kiválasztja a szoba táblából az egyes szobatípusokhoz tartozó férőhelyek számát és ezek előfordulási gyakoriságát.

query = '''

SELECT ferohely, COUNT(\*) AS count

FROM szoba

GROUP BY ferohely

'''

A következő lépésben a pd.read\_sql() függvény segítségével végrehajtottuk az SQL lekérdezést és beolvastuk az eredményeket egy pandas DataFrame-be df változó név alatt. Ez az oszlopdiagram elkészítéséhez szükséges adathalmaz.

# Lekérdezés végrehajtása és eredmények lekérése

df = pd.read\_sql(query, conn)

Miután beolvastuk az adatokat, bezártuk a kapcsolatot az adatbázissal a conn.close() függvénnyel.

A plt.bar() függvény segítségével létrehoztunk egy oszlopdiagramot. Az x tengelyen a szobák férőhelyeit, míg az y tengelyen az adott férőhelyszámhoz tartozó szobák számát ábrázoltuk. Címkéket adtunk az x és y tengelyeknek, valamint a diagram címét is beállítottuk, végül pedig a `plt.show()` függvénnyel megjelenítettük az oszlopdiagramot.

# Adatok ábrázolása

plt.bar(df['ferohely'], df['count'])

plt.xlabel('Férőhely')

plt.ylabel('Szobák száma')

plt.title('Különböző férőhelyes szobák száma')

plt.show()

A képen szöveg, képernyőkép, diagram, sor látható

Automatikusan generált leírás

A diagrammon jól látszik, hogy kettő, illetve három férőhelyes szobák közül bőven válogathat a vendég. Viszont egy és négy fős szobát mér nehezebb találni.

### Klímás szobák aránya

Ez a kódrészlet nagyban hasonlít az előzőhöz felépítését illetően.

Definiáltuk az SQL lekérdezést, amely a szoba táblából származó adatokat használja fel, és megadja, hogy mennyi klímás és nem klímás szoba található az szálláshely adatbázisban.

# SQL lekérdezés definiálása

query = '''

SELECT klimas, COUNT(\*) AS count

FROM szoba

GROUP BY klimas

'''

Ezt követően végrehajtottuk a lekérdezést, majd az eredményeket egy DataFrame-be tároltuk.

# Lekérdezés végrehajtása és eredmények lekérése

df = pd.read\_sql(query, conn)

A kapcsolat bezárása után kördiagrammon ábrázoltuk az adatokat.

# Adatok ábrázolása

plt.pie(df['count'], labels=df['klimas'], autopct='%1.1f%%', startangle=140)

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.title('Klímás szobák aránya')

plt.show()

A képen szöveg, képernyőkép, diagram, kör látható

Automatikusan generált leírás

A diagramm megmutatja, hogy a szobák majdnem háromnegyede rendelkezik valamilyen klímaberendezéssel, ha egy kényesebb vendég például nyáron utazik, nagy eséllyel talál majd az igényeinek megfelelő hűtött szobát.

*Ez a csoportos projektmunkánk, sokat dolgoztunk vele, köszönjük a figyelmet.*